ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

GitLab integration plugin pro systém Moodle

David Hornof
Softwarové inženýrství a technologie

Květen 2019
Vedoucí práce: Ing. Pavel Náplava, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hornof** Jméno: **David** Osobní číslo: **466001**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Gitlab integration plugin pro systém Moodle

Název bakalářské práce anglicky:

Gitlab integration plugin for Moodle system

Pokyny pro vypracování:

Navrhněte a implementujte plugin, který umožní integrovat systémy Gitlab a Moodle. Požadavky na plugin:
• Plugin umožní učitelům zjednodušenou tvorbu a správu kurzů v systému Moodle, bez nutnosti využívání uživatelského rozhraní Moodle, za pomocí git rozhraní.
• Plugin bude překládat html, markdown soubory v repozitáři git do stránek v kurzu a dokumentové soubory budou do kurzu vkládány jako soubory ke stažení.
• Plugin umožní učitelům editovat kurz offline a nahrávát nové verze pomocí ‚git push'
Postupujte následujícím způsobem:
1) Seznamte se se systémy Moodle, Gitlab, Git.
2) Analyzujte možnosti napojení Gitlab repozitářů na systém Moodle.
3) Navrhněte, implementujte a uživatelsky otestujte nový plugin, který bude napojený Gitlab repozitář překládat na kurzy v systému Moodle.

Seznam doporučené literatury:

[1] HODSON, Ryan. Ry's Git Tutorial. RyPress, 2014. ISBN 978-0-9850-7231-5.
[2] CHACON, Scott a Ben STRAUB. Pro Git: [everything you need to know about the Git distibuted source control tool]. 2nd ed. New York, NY: Apress, 2014. ISBN 978-1-4842-0076-6.
[3] WILD, Ian. Moodle 3.x Developer's Guide. Birmingham: Packt Publishing, 2017. ISBN 978-1-7864-6711-9.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, Ph.D., katedra ekonomiky, manažerství a humanitních věd FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.
Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování / Prohlášení

Rád bych tímto poděkoval Ing. Pavlu Náplavovi, Ph.D. za jeho čas, cenné rady a profesionální přístup, se kterým tuto práci vedl.

Také děkuji Bc. Jiřímu Fryčovi za poskytnuté technické konzultace a součinnost ze strany Centra znalostního managementu.

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

V Praze dne

.....

Abstrakt / Abstract

Tato bakalářská práce se nejprve zabývá analýzou systémů Moodle, Git a GitLab se zaměřením to, jak tyto systémy pracují s daty. Na základě toho je v další části práce navržen plugin pro systém Moodle umožňující import obsahu Git repozitáře hostovaného v prostředí aplikace GitLab do obsahu kurzu v Moodle. Součástí návrhu je také funkce importu s automatickým sledováním změn, který za pomoci webhooku v aplikaci GitLab dovoluje automatickou aktualizaci importovaného obsahu při změnách ve zdrojovém repozitáři. Následně se práce věnuje popisu realizace tohoto návrhu, což zahrnuje popis implementace, ale také instalace, konfigurace a používání tohoto pluginu z pohledu uživatele. V závěru je popsán průběh testování implementovaného pluginu, na kterém se podílel i jeho budoucí uživatel. Práce se také pozastavuje nad budoucím rozvojem tohoto pluginu.

Klíčová slova: Moodle, plugin, Git, GitLab, integrace, REST, API, webservice

This bachelor thesis opens up with an analysis of Moodle, Git and GitLab that is focused on how these systems handle their data. In the next part of the thesis, a plugin for Moodle that can import files from a Git repository hosted in GitLab to a content of a Moodle course is designed. This plugin also features an import with automatic updates of imported content after a change in the source repository using GitLab webhook. Next comes a chapter dedicated to implementation followed by an installation and configuration as well as user's guide. Finally, testing of the plugin that was conducted is described, which includes testing done with plugin's future user. Further development of the plugin is also discussed.

Title translation: GitLab integration plugin for Moodle system

Keywords: Moodle, plugin, Git, GitLab, integration, REST, API, webservice

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Jedním z dlouhodobých cílů Fakulty elektrotechnické ČVUT je sjednocení systémů pro podporu výuky, zejména pak, aby se studijní materiály a informace o předmětech nacházely na jednom místě. V současné době fungují v tomto ohledu na Fakultě dva velké systémy: Moodle a Courseware. Centrum znalostního managementu, s jehož spoluprací tato práce vznikla, však preferuje systém Moodle, mimo jiné kvůli množství funkcionalit a jeho poměrně snadné rozšiřitelnosti. Krom těchto systémů však někteří vyučující poskytují materiály a informace o svých předmětech na vlastních stránkách.

Důvodů, proč množství vyučujících nevyužívá systému Moodle, je celá řada. Patří mezi ně zejména potřeba možnosti spravovat obsah kurzu bez připojení k internetu, možnost používání vlastních dobře známých programů pro tvorbu obsahu a netrávit „zbytečně“ čas učením se pracovat s novým systémem (kterým je právě Moodle).

Na základě těchto potřeb uživatelů je cílem této práce vytvořit podpůrný systém, jenž umožní svému uživateli naplnit kurz v systému Moodle obsahem, který existuje mimo něj. Konkrétně půjde o obsah v Git repozitáři, jelikož takový repozitář je využitelný k uchovávání souborů a jeho velkou výhodou je možnost verzování těchto souborů. Tento repozitář se bude nacházet v prostředí aplikace GitLab (hostované na Fakultě elektrotechnické), protože se jedná o standardní způsob uchovávání Git repozitářů souvisejících s prací a výukou na Fakultě. Prostřednictvím tohoto repozitáře budou uživatelé vytvářet a upravovat soubory a složky představující samotný obsah kurzu podle svých potřeb.

Toto řešení bude odpovědí na uvedené problémy řady vyučujících, kteří zatím na Moodle nepřešli – výrazně se pomocí něho redukuje nutnost práce se systémem Moodle pouze na použití navrhovaného podpůrného systému, bude umožněna práce na obsahu kurzu offline a za pomoci uživatelem zvolených nástrojů (uživatel vytváří obsah lokálního Git repozitáře, který poté synchronizuje s verzí v prostředí GitLabu) a systém Git navíc umožní verzování vytvářeného obsahu.

Hlavním přínosem tohoto podpůrného systému tedy bude usnadnění přesunu materiálů k dalším předmětům do systému Moodle, čímž se podpoří výše zmíněný dlouhodobý cíl Fakulty.

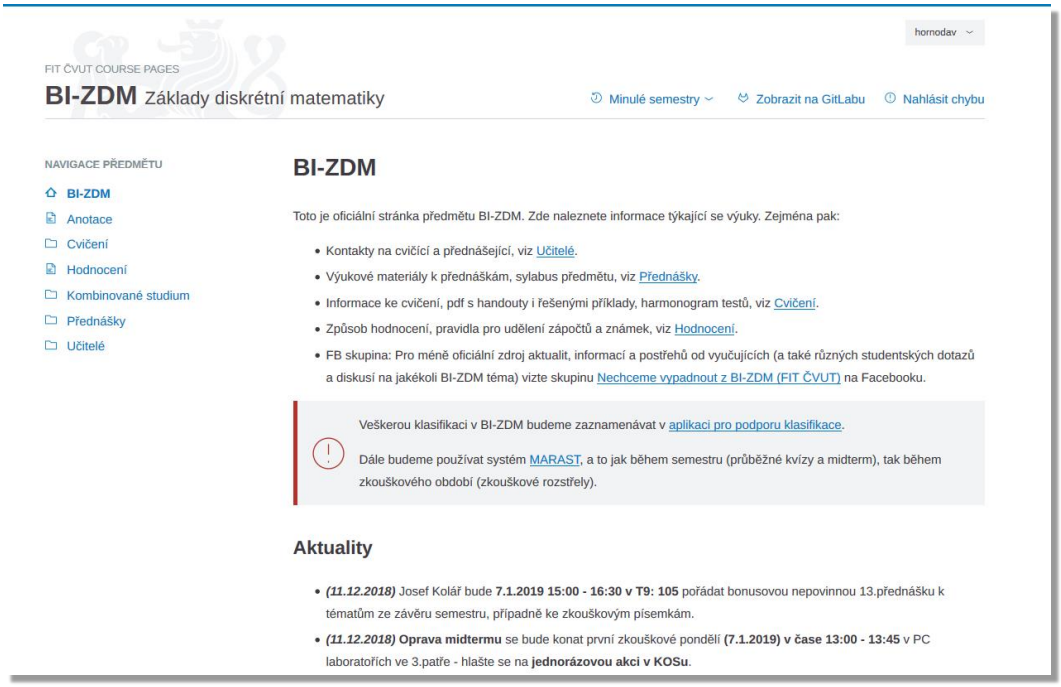
1.1 Struktura práce

Tato práce se nejprve v kapitole 2 zabývá analýzou zaměřenou na rozbor existujících aplikací a možnosti jejich propojení. Výstupem této analýzy jsou konkrétní požadavky na navrhovaný systém. Další částí práce je kapitola 3, ve které je proveden návrh systému a kapitola 4 popisující implementaci tohoto návrhu. Následuje kapitola 5, která se věnuje tomu, jak vytvořený systém nainstalovat a nakonfigurovat. Kapitola 6 pak popisuje používání systému z hlediska uživatele. Kapitola 7 se zabývá tím, jak byl systém testován a kapitola 8 se věnuje současnému stavu a budoucímu rozvoji systému. V závěrečné kapitole 9 je provedeno zhodnocení celé práce.

1.2 Systém Course Pages

Poznámka: V prostředí ČVUT již existuje systém s funkcionalitou, která je podobná navrhovanému systému. Je jím systém Course Pages používaný na Fakultě informačních technologií. Ten z obsahu v Git repozitáři vytváří jednoduché webové stránky, na kterých se nachází informace a studijní materiály pro daný předmět. Tím se zásadně liší od navrhovaného systému, který bude na základě obsahu Git repozitáře plnit kurz v systému Moodle obsahem namísto vytváření samostatných stránek.

Na obrázku 1.1 je možné vidět ukázku stránek předmětu vygenerovaných tímto systémem. Stránky kopírují strukturu (obsah) zdrojového repozitáře, což lze pozorovat v navigačním menu v levé části. V tomto ohledu se systém Course Pages podobá navrhovanému systému, který bude obsah zdrojového repozitáře reprezentovat v kurzu v systému Moodle.



Obrázek 1.1. Stránky předmětu vygenerované systémem Course Pages.

Kapitola 2
Analýza

Tato kapitola se zabývá analýzou existujících aplikací, se kterými bude navrhovaný systém spolupracovat, zejména pak jejich částí vztahujících se k práci s daty a možnostem integrace s externími systémy. V rámci této analýzy jsou prezentovány i poznatky z průzkumu využití těchto aplikací v prostředí Fakulty elektrotechnické ČVUT. V závěru kapitoly jsou definovány konkrétní požadavky na navrhovaný systém.

2.1 Existující aplikace

V této části se nachází stručný popis existujících aplikací dotýkajících se navrhovaného systému, kterými jsou Moodle, Git a s ním související GitLab.

2.1.1 Moodle

Moodle je webová aplikace určená primárně pro e-learning. Na Fakultě elektrotechnické má své využití jakožto jeden z řady nástrojů pro podporu výuky, kde slouží hlavně jako prostředek, pomocí kterého vyučující poskytují svým studentům výukové materiály. Moodle ale nabízí mnoho dalších funkcí, jako je evidence docházky, známkování nebo zadávání testů a domácích úkolů. Krom toho je tato aplikace navržena modulárně, díky čemuž podporuje rozšiřování svých funkcionalit formou pluginů. Většina základních funkcí aplikace je též realizována formou pluginů, které jsou přítomné už při instalaci aplikace. Moodle lze proto rozdělit do dvou částí: *Moodle core*, představující společný základ pro pluginy a samostatné pluginy, které společnému základu dodávají funkcionality. [1–2]

Krom toho Moodle pro možnosti integrace s externími aplikacemi poskytuje vlastní REST API, které mohou pluginy též rozšiřovat a vystavovat tak vlastní funkce. [3–5]

2.1.2 Git

Git je distribuovaný systém pro správu verzí (verzovací systém), původně vytvořený Linusem Torvaldsem pro správu zdrojových kódu jádra operačního systému Linux. Jeho úkolem je tedy převážně uchovávání a správa historie zdrojových kódů, respektive textových souborů. Systém však zvládá i práci s binárními soubory. Systém Git lze používat pomocí programu `git`, dostupného pro všechny běžné platformy. [6–7]

Základním prvkem při práci s tímto systémem je repozitář. Repozitář je místo, ve kterém jsou uloženy soubory včetně jejich historie v podobě takzvaných snapshotů (*snapshots*). Přidání souborů (či zanesení změn provedených v souborech) do repozitáře se skládá ze dvou fází: *stage* a *commit*. Fáze *stage* se provádí příkazem `git add` a jejím výsledkem je snapshot obsahující aktuální stav souborů. Dalším krokem je *commit*, prováděný příkazem `git commit`, který vytvořený snapshot přidá do repozitáře. [6]

Další důležitou součástí systému Git jsou větve (*branches*). Jak název napovídá, jedná se o „větve“ v historii repozitáře, jejichž obsahem je alternativní historie daného repozitáře. Hlavní větev repozitáře se nazývá *master* a lze se od ní v libovolné chvíli

odchýlit – vytvořit novou větev; stejně tak se lze v libovolné chvíli odchýlit i od jakékoliv jiné větve. Odchýlenou větev pak lze spojit zpět do větve hlavní, obdobně jako lze dvě odchýlené větve spojit do sebe. [6, 8]

Jelikož je Git distribuovaný systém, obsahuje také funkce pro práci se vzdálenými repozitáři, nazývaných *remote repositories* – zkráceně *remotes*. Důležitou operací pro práci s nimi je klonování (`git clone`), které udělá kompletní kopii vzdáleného repozitáře. Při práci s tímto naklonovaným repozitářem jsou pak významné operace `git pull` a `git push`. *Pull* je operací, která promítne změny provedené ve vzdáleném repozitáři do repozitáře lokálního. Operace *push* je pravým opakem – slouží k přenesení změn provedených v lokálním repozitáři do repozitáře vzdáleného. [6, 8]

Důležitým poznatkem z této části je, jakým způsobem systém Git nakládá se soubory. Konkrétně, že existuje hierarchie **repozitář** → **větev** → **commit**, tudíž pokud chci přistupovat k souboru ve verzovacím systému Git, musím mít přístup k repozitáři, v rámci repozitáře znát větev, ze které chci číst, a v rámci ní přistupovat ke konkrétnímu commitu.

2.1.3 GitLab

GitLab je webová aplikace disponující mnoha funkcemi sloužících pro podporu činností spojených s vývojem softwaru. Nejdůležitější z nich je možnost hostování (uchovávání) a správy Git repozitářů, což znamená, že se aplikace dovede chovat jako vzdálený Git repozitář popsaný v části 2.1.2. Zajímavou funkcí s tím související je *webhook*. Jde o funkci dovolující automaticky reagovat na události, které v hostovaném repozitáři nastanou, mimo jiné i na události typu push. Webhook tedy dovoluje automaticky reagovat na změny obsahu v hostovaném repozitáři. [9–10]

Z hlediska terminologie se v aplikaci GitLab rozlišují pojmy repozitář a projekt. Repozitářem se rozumí pouze konkrétní Git repozitář, zatímco projektem se rozumí repozitář včetně dodatečných funkcí poskytovaných aplikací GitLab, jako jsou například issues nebo zmíněný webhook. [11]

Pro integraci s externími aplikacemi nabízí GitLab rozsáhlé REST API, pomocí kterého lze přistupovat k prakticky každé součásti této aplikace. [12]

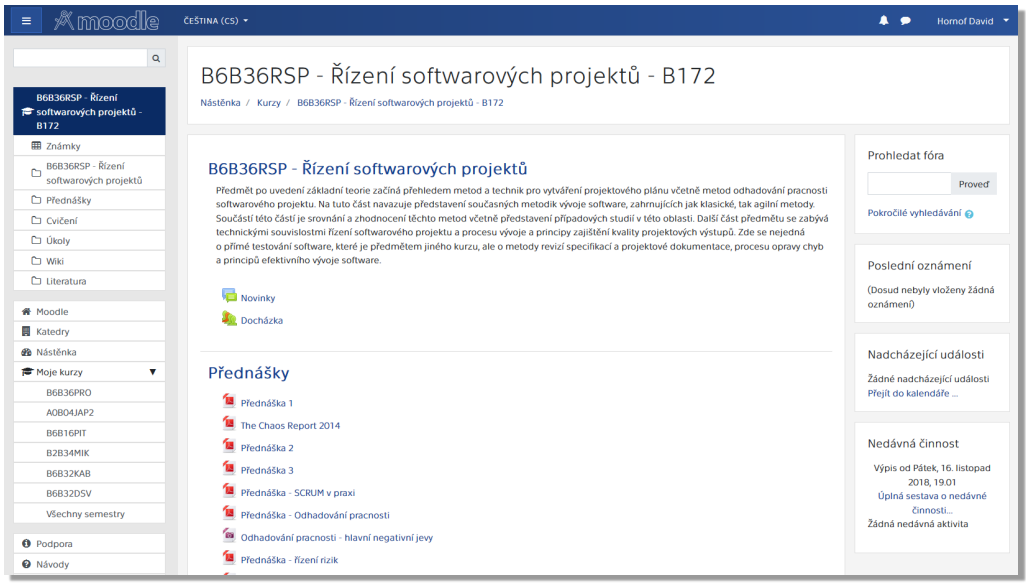
2.2 Kurzy v aplikaci Moodle

Vytváření kurzů je jednou z nejdůležitějších funkcí aplikace Moodle. Kurz je jakýsi virtuální prostor obsahující prostředky pro výuku, které vyučující (respektive správci kurzu) poskytují svým studentům. Ti následně tyto prostředky využívají pro vzdělávání se. [13] Ukázku kurzu v aplikaci Moodle je možné vidět na obrázku 2.1.

Základními stavebními prvky kurzu jsou hlavní stránka a sekce. Sekce jsou uživatelsky či automaticky definované části hlavní stránky. Jejich význam spočívá zejména v tom, že se promítají do navigačního menu. Minimální kurz se skládá z hlavní stránky a úvodní sekce. Tyto části jsou přítomné v každém kurzu a není možné je odstranit. Hlavní stránka je výchozím zobrazením pro kurz, tedy tím, co se zobrazí ve chvíli, kdy uživatel kurz otevře. Úvodní sekce je první sekce této stránky, její obsah a nadpis lze libovolně měnit. [14]

Sekce se skládají z nadpisu, textového popisu a takzvaných činností (*activities*) a studijních materiálů (*resources*). Činnosti a aktivity jsou souhrnně označovány jako moduly[1] (*activity modules*). Činností je většinou prvek umožňující interakci vyučujícího

[1] Modul (*activity module*) je typ pluginu, který dovoluje přidávání svých instancí do kurzů. Tyto instance jsou jednotlivé činnosti a studijní materiály. [2, 15]



Obrázek 2.1. Kurz v aplikaci Moodle Fakulty elektrotechnické ČVUT.

a studenta, oproti tomu studijní materiál představuje pouze statický obsah vytvořený vyučujícím. Na FEL ČVUT jsou obvykle používanými činnostmi zejména testy a domácí úkoly, zatímco obvyklými studijními materiály jsou hlavně soubory a statické stránky. Dále je důležité poznamenat, že instance modulů nelze skládat do sebe, takže kupříkladu na stránce nemůže být umístěna žádná další činnost ani studijní materiál.[1] Všechny instance modulů v rámci kurzu jsou proto dostupné ze sekcí na hlavní stránce tohoto kurzu. [15–17]

Konkrétní struktura kurzu pro předmět není pevně daná a liší se dle potřeb a preferencí vyučujícího. V prostředí FEL ČVUT (v existujících Moodle kurzech, ale i v jiných prostředcích pro podporu výuky, zejména Courseware FEL a webových stránkách vyučujících z katedry matematiky) lze však identifikovat části využívané ve většině kurzů:

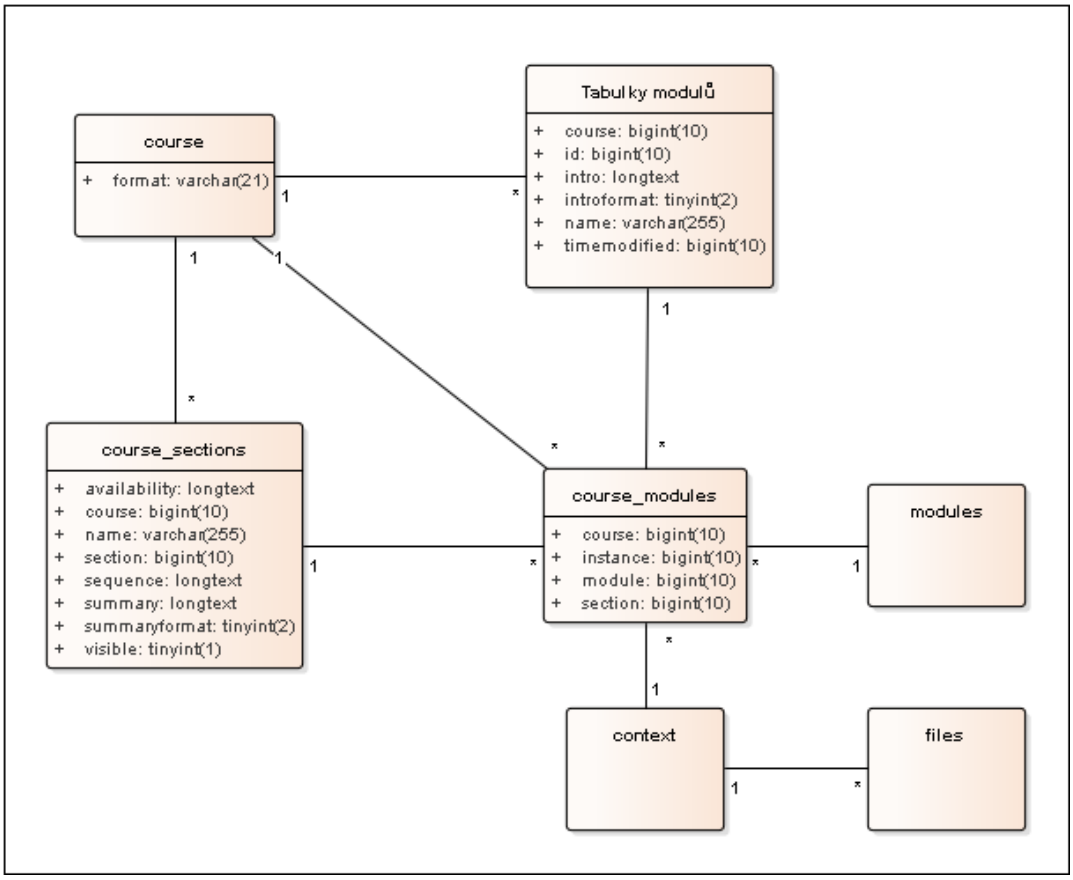
- **Úvod:** synopse předmětu, obecné informace.
- **Podmínky ukončení předmětu:** požadavky na zápočet, informace o zkoušce a celkovém hodnocení předmětu.
- **Přednášky:** harmonogram přednášek, soubory s prezentacemi a jinými materiály použitých v jednotlivých přednáškách.
- **Cvičení:** harmonogram cvičení, soubory s materiály pro jednotlivá cvičení.
- **Literatura:** seznam doporučené literatury.

Další funkcí využívanou také nejen v Moodle kurzech je vytváření obsahu skrytého před studenty a dostupného pouze vyučujícím.

2.3 Data v aplikaci Moodle

Aplikace Moodle používá pro persistenci dat, se kterými pracuje, dva prostředky: relační databázi a souborové úložiště. V souborovém úložišti se nachází uživateli nahrané

[1] Lze však na již vytvořené aktivity a studijní materiály (instance modulů) odkazovat. Krom toho některé moduly dovolují v rámci svých instancí nahrání souborů, na které lze také odkazovat. Takto nahrané soubory ovšem nejsou instancemi výchozího modulu pro nahrávání souboru.



Obrázek 2.2. Diagram vzájemného vztahu tabulek z databáze aplikace Moodle významných pro navrhovaný systém.

soubory. Vše ostatní, zejména uživatelské nastavení a obsah kurzů je uloženo v databázi. Obrázek 2.2 obsahuje diagram vytvořený autorem textu za účelem ilustrace[1] části této databáze, která je významná pro navrhovaný systém. [1, 18]

Data kurzů se ukládají do tabulky `course`. Jelikož navrhovaný systém nebude kurzy vytvářet, je pro něj z této tabulky významný hlavně sloupec `format`, který určuje typ zobrazení kurzu (v terminologii Moodle nazývaný jako formát kurzu – *course format*). Jedním ze základních, a zároveň na FEL ČVUT často využívaným, typů zobrazení je `topics` formát, pro který hodnota tohoto sloupce nabývá hodnoty `topics`. Tento formát kurzu na hlavní stránce zobrazuje pouze uživatelem definované sekce. Příkladem jiného formátu může být `weekly format`, který automaticky vytváří sekce pro každý týden doby trvání kurzu a zároveň zvýrazňuje sekci pro aktuální týden. [19]

Data vztahující se k sekcím hlavních stránek kurzů se ukládají do databázové tabulky `course_sections`, ve které se nachází celá řada sloupců důležitých pro navrhovaný systém:

- `name`: název (nadpis) sekce,
- `summary`: text (obsah) sekce,
- `summaryformat` identifikátor formátu, ve kterém je uložen text ve sloupci `summary`,
- `course`: identifikátor kurzu, ke kterému se sekce váže (cizí klíč odkazující na sloupec `id` v tabulce `course`),
- `section`: pořadí sekce v kurzu, číslováno od 0 (0 je úvodní sekce),

[1] Nejedná se tedy o přesné či úplné schéma.

- `sequence`: sloupec obsahující identifikátory instancí modulů (hodnoty sloupce `id`) z tabulky `course_module` oddělené čárkou v takovém pořadí, v jakém se mají v sekci zobrazovat,
- `visible`: nabývá hodnoty 1, pokud je sekce viditelná uživatelům bez oprávnění spravovat kurz a hodnoty 0 pokud ne,
- `availability`: obsahuje podmínky dostupnosti sekce pro uživatele bez oprávnění spravovat kurz ve formátu JSON.

Data představující instance modulů v sekcích (a tedy zároveň v kurzech) se skládají ze dvou částí. První částí jsou záznamy v tabulce `course_modules`. V této tabulce jsou významné zejména následující sloupce:

- `course`: identifikátor kurzu, ke kterému instance patří (cizí klíč odkazující na sloupec `id` v tabulce `course`),
- `module`: identifikátor modulu, kterému instance patří (cizí klíč odkazující na sloupec `id` v tabulce `modules`),
- `instance`: identifikátor instance modulu (cizí klíč odkazující na sloupec `id` v hlavní tabulce, kterou modul používá pro ukládání dat o svých instancích),
- `section`: identifikátor sekce, ve které se tato instance má zobrazovat (cizí klíč odkazující na sloupec `id` v tabulce `course_section`).

Druhou částí jsou pak záznamy v hlavní tabulce, kterou modul využívá k uchovávání informací o svých instancích. Obecně platí, že pluginy (nejen typu modul) využívají k ukládání svých dat přímo databázi aplikace Moodle – podle svého typu si vytvoří vlastní tabulku (či tabulky) nebo využívají již existujících tabulek. Hlavní tabulka pro každý plugin může být pouze jedna a musí nést stejný název, jako plugin samotný. Schéma hlavní tabulky (a případných dalších tabulek) se liší podle potřeb pluginu. Pro některé typy pluginů však Moodle hlavní tabulku vyžaduje, jelikož definuje sloupce, které v ní očekává. Tak je tomu v případě modulů, pro které jsou očekávány následující sloupce: [15]

- `id`: primární klíč tabulky (identifikátor záznamu),
- `course`: identifikátor kurzu, ke kterému instance modulu patří (cizí klíč odkazující na sloupec `id` v tabulce `course`),
- `name`: název instance,
- `timemodified`: časová značka poslední změny instance,
- `intro`: text popisu instance,
- `introformat`: identifikátor formátu, ve kterém je uložen text ve sloupci `intro`.

Důležitou záležitostí spjatou s vytvářením instancí modulů je kontext. Kontext je součástí poměrně složitého systému pro přiřazování a kontrolu uživatelských oprávnění v aplikaci Moodle. Jedná se o abstrakci pro součást této aplikace, kterou může uživatel používat. Moodle rozeznává šest typů kontextů, jedním z nichž je `module`, který představuje právě instanci modulu. Každá instance modulu musí mít přiřazenou instanci kontextu. Toto přiřazení je realizováno tabulkou `context`. [20–23]

V neposlední řadě za zmínku stojí tabulka `files`, v níž jsou uchované informace o souborech nacházejících se v souborovém úložišti. K přiřazení souboru ze souborového úložiště ke konkrétní instanci modulu slouží kontext. [24]

2.3.1 Podporované textové formáty

Aplikace Moodle je schopná pracovat s textovým obsahem v několika různých formátech, kterými jsou:

- **Moodle:** Vlastní formát aplikace Moodle. Jedná se o variantu formátu HTML, která není přesně zdokumentovaná.
- **HTML:** Aplikace Moodle vypíše obsah na stránku tak, jak je uložený, bez jakýchkoli úprav.
- **Prostý text** (*plain text*): Aplikace Moodle před výpisem převede všechny znaky mající význam pro jazyk HTML na HTML entity tak, aby se správně zobrazily.
- **Markdown:** Aplikace Moodle před výpisem převede formát Markdown na formát HTML, ponechá při tom však bez dotknutí všechny validní sekvence HTML kódu (tagy a entity).

Identifikátory těchto formátů jsou uvedeny v tabulce 2.1.

Formát	Identifikátor
Moodle	0
HTML	1
Prostý text	2
Markdown	4

Tabulka 2.1. Identifikátory textových formátů podporovaných aplikací Moodle.

2.4 Obecné požadavky na systém

Na základě informací z předešlých částí je možné definovat obecné požadavky kladené na navrhovaný systém.

1. **Import kurzu:** Systém bude na základě souborů v Git repozitáři hostovaném v prostředí aplikace GitLab schopný naplnit kurz v aplikaci Moodle obsahem (importovat kurz), protože to je jeho účel.
2. **Uživatelské rozhraní:** Aby systém mohl importovat kurz ze zdrojového repozitáře, musí uživateli umožnit vybrat si repozitář, jehož obsah chce uživatel importovat.
3. **Propojení s aplikací GitLab:** Systém musí být schopný číst soubory v Git repozitáři hostovaném v prostředí aplikace GitLab, aby je mohl zpracovat.
4. **Propojení s aplikací Moodle:** Účelem systému je import Moodle kurzu. Aby toho mohl docílit, musí být vhodně napojen na aplikaci Moodle, respektive na její datovou vrstvu. Toto napojení navíc musí být realizováno korektně, aby nedošlo k problémům s funkčností aplikace Moodle.
5. **Tvorba strukturovaného kurzu:** Systém musí umět obsah kurzu vytvořit ve vhodně strukturované podobě, aby byla zachována uživatelská přívětivost při následném používání importovaného kurzu. Vhodně strukturovaná podoba znamená minimálně přítomnost sekcí na hlavní stránce kurzu.
6. **Automatická aktualizace obsahu:** Vzhledem k tomu, že zdrojem pro import je Git repozitář, bylo by pro zvýšení uživatelské přívětivosti vhodnou funkcionalitou využít schopnosti aplikace GitLab reagovat na události typu push v hostovaném repozitáři a bez nutnosti uživatelova zásahu automaticky promítnout změny provedené ve zdrojovém repozitáři do importovaného kurzu.

2.5 Možnosti propojení existujících aplikací

Jak je patrné z předešlých částí, klíčovým aspektem navrhovaného systému je propojení aplikací GitLab a Moodle. Z části 2.3 dále vyplývá silný požadavek na velmi úzké

napojení na aplikaci Moodle, respektive na její datovou vrstvu, se kterou navíc musí být zacházeno korektně, aby byla zajištěna správná funkčnost importovaných kurzů a zároveň nebyla narušena funkčnost ostatních kurzů či nedošlo k narušení funkčnosti aplikace Moodle jakožto celku.

Pokud jde o aplikaci GitLab, zde systém vyžaduje možnost čtení souborů a informací o zdrojovém repozitáři. Dále pak pro uspokojení požadavku na automatické reakce na změny v tomto repozitáři definovaném v části 2.4 je potřeba i přístup ke změně nastavení repozitáře pro nastavení webhooku popsaném v části 2.1.3.

2.5.1 Napojení na Moodle

Pro napojení na Moodle se jeví jako velmi vhodná možnost využít podpory pro pluginy, kterou tato aplikace nabízí. Pluginům je poskytováno bohaté aplikační rozhraní (API), mimo jiné pro manipulaci s datovou vrstvou (databází i souborovým úložištěm) a práci s kontexty. Dále by realizace systému jakožto pluginu umožnila integraci uživatelského rozhraní systému přímo do aplikace Moodle. [2]

Vhodné je také zmínit, že Moodle rozlišuje velkou škálu typů pluginů, které se mezi sebou liší podle svého účelu a zamýšlených funkcionalit. To znamená rozdíl hlavně v tom, jakým způsobem jsou uživateli prezentovány a jak s ním interagují. API poskytovaná pluginům se neliší v závislosti na jejich typu. [2, 25]

2.5.2 Napojení na GitLab

Jak je zmíněno v části 2.1.3, aplikace GitLab poskytuje pro potřeby integrace s externími aplikacemi REST API. Součástí tohoto API jsou mimo jiné metody pro získávání informací a změny nastavení hostovaných repozitářů, stejně tak jako metody, které přímo dovolují z repozitářů číst soubory. [26–27]

Jelikož navrhovaný systém potřebuje přístup ke čtení dat z repozitářů uživatelů pomocí API, stejně tak jako možnost změnit konfiguraci projektu v případě potřeby (nastavení webhooku), je potřeba vyřešit problém autentizace, tedy získání práv pro přístup. Aplikace GitLab nabízí několik možností, jak uživatele autentizovat: [12]

- **OAuth2:** autentizace pomocí protokolu OAuth2. [28] Tuto metodu autentizace instance aplikace GitLab patřící Fakultě elektrotechnické nepodporuje a nelze ji tudíž použít.
- **Personal access token:** jedná se o speciální kód (nazývaný *token*), který si uživatel může v aplikaci GitLab vygenerovat. Tento token nahrazuje kombinaci jeho uživatelského jména a hesla, zejména pro použití GitLab API. Je možné omezit rozsah (*scope*) tohoto tokenu, tedy to, k jakým částem aplikace dovoluje přístup. Pro účely čtení informací o repozitářích, změny nastavení a čtení souborů z repozitářů pomocí API je nutný scope `api`. [29]
- **Session cookie:** autentizace pomocí HTTP session cookie. Tato metoda je užitečná pro testování API v internetovém prohlížeči, není však využitelná pro externí aplikaci vzhledem k tomu, že neexistuje vhodný způsob, jak cookie vygenerovat. [12]
- **Deploy token:** podobně jako personal access token se jedná o speciální kód, který si uživatel může v aplikaci GitLab vygenerovat, vztahuje se však pouze ke konkrétnímu repozitáři. Tento kód nahrazuje jeho heslo, je ale použitelný pouze pro naklonování repozitáře, ke kterému je navázaný, pomocí programu `git` (operace `git clone`). [30]
- **Deploy key:** autentizace pomocí SSH klíče, který nahrazuje uživatelské jméno a heslo. Tato metoda je použitelná pouze pro přístup (ke čtení i k zápisu v závislosti na uživatelském nastavení) k repozitáři pomocí programu `git`. [31–32]

Z toho plyne, že jediným použitelným způsobem, jak se z externí aplikace autentizovat pro použití API aplikace GitLab Fakulty elektrotechnické ČVUT, je využít personal access token.

2.6 Konkrétní požadavky na realizaci pluginu

Na základě analýzy provedené v této kapitole byly obecné požadavky kladené na realizaci navrhovaného systému z části 2.4 konkretizovány následovně:

2.6.1 Funkční požadavky

1. **Uživatelské rozhraní:** Aby uživatelé mohli systém používat, bude poskytovat uživatelské rozhraní.
2. **Import kurzu:** Systém bude na základě souborů v Git repozitáři hostovaném v prostředí aplikace GitLab schopný naplnit kurz v aplikaci Moodle obsahem (importovat kurz), protože to je jeho účel.
3. **Tvorba strukturovaného kurzu:** Aby byla zachována uživatelská přívětivost při používání importovaného kurzu v aplikaci Moodle, bude systém importovat kurzy ve vhodně strukturované podobě, což znamená, že hlavní stránka kurzu bude rozdělena na sekce.
4. **Tvorba skrytých sekcí:** Systém bude v rámci importu umožňovat tvorbu sekcí skrytých před studenty kurzu, protože se jedná o funkci využívanou vyučujícími.
5. **Import souborů a složek:** Protože v prostředí FEL ČVUT jsou nejčastěji poskytovaným studijním materiálem soubory, bude je systém umožňovat do kurzu importovat stejně tak jako složky, které jsou přímo spjaté s organizací souborů.
6. **Import stránek:** Jelikož statická stránka je v prostředí FEL ČVUT často využívaným a užitečným studijním materiálem, bude je systém umožňovat importovat do kurzu.
7. **Reakce na změny ve zdrojovém repozitáři:** Pro zvýšení své uživatelské přívětivosti bude systém při změně dat ve zdrojovém repozitáři umožňovat tyto změny automaticky přenést do importovaného kurzu.

2.6.2 Nefunkční požadavky

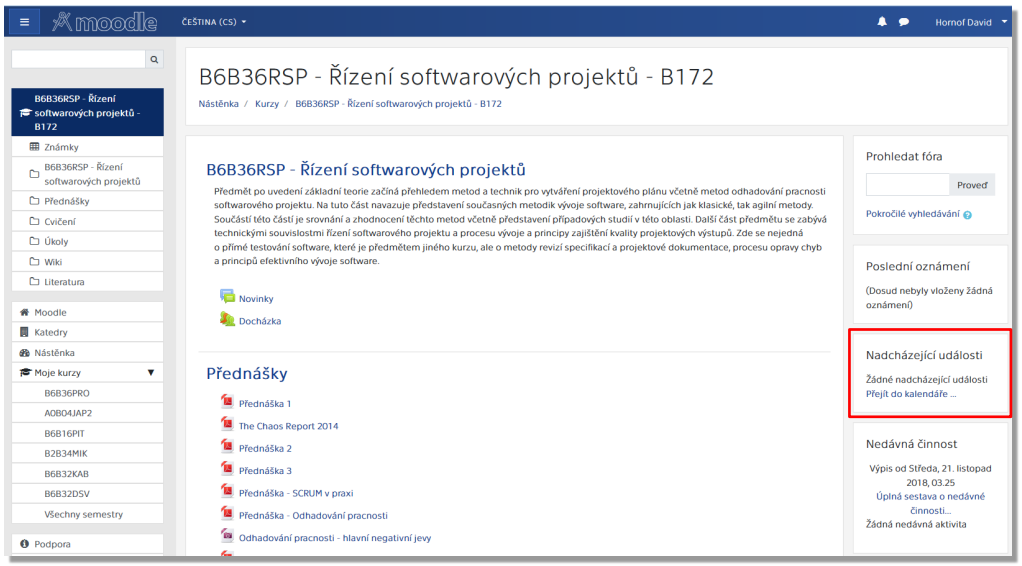
1. **Napojení na aplikaci GitLab:** Aby systém mohl číst data z repozitářů hostovaných v aplikaci GitLab a reagovat na změny v nich, musí být na aplikaci GitLab vhodně napojen.
2. **Korektní spolupráce s existujícími aplikacemi:** Systém svojí činností nenaruší funkčnost aplikací, na které bude napojený, protože by to znemožňovalo jejich používání.
3. **Moodle plugin:** Aby bylo zajištěno správné napojení na aplikaci Moodle, bude systém realizován jako plugin pro tuto aplikaci.
4. **PHP:** Aby mohl být systém realizován jako plugin pro aplikaci Moodle, musí být naprogramovaný v programovacím jazyce PHP.
5. **Bezpečnost:** Protože systém bude pracovat se vstupy od uživatelů (ať už v podobě obsahu z repozitářů nebo v uživatelském rozhraní), bude pro zachování bezpečnosti odolný proti útokům, zejména pak proti útoku cross-site scripting (známý pod zkratkou XSS).
6. **Nezávislost:** Systém bude ke svojí činnosti vyžadovat pouze GitLab FEL a Moodle FEL s výchozími pluginy, protože není důvod systém vázat na jiné pluginy nebo aplikace.

Kapitola 3
Návrh pluginu

V návaznosti na poznatky a požadavky definované v analýze je v této kapitole představen konkrétní návrh na realizaci pluginu.

3.1 Souhrnný popis návrhu

Navrhovaný systém bude realizován jakožto plugin pro aplikaci Moodle, konkrétně plugin typu *block*. Tento typ pluginu se v uživatelském rozhraní aplikace Moodle projevuje, jak je z názvu patrné, jako „blok“, který se nachází na okraji obrazovky. Příklad takového bloku je zobrazen na obrázku 3.1. Toto je vhodné, neboť uživatelské rozhraní navrhovaného pluginu bude poměrně jednoduché a zároveň bude možné pomocí něho udržovat uživatele v obraze o stavu pluginu, respektive stavu importu pro daný kurz. [33]



Obrázek 3.1. Uživatelské rozhraní pluginu typu blok v aplikaci Moodle Fakulty elektrotechnické ČVUT.

Plugin bude dovolovat provést jednorázový import a import s následným automatickým sledováním změn. V rámci importu s automatickým sledováním změn plugin provede nastavení GitLab webhooku pro automatickou reakci na události typu push (tedy k automatické reakci na změny provedené ve zdrojovém repozitáři). Důvodem pro tento typ importu je snaha o optimalizaci využití zdrojů. Pokud uživatel ví, že bude obsah kurzu spravovat pomocí Git repozitáře, použije tento typ importu, což umožní při následných změnách pouze aplikovat tyto změny bez nutnosti opakovaného importu celého kurzu (kurz může dosahovat velikostí desítek MiB, změny se mohou týkat souborů o velikosti desítek KiB). Součástí správné implementace tohoto importu musí být

i zrušení nastaveného webhooku ve chvíli, kdy si uživatel nadále nepřeje import změn ze zdrojového repozitáře a zruší tedy nastavenou automatickou synchronizaci.

Import bude prováděn ve formátu kurzu `topics` (tematické zobrazení), který se skládá pouze z uživatelsky definovaných sekcí. [19] Pro provedení importu bude od uživatele vyžadována informace o názvu zdrojového repozitáře a názvu větve, ze které chce data importovat. Obsah zdrojové větve bude importován v takovém stavu, v jakém se nachází v posledním commitu v této větvi.

Důležitou součástí pluginu také bude služba dostupná z vnějšku, kterou bude využívat webhook aplikace GitLab pro notifikace navrhovaného pluginu o změnách ve zdrojových repozitářích. To bude realizováno pomocí *Web services API* aplikace Moodle. [5]

3.2 Propojení s aplikací GitLab

Plugin bude napojený na aplikaci GitLab pomocí REST API, které tato aplikace nabízí. Bude využívat zejména:

- **Branches API** pro získávání informací o větvích repozitářů. [34]
- **Commits API** pro získávání informací o commitech. [35]
- **Projects API** pro čtení informací o projektech (repozitářích[1]) a nastavování webhooků. [36]
- **Repositories API** pro čtení obsahu (struktury) repozitářů. [26]
- **Repository files API** pro čtení souborů z repozitáře. [27]

Pro autentizaci uživatele bude použitý personal access token popsaný v části 2.5.2. Každý uživatel pluginu bude používat vlastní token.

3.3 Datová vrstva

Plugin bude pracovat se třemi kategoriemi dat. Pro jejich persistenci bude využívat databázi aplikace Moodle.

První kategorií jsou uživatelská nastavení. Ty představují pouze personal access tokeny jednotlivých uživatelů. Pro práci (zápis a čtení) s nimi bude použito *Preference API* aplikace Moodle, jelikož toto API je určené pro práci s tímto typem dat. [37–38]

Druhou kategorií je nastavení importů s automatickým sledováním změn. Bude nutné ukládat následující hodnoty:

- identifikátor kurzu,
- identifikátor uživatele („majitele“ importu) – nutné pro získání jeho personal access tokenu během aktualizačních importů,
- identifikátor repozitáře (`id` projektu) z aplikace GitLab,
- název zdrojové větve v repozitáři,
- id vytvořeného webhooku (kvůli jednoznačné identifikaci pro jeho odstranění při zrušení nastaveného importu),
- hash posledního importovaného commitu.

Pro ukládání těchto dat bude plugin využívat vlastní tabulky, kterou si v databázi vytvoří.

Poslední kategorii dat tvoří záznamy o kurzech, ve kterých v danou chvíli probíhají importy. Tyto záznamy je potřeba uchovávat, aby mohlo být zabráněno případnému

[1] Vysvětlení pojmů „projekt“ a „repozitář“ v kontextu aplikace GitLab lze nalézt v části 2.1.3.

souběhu více importů v jednom kurzu – podrobnější popis tohoto problému je možné nalézt v části 3.7.4. Záznamy budou ve vlastní tabulce a budou tvořeny pouze identifikátory kurzů s aktivními importy.

3.4 Reprezentace kurzů

Dle specifikovaných požadavků bude plugin umožňovat import sekcí, skrytých sekcí, statických stránek, souborů a složek, což znamená nutnost tyto entity reprezentovat ve zdrojovém repozitáři. Statické stránky a popisy sekcí se navíc skládají ze strukturovaného textu, což je potřeba také reflektovat.

Pro reprezentaci strukturovaného textu bude plugin využívat přímo formátů, které aplikace Moodle podporuje: Markdown, HTML a prostý text.

S použitím prostého textu není problém, protože aplikace Moodle sama řeší všechny problémy.

U formátu HTML může nastat celá řada problémů. V první řadě je potřeba odstranit případné nežádoucí části (hlavička, metadata) a navíc zajistit, že výstup nebude obsahovat žádný rizikový obsah, který může mít mnoho podob (například jen možné podoby XSS útoku jsou uvedeny v [39]). Nejjednodušším řešením proto bude na základě vstupu od uživatele vytvořit nový fragment HTML, který se bude skládat jen z povolených tagů bez nežádoucích atributů a až ten ukládat v databázi. Vzhledem k tomu, že tvorba popsaných bezpečných HTML fragmentů je poměrně komplikovanou záležitostí, bude pro tento účel v rámci implementace použita již existující knihovna.

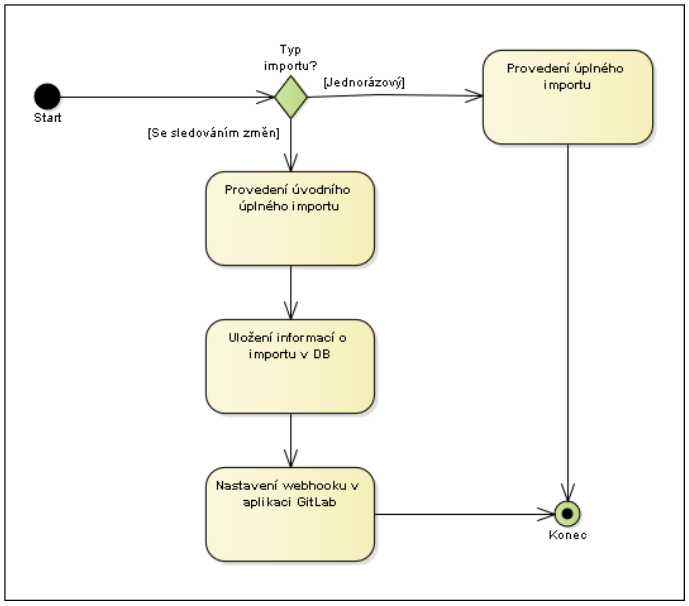
Pro použití formátu Markdown nabízí aplikace Moodle formátovací funkci, která vstup v tomto formátu převede na fragment HTML. V tomto fragmentu však zůstávají i řetězce odpovídající HTML tagům přítomné ve vstupním textu, ať už jde o platné tagy či nikoli, z čehož plyne riziko útoku XSS. Jako vhodná strategie, jak se s tímto problémem vypořádat, se jeví pracovat s výstupem této formátovací funkce jako s uživatelským HTML vstupem, tedy převést ho na „platný“ fragment skládající se pouze z dovolených tagů.

Sekce budou reprezentovány adresáři ve zdrojovém Git repozitáři. Kořenový adresář bude představovat úvodní sekci, podadresáře první úrovně pak jednotlivé sekce. Názvy sekcí budou shodné s názvy zdrojových adresářů. Popis sekce bude obsažen v souboru `index` s příponou v závislosti na použitém formátu (`html`, `md`, nebo `txt`) v příslušném adresáři. Zde nastává problém možné kolize ve chvíli, kdy se v jednom adresáři nachází více souborů s názvem `index` a podporovanou příponou, například `index.html` a `index.md`. Tento problém je blíže rozebrán v části 3.7.1. Další soubory v daném adresáři s příponami `html`, `md` a `txt` budou importovány jako statické stránky, jejichž název bude shodný s názvem zdrojového souboru, vždy jako moduly sekce v závislosti na tom, v jakém adresáři se nachází. Zde opět nastává problém možné kolize rozebraný v části 3.7.1. Všechny ostatní soubory budou importovány jako studijní materiály typu soubor opět v příslušné sekci v závislosti na adresáři. Nakonec podadresáře druhé úrovně budou importovány jako složky v příslušné sekci. Podadresáře vyšší úrovně budou ignorovány, jelikož jejich užitná hodnota je v kontextu navrhovaného systému nízká.

Zvláštní význam bude mít soubor `hidden.txt`. Pokud tento soubor bude existovat v libovolném podadresáři první úrovně, tak nezávisle na jeho obsahu bude příslušná sekce označena jako skrytá před studenty kurzu.

3.5 Typy importů z hlediska chování

Z hlediska svého chování je možné importy rozdělit na *jednorázový import* a *import s automatickým sledováním změn ve zdrojovém repozitáři*. Provedení jednorázového importu se skládá pouze z provedení samotného importu obsahu. V případě importu s automatickým sledováním změn dojde po provedení importu obsahu k uložení informací o tomto importu (popsaných v části 3.3) a nastavení webhooku v aplikaci GitLab pro automatické sledování změn. Diagram procesů těchto importů je zobrazen na obrázku 3.2.



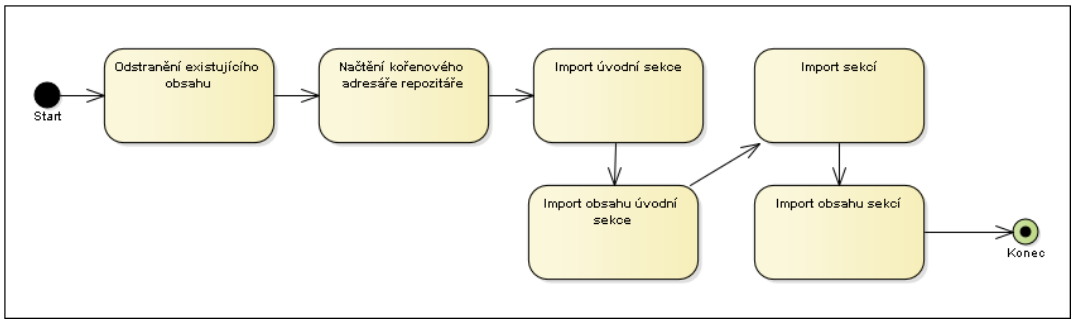
Obrázek 3.2. Diagram znázorňující procesy importů z hlediska jejich chování.

3.6 Typy importů z hlediska rozsahu

Z hlediska rozsahu lze importy rozdělit na úplné a aktualizační. V obou případech půjde časově náročné operace (řádově desítky sekund až minuty v závislosti na velikosti zdrojového repozitáře), bude proto nutné importy spouštět asynchronně. K tomuto účelu nabízí aplikace Moodle *Task API*. [40] V důsledku toho nastává i možný problém v podobě souběhu více probíhajících importů v rámci jednoho kurzu. *Task API* nabízí možnost detekce duplicitních úloh a následnému zamezení jejich spuštění, tato detekce ale není pro řešení popsaného problému dostatečná, protože stále může dojít například ke spuštění dvou importů z různých repozitářů v rámci jednoho kurzu najednou (pro *Task API* se úlohy nebudou jevit jako duplicitní, protože budou mít rozdílná vstupní data – identifikátory repozitářů). Proto bude navrhovaný systém zvlášť evidovat kurzy, ve kterých probíhá import, jak bylo popsáno v části 3.3, a nedovolí spuštění nového importu pokud v kurzu v danou chvíli už nějaký import probíhá.

3.6.1 Úplný import

Úplný import se skládá z odstranění veškerého obsahu kurzu a následném kompletním převodu obsahu zdrojové větve (v takovém stavu, v jakém se nachází po posledním commitu ve zdrojové větvi v době spuštění tohoto importu). Úplný import se provádí v případě jednorázového importu a jako úvodní import při importu s automatickým sledováním změn. Proces úplného importu je znázorněn na obrázku 3.3.



Obrázek 3.3. Diagram znázorňující proces úplného importu.

1) **Odstranění existujícího obsahu:** Plugin odstraní všechny záznamy z tabulky `course_modules` pro daný kurz spolu s navázanými záznamy v ostatních tabulkách, stejně tak odstraní pro daný kurz všechny záznamy z tabulky `course_sections`.
2) **Načtení kořenového adresáře repozitáře:** Plugin získá obsah zdrojové větve (názvy souborů a podadresářů první úrovně) a seřadí ho abecedně. V tomto pořadí bude načtený obsah zpracovávat.
3) **Import úvodní sekce:** Plugin vytvoří záznam v tabulce `course_sections` s hodnotou sloupce `section` nastavenou na `0`. Pokud se v kořenovém adresáři bude nacházet soubor `index` s příponou představující strukturovaný textový obsah[1], bude obsah tohoto souboru převeden jako textový obsah sekce (sloupec `summary`), v opačném případě bude hodnota sloupce ponechána jako `NULL`. Dále se budou provádět následující akce (**import obsahu úvodní sekce**):
   a) Každý získaný soubor v adresáři s příponou představující soubor se strukturovaným textovým obsahem bude importován jako stránka. To znamená vytvoření záznamu v tabulce `page`, dále navázaného záznamu v tabulce `course_modules` a nakonec aktualizaci sloupce `sequence` v tabulce `course_sections` pro záznam představující danou sekci.
   b) Každý zbylý soubor bude importován jako studijní materiál typu soubor. Plugin vytvoří záznam v tabulce `resource`, navázaný záznam v tabulce `course_modules`, aktualizuje sloupec `sequence` v tabulce `course_sections` a s využitím *File API* aplikace Moodle soubor nahraje do jejího souborového úložiště.
4) **Import sekcí:** Pro každý adresář v kořenovém adresáři bude plugin provádět následující akce (**import obsahu sekcí**):
   a) Načte jeho obsah – soubory a podadresáře druhé úrovně a opět je seřadí a následně zpracovává abecedně.
   b) Vytvoří záznam v tabulce `course_sections` představující sekci. Pokud se v tomto adresáři nacházel soubor `index` s příponou představující strukturovaný textový obsah, bude obsah tohoto souboru převeden jako textový obsah sekce (sloupec `summary`), v opačném případě bude hodnota sloupce ponechána jako `NULL`. Pokud se zde nacházel i soubor `hidden.txt`, je sekce označena jako skrytá (hodnota sloupce `visible` bude `0`).
   c) Každý získaný soubor v adresáři s příponou představující soubor se strukturovaným textovým obsahem bude importován jako stránka. To znamená vytvoření záznamu v tabulce `page`, dále navázaného záznamu v tabulce `course_modules` a nakonec aktualizaci sloupce `sequence` v tabulce `course_sections` pro záznam představující danou sekci.

[1] Tím se rozumí soubory s příponami `html`, `md` a `txt`, jak bylo popsáno v části 3.4.

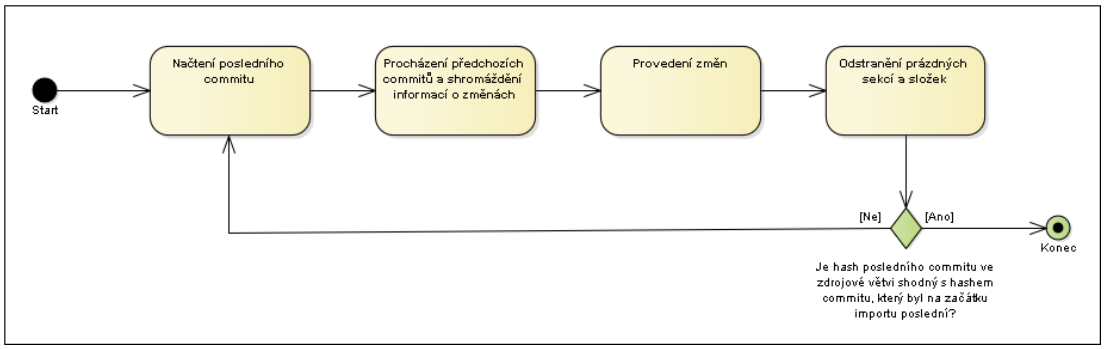
d) Každý získaný podadresář bude importován jako složka. Proces bude obdobný jako v předchozím kroku, skládá se z vytvoření záznamu v tabulce `folder`, navázaného záznamu v tabulce `course_modules` a konečně aktualizace sloupce `sequence` v tabulce `course_sections`. Bude zde navíc ještě krok náhrání souborů složky do souborového úložiště aplikace Moodle, k čemuž musí být využito *File API* této aplikace. Provázání nahraných souborů se složkou se děje pomocí kontextu.

e) Každý zbylý soubor bude importován jako studijní materiál typu soubor. Plugin vytvoří záznam v tabulce `resource`, navázaný záznam v tabulce `course_modules`, aktualizuje sloupec `sequence` v tabulce `course_sections` a s využitím *File API* aplikace Moodle soubor nahraje do jejího souborového úložiště.

3.6.2 Aktualizační import

Pokud byl kurz již importován a jsou v něm sledované změny, každý další import, spuštěný uživatelem či GitLab webhookem, bude aktualizační. Rozdíl oproti úplnému importu spočívá v tom, že tento import bude za použití GitLab API pouze aplikovat změny provedené v repozitáři do cílového kurzu.

Změny ve zdrojovém repozitáři mohou být dvojího typu: odstraněný soubor a upravený soubor. Nový soubor je pro účely zpracování pluginem totéž co upravený soubor. Přejmenovaný soubor odpovídá odstranění souboru se starým názvem a vytvoření souboru s novým názvem. Proces aktualizačního importu je znázorněn na obrázku 3.4.



Obrázek 3.4. Diagram znázorňující proces aktualizačního importu.

1) **Načtení posledního commitu:** Načtení informací o posledním commitu ve zdrojové větvi.
2) **Procházení předchozích commitů a shromáždění informací o změnách:** Shromáždění informací o změnách (změněné a odstraněné soubory – tyto informace poskytuje GitLab Commits API [35]) pro každý commit počínaje od posledního commitu ve větvi. Následuje načtení informací o commitu předchozím, ze kterého jsou opět shromážděny informace o změnách. V potaz je brán pouze výsledný stav repozitáře, tedy pokud byla v chronologicky dřívějším commitu detekována změna souboru, který byl v pozdějším commitu odstraněn, je tato změna ignorována. Tento proces se opakuje, dokud se nedojde ke commitu, jehož hash je uložen v databázi jako poslední importovaný.
3) **Provedení změn:** Aktualizace dat v databázi a v souborovém úložišti aplikace Moodle na základě shromážděných informací. Vytváření souborů a adresářů je shodné jako v případě úplného importu. V případě úpravy souboru stačí dotčený soubor přepsat v souborovém úložišti bez nutnosti upravovat jiné databázové záznamy. Odstranění souborů (či adresářů) znamená nutnost odstranění záznamů v příslušných

tabulkách a dotčených souborů ze souborového úložiště. Při změnách u souborů se strukturovaným textovým obsahem stačí aktualizovat správná pole v příslušných databázových tabulkách. V případě změny obsahu studijních materiálů typu složka stačí nahrát (případně odstranit) dotčené soubory a správně je se složkou provázat pomocí kontextu. V případě, že byl vytvořen nebo odstraněn soubor `hidden.txt` v podadresáři první úrovně, stačí patřičně aktualizovat hodnotu sloupce `visible` v tabulce `course_sections` u záznamu příslušné sekce. Důležitou součástí celého tohoto kroku je aktualizace sloupce `sequence` v tabulce `course_sections` u sekcí, kde došlo ke změně modulů, aby bylo zachováno jejich abecední pořadí.

4) **Odstranění prázdných sekcí a složek:** Během aktualizačního importu mohly v kurzu vzniknout prázdné složky a prázdné sekce (jelikož změny jsou sledovány na úrovni souborů a ne adresářů). Aby tyto pozůstatky „nepřekážely“, budou odstraněny.
5) **Kontrola posledního commitu ve zdrojové větvi:** Ověření, zdali se hash posledního commitu ve zdrojové větvi shoduje s hashem commitu, který byl na začátku importu načten jako poslední. Pokud ano, import se ukončí a pokud ne, znamená to, že se během importu obsah zdrojového repozitáře změnil, proces je tedy spuštěn znovu od začátku, aby aplikoval i nově provedené změny.

3.7 Možné problémy při provozu pluginu

Tato část se věnuje popisu problémů, na které může plugin narazit během svého provozu a se kterými by se měl být schopen vyrovnat. V případě, že plugin narazí na problém, o kterém je potřeba informovat uživatele (například selhání při importu), bude uživateli zaslána notifikace v aplikaci Moodle pomocí poskytovaného *Message API*. [41]

3.7.1 Kolize textových souborů

Vzhledem k možnostem reprezentace stejného obsahu (statické stránky, textové části sekce) ve více formátech existuje riziko kolize těchto reprezentací, tedy situace, kdy budou najednou ve stejném adresáři existovat dva soubory (případně i více souborů) se stejným jménem ale jinými z přípustných přípon (`html`, `md`, `txt`). Plugin bude zpracovávat položky v repozitáři v abecedním pořadí (vzestupně), proto bude výsledkem importu při kolizi obsah posledního zpracovaného souboru. V případě následných změn bude výsledkem importu poslední změněný soubor, v případě více změněných souborů najednou to opět bude obsah posledního zpracovaného souboru ze změněných souborů.

3.7.2 Chybný token

Personal access token používaný pro autentizaci není ve správě navrhovaného pluginu a může dojít ke ztrátě jeho platnosti. Uživatel může token v aplikaci GitLab odstranit, tokenu může skončit platnost, pokud ji uživatel omezil, nebo může na tokenu být chybně nastavený rozsah (*scope*). V případě, že se plugin setká s chybou autentizace způsobenou chybným tokenem, informuje o této chybě uživatele.

3.7.3 Nedostupné GitLab API

GitLab je samostatnou aplikací a může dojít k jejímu výpadku, což způsobí mimo jiné nemožnost používání jejího API, důsledkem čehož je nemožnost provést import. V případě této chyby plugin informuje uživatele o nemožnosti provést import z důvodu nedostupnosti aplikace GitLab.

3.7.4 Souběh více importů

Protože proces importu probíhá asynchronně, může nastat situace, ve které si uživatel pluginu (nebo webhook aplikace GitLab) vyžádá spuštění nového importu, zatímco v danou chvíli už import v kurzu probíhá. Řešení tohoto problému je popsáno v části 3.6.

3.7.5 Velikost importovaných souborů

Vzhledem k tomu, že se ve zdrojovém repozitáři mohou nacházet i velké soubory, které není z provozních důvodů žádoucí importovat (zátěž sítě a úložiště), bude navrhovaný plugin před zpracováním jakéhokoli souboru kontrolovat jeho velikost (GitLab API poskytuje informace o velikosti souboru [27]). Pokud by došlo k překročení limitu maximální velikosti souboru, který bude v pluginu definovaný, soubor nebude importován a uživatel bude o této chybě informován. Zbytek importu nebude ovlivněn.

3.7.6 Kódování u souborů s textovým obsahem

U souborů, které představují textový obsah, je potřeba brát v potaz také jejich znakové kódování. Jelikož systém Git dovoluje v repozitáři nastavit požadované kódování souborů a soubory s jiným kódováním automaticky převádět [42], nebude navrhovaný plugin tento problém řešit přímo – zajištěna v něm bude podpora pro kódování UTF-8, které je dnes nejobvykleji používaným kódováním [43], a bude na uživateli, aby toto kódování používal, nebo správně nastavil zdrojový Git repozitář.

3.8 Omezení pluginu

Z povahy navrhovaného pluginu vyplývá i řada omezení. Tři významné z nich byly identifikovány a v této části popsány.

3.8.1 Řazení souborů

Na FEL ČVUT bývá zvykem materiály z přednášek a cvičení poskytovat pro větší přehlednost seřazené podle data jejich konání. Jelikož navrhovaný plugin nebude mít k dispozici informace, na základě kterých by toto řazení mohl provést, a obecně nebude ani schopen rozeznávat, co jednotlivé studijní materiály v podobě souborů představují (jestli se jedná o materiály z přednášek, cvičení, nebo úplně jiné soubory), nebude umožňovat toto řazení. Plugin umožní tento problém obejít vhodným pojmenováním zdrojových souborů a složek, jelikož pořadí importovaných modulů (a sekcí) bude pevně definováno – moduly (a sekce) budou po importu seřazeny abecedně podle svého názvu.

3.8.2 Pokročilé funkce systému Git

Systém Git nabízí i množství pokročilejších funkcí, týkající se především práce s repozitářem (případně distribuovanými repozitáři), existuje však jedna významná pokročilejší funkce týkající se práce se soubory: submoduly. Submoduly jsou „repozitáře v repozitáři“ – adresáře obsahující jiný Git repozitář, důležité však je, že soubory v submodulu nejsou součástí hlavního repozitáře. Submodul je jen prázdný adresář spolu s informací o tom, kde se nachází napojený repozitář. Program `git` je na základě této informace schopný napojený repozitář získat a uživatel má při práci k dispozici jak soubory hlavního repozitáře, tak soubory submodulu. Navrhovaný plugin však nebude umožňovat čtení obsahu submodulů, protože, jak bylo zmíněno, v hlavním repozitáři se submodul projevuje pouze jako prázdný adresář a získání obsahu napojeného repozitáře je velmi komplikovaná záležitost už jen proto, že se tento repozitář vůbec nemusí v GitLabu

FEL nacházet. Krom toho se nepředpokládá široké využívání této funkce budoucími uživateli pluginu. [8]

3.8.3 Uživatelské změny v importovaném kurzu

V importovaném kurzu bude uživatel mít možnost provádět změny, tedy vytvářet dodatečné moduly a sekce nebo importované moduly a sekce přejmenovávat – tuto schopnost není možné vhodně omezit. Problémem bude neschopnost navrhovaného pluginu správného rozpoznání a následné práce s dotčenými moduly a sekcemi (zejména během aktualizačního importu), způsobená absencí reprezentace těchto entit ve zdrojovém repozitáři. Úplný import přepíše obsah kurzu vždy. Aktualizační import bude moduly a sekce bez reprezentace ve zdrojovém repozitáři ignorovat, prázdné sekce a moduly typu složka však v rámci aktualizačního importu odstraní.

Kapitola 4
Implementace

Implementované řešení vychází z požadavků specifikovaných v části 2.6 a odpovídá návrhu z kapitoly 3. V souladu s požadavky byla implementace provedena v programovacím jazyce PHP, aby vzniklé řešení mohlo fungovat jako plugin pro aplikaci Moodle. Plugin byl pojmenován *gitlabi*, jakožto zkratka z *GitLab Integration*. Vzhledem k prostředí Fakulty, kde se předpokládá znalost anglického jazyka a tomu, že výchozím jazykem aplikace Moodle je též angličtina [44], bylo uživatelské rozhraní implementováno právě v anglickém jazyce.

4.1 Adresářová struktura pluginu

Adresářová struktura pluginu vychází z požadavků kladených aplikací Moodle. Adresáře `classes`, `db` a `lang` slouží k napojení na tuto aplikaci – této problematice se detailně věnuje část 4.2. Adresář `htmlpurifier` obsahuje knihovnu *HTML Purifier*, která slouží k tvorbě bezpečných fragmentů HTML, o čemž je blíže pojednáno v části 4.4. V kořenovém adresáři jsou dodatečné soubory pro napojení na aplikaci Moodle a některé pomocné soubory pluginu (zejména soubory obsahující definice vstupních formulářů pro konfiguraci importu).

4.2 Napojení na aplikaci Moodle

O napojení na aplikaci Moodle lze v kontextu tohoto pluginu hovořit na dvou úrovních – jednak o standardním napojení jakožto pluginu a jednak o specifickém napojení na úrovni datové vrstvy pro provádění importu kurzu.

4.2.1 Napojení na úrovni pluginu

Napojení pluginu do aplikace Moodle je realizováno pomocí specifických „objektů“ (třídy a proměnné). Aplikace Moodle specifikuje očekávané umístění těchto „objektů“, které jsou celkem 4:

- kořenový adresář pluginu,
- adresář `classes`,
- adresář `db`,
- adresář `lang`.

V kořenovém adresáři napojení zajišťují soubory `version.php`, `settings.php`, `externallib.php` a `block_gitlabi.php`.

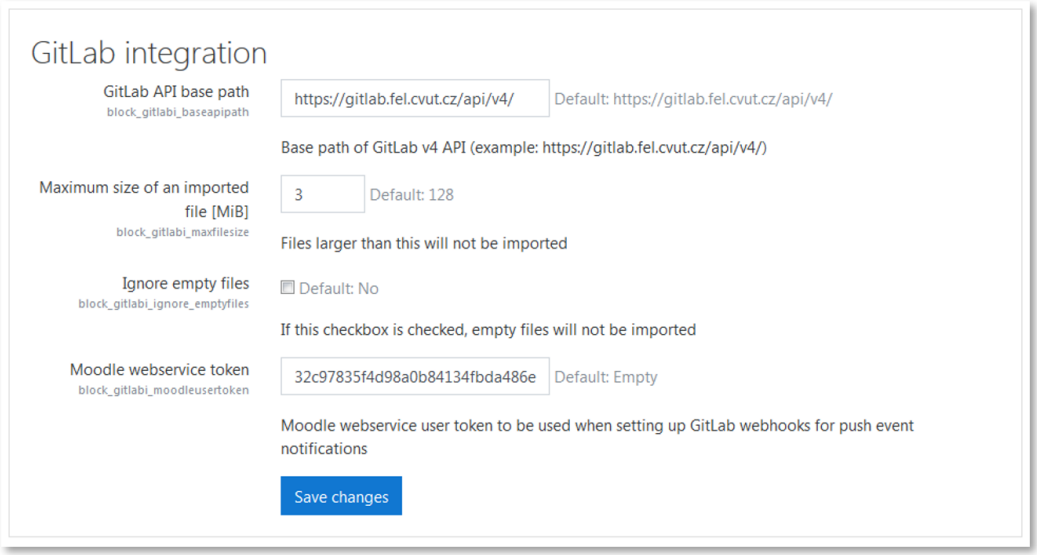
V souboru `version.php` je definován název pluginu a verze pluginu spolu se závislostmi na ostatní pluginy. [45] Tento plugin závisí na verzi jádra aplikace Moodle `2018120302` (označení pro verzi 3.6.2, pro kterou byl plugin vyvíjen). Dále definuje závislost na modulech `mod_resource`, `mod_page` a `mod_folder`, protože jde o moduly, jejichž instance plugin vytváří při importu (moduly soubor, stránka a složka). Jde sice

o moduly přítomné ve výchozí instalaci aplikace Moodle, přesto je kontrola jejich přítomnosti nutná, protože je možné tyto moduly z aplikace Moodle odstranit. V takové situaci by plugin nemohl fungovat.

Soubor `settings.php` obsahuje definice vstupních polí pro globální konfiguraci pluginu v administračním rozhraní [45], v tomto případě jde o 4 vstupy:

- Základní URL API aplikace GitLab,
- maximální velikost importovaného souboru,
- volbu, zda při importu ignorovat prázdné soubory,
- token Moodle uživatele pro použití webservice.

Podobu globálního nastavení v administračním rozhraní je možné vidět na obrázku 4.1. Bližší informace o významu těchto polí lze nalézt v části 5.2.

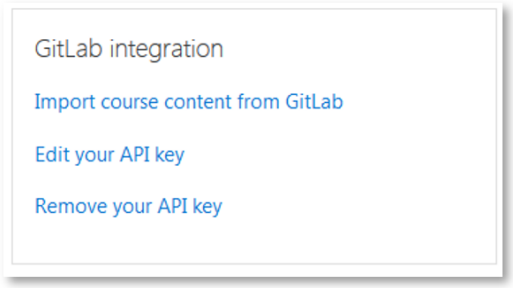


Obrázek 4.1. Uživatelské rozhraní pro konfiguraci pluginu GitLab integration.

V souboru `externallib.php` se nachází třída `block_gitlabi_external` obsahující metody takzvaného externího rozhraní pluginu, tedy funkcí, které plugin poskytuje navenek. U tohoto pluginu jde o jedinou metodu, a to `gitlab_repo_push`. Tato metoda je volána prostřednictvím REST API aplikace Moodle webhookem z aplikace GitLab při událostech typu push. Třída dále obsahuje pomocné metody, jejichž návratové hodnoty slouží pro popis vstupních a návratových hodnot poskytovaných externích funkcí (v tomto případě jediné funkce).

Soubor `block_gitlabi.php` obsahuje třídu `block_gitlabi`, která je pro celý plugin klíčová – aplikace Moodle ji totiž využívá pro vykreslování obsahu uživatelského rozhraní, tedy zobrazovaného bloku (který je možné vidět na obrázku 4.2). Tato třída zároveň obsahuje i pomocné metody, jejichž návratová hodnota určuje chování bloku – například metoda `has_config()` vždy vrací hodnotu `true`, což indikuje, že plugin má vlastní nastavení (popsané v souboru `settings.php`), metoda `instance_allow_multiple()`, vracející vždy hodnotu `false` indikuje, že není možné mít víc než jednu instanci tohoto bloku pro kurz (vzhledem k funkcionalitě pluginu to nedává smysl). [33]

V adresáři `lang` se nacházejí podadresáře s dvoumístným kódem jazyka (například `en`). V každém tomto podadresáři se nachází soubor `block_gitlabi.php` obsahující textové řetězce, které využívá *String API* aplikace Moodle pro účely lokalizace textového obsahu. [44]



Obrázek 4.2. Hlavní uživatelské rozhraní pluginu GitLab integration.

Adresář `db` obsahuje soubory, které popisují volitelná napojení na různé další části aplikace Moodle, včetně definice vlastních tabulek v databázi. Tato napojení spočívají v naprosté většině případu v definici proměnné (asociativního pole) s předem daným názvem. U tohoto pluginu lze v tomto adresáři nalézt soubory `access.php`, `events.php`, `messages.php`, `services.php` a `install.xml`.

Soubor `access.php` obsahuje definici takzvaných *capabilities*, což jsou „jednotky“ oprávnění – jde o způsob, jakým lze aplikaci Moodle řídit uživatelská práva. Uživatelům lze pomocí rolí přiřazovat různé oprávnění pro jednotlivé capabilities (tyto oprávnění jsou *allow*, *deny* a *prohibit*) v různých kontextech, výsledkem čehož je sada oprávnění pro každého uživatele aplikace. U tohoto pluginu je jediným definovaným oprávněním (vyžadováno aplikací Moodle) `addinstance`. Jak je z názvu patrné, jde o oprávnění přidávat do kurzu instance bloku tohoto pluginu. [23]

V souboru `events.php` lze nalézt definice observerů („pozorovatelů“) pro události (*events*). Příkladem události v aplikaci Moodle může být smazání kurzu, což je zrovna událost, pro kterou je v tomto pluginu definován observer. Observer události smazání kurzu v tomto pluginu zajistí, že pokud v kurzu byl nastaven import s automatickým sledováním změn, dojde v rámci smazání kurzu k odstranění tohoto nastavení, tedy k vymazání příslušného záznamu z databázové tabulky pluginu a pokusu o zrušení nastaveného webhooku v aplikaci GitLab. [45]

Soubor `messages.php` obsahuje definici typů zpráv (v terminologii Moodle *message providers*), které plugin posílá. Účel těchto definic je dovolit uživateli, aby si mohl zvolit, zda chce konkrétní typy zpráv dostávat a jak na ně chce být upozorňován (zdali chce být informován v aplikaci Moodle, e-mailem nebo obojím). Tento plugin definuje jediný typ zpráv – upozornění související s importy. [41, 45]

V souboru `services.php` se nachází definice externích služeb doplňující API aplikace Moodle. [45] V případě tohoto pluginu jde o metodu `gitlab_repo_push`, což je metoda, kterou pomocí REST API aplikace Moodle používá webhook aplikace GitLab při událostech typu push ve sledovaných repozitářích.

Soubor `install.xml` obsahuje definici databázového schématu pluginu ve formátu XML, respektive ve formátu XMLDB. [45–46] Tento plugin definuje dvě tabulky v souladu s návrhem z části 3.3: `block_gitlabi`, jakožto hlavní tabulku pluginu, která obsahuje nastavení importů s automatickým sledováním změn a druhou tabulku, `block_gitlabi_active_imports`, která obsahuje identifikátory kurzů, ve kterých probíhá import. V obou tabulkách je definován index nad sloupcem `courseid` (sloupec obsahující identifikátor kurzu), jelikož prakticky veškeré dotazování nad těmito tabulkami bude probíhat s vyhledáváním v tomto sloupci. Tento index je navíc označen jako unikátní pro zajištění dodatečné integritní kontroly, protože nelze mít více nastavených

importů s automatickým sledováním změn v jednom kurzu, stejně tak jako není možné mít více aktivních importů v jednom kurzu.

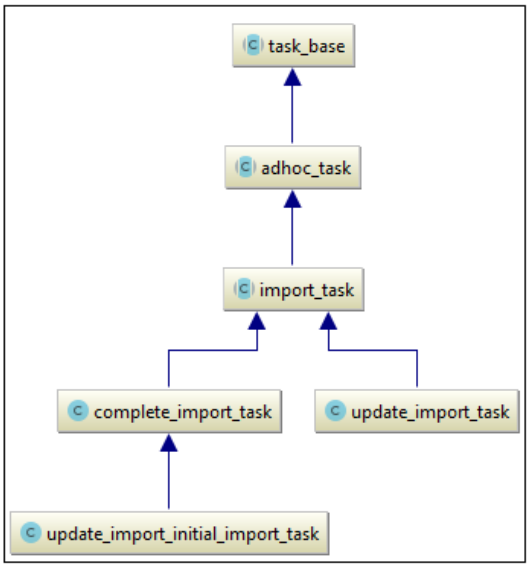
Adresář `classes` obsahuje různé třídy, které plugin využívá. Výhodou tohoto adresáře je, že aplikace Moodle nad ní provádí takzvaný *autoloading* (ačkoli musí být dodrženy některé konvence pro pojmenovávání souborů a tříd), což znamená, že není nutné soubory se zdrojovým kódem z tohoto adresáře a jeho podadresářů ručně importovat na jiných místech. [47] Pro tento plugin obsahuje tento adresář dva podadresáře: `gitlab`, obsahující implementaci wrapperů nad REST API aplikace GitLab, které jsou blíže popsané v části 4.3, a `task`, obsahující implementaci asynchronních adhoc úloh provádějící samotný import, která je blíže popsaná v části 4.2.2. Dále lze v tomto adresáři nalézt soubor `moodle_course_adapter.php` obsahující třídu zajišťující napojení na datovou vrstvu aplikace Moodle pro účely provedení importu (o tom je pojednáno v části 4.2.3), `event_observers.php` se statickou třídou s implementací dříve zmíněného observeru událostí smazání kurzu a `gitlabi_helpers.php` obsahující třídu se statickými pomocnými metodami používanými napříč celým pluginem.

4.2.2 Asynchronní úlohy pro provedení importu

Samotný proces importu je prováděn v rámci asynchronních adhoc úloh, pro jejichž vykonávání aplikace Moodle poskytuje *Task API*. [40] V rámci pluginu gitlabi jsou tyto úlohy implementovány ve 4 třídách:

- `import_task`,
- `complete_import_task`,
- `update_import_initial_import_task`,
- `update_import_task`.

Vztah těchto tříd je zachycen na obrázku 4.3. Třídy `task_base` a `adhoc_task` patří jádru aplikace Moodle.



Obrázek 4.3. Vztah tříd implementující úlohy pro import obsahu kurzu.

Třída `import_task` je abstraktním předkem pro ostatní třídy implementující úlohy importu. Sama implementuje metodu `execute()`, která je volána, když má být úloha vykonána. V rámci této metody zabalí celé provedení importu do databázové transakce. S ohledem na to, jak je implementováno souborové úložiště aplikace Moodle to

pak znamená, že pokud dojde během importu k chybě, může být proveden rollback započaté transakce, po kterém se kurz dostane do stejného stavu, v jakém byl před započetím importu (tedy i bez odstranění jakéhokoli obsahu). Další součástí této metody je i odeslání notifikace uživateli, jenž započal import s informací o výsledku importu (úspěch či neúspěch). Samotné provedení importu je delegováno na abstraktní metodu `perform_import()`, ve které třídy pro konkrétní import implementují vlastní logiku importu.

Úplný import je implementovaný v třídě `complete_import_task`, aktualizační import ve třídě `update_import_task`. Jejich logika odpovídá popisu z části 3.6. Třída `update_import_initial_import_task` implementuje logiku úplného úvodního importu pro import s automatickým sledováním změn, ve které se před započetím importu ujistí, že na kurz nejsou navázány žádné existující importy s automatickým sledováním změn (případně je odstraní, ačkoli by tato situace neměla nastat) a po skončení importu uloží potřebné informace v databázi a vytvoří webhook v aplikaci GitLab.

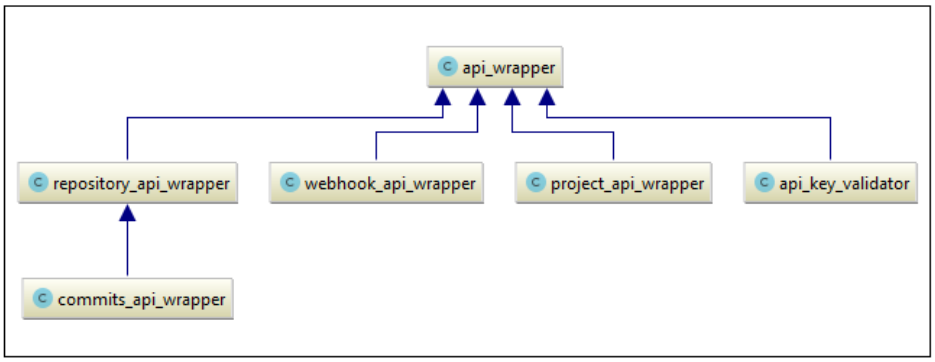
4.2.3 Napojení na úrovni datové vrstvy

Tato úroveň napojení je specifická pro tento plugin, přímo totiž vyplývá z jeho funkcionality. Jedná se o práci s datovou vrstvou (zejména s databází) na úrovni kurzů a modulů, zejména jde o „umělou“ tvorbu instancí jiných modulů. Celé toto napojení je realizováno třídou `moodle_course_adapter`, která obsahuje metody pro přidání, odstranění a případně i úpravy entit aplikace Moodle, se kterými se pracuje při importu, tedy sekcí a modulů soubor, stránka a složka. Rovněž obsahuje pomocné metody, které zajišťují například úplný reset kurzu (vymazání veškerého obsahu prováděné na začátku úplných importů) nebo odstranění prázdných sekcí.

4.3 Napojení na aplikaci GitLab

Implementovaný plugin komunikuje s aplikací GitLab pomocí jejího REST API. Pro tuto komunikaci byl implementován vlastní *wrapper* (respektive sada *wrapperů*), tedy obálka, která od této komunikace odstiňuje zbytek pluginu v souladu s principem *Loose Coupling*. [48]

Zdrojový kód těchto wrapperů je možné nalézt v podadresáři `gitlab` v adresáři `classes`. Jedná se o 6 tříd, jejichž vztah je zobrazen na obrázku 4.4.



Obrázek 4.4. Vztah tříd zprostředkovávajících komunikaci s aplikací GitLab pomocí jejího REST API.

Třída `api_wrapper` je předkem všech zbylých tříd, obsahuje metody pro obecné volání REST API endpointu aplikace GitLab.

Třída `api_key_validator` obsahuje jedinou metodu, jejíž účel je ověřit, zda je zadaný personal access token platný.

Zbylé třídy se starají o komunikaci s příslušnými API popsaných v části 3.2, tedy `webhook_api_wrapper` poskytuje metody pro vytvoření a odstranění webhooku, `project_api_wrapper` poskytuje metody pro vyhledávání projektů podle jména a dále pak čtení informací o projektu v aplikaci GitLab, jako jsou jeho jméno a větve v obsaženém Git repozitáři.

Třída `repository_api_wrapper` slouží ke čtení informací o souborech a adresářích v repozitáři a i přímo ke čtení souborů z repozitáře. Třída `commits_api_wrapper` tuto třídu rozšiřuje o metody pro zpracování informací o commitech pro účely aktualizačního importu. Této problematice je věnována část 4.3.1.

4.3.1 Zpracování commitů

Aktualizační import spočívá v promítnutí změn, které byly provedeny ve zdrojovém repozitáři, do dotčeného kurzu. Aby mohly být změny aplikovány, musí být nejdříve shromážděny informace o tom, k jakým změnám došlo. Tento proces spočívá v procházení commitů chronologicky od posledního commitu ve větvi až k poslednímu importovanému commitu. V rámci každého individuálního commitu pak dochází k zaznamenání informací o změněných a odstraněných souborech, což jsou dva základní typy změn, na které je možné všechny ostatní redukovat.

Jelikož každý commit může mít libovolný počet předků (commity tvoří orientovaný acyklický graf), je procházení realizováno iterativně s použitím zásobníku – na zásobník se přidávají nezpracovaní předci aktuálně zpracovávaného commitu. Nejdříve je vytvořen seznam všech commitů předcházejících poslednímu importovanému commitu, tyto commity jsou nazvány jako zpracované. Následuje podobný proces (vytvoření seznamu předcházejících commitů) pro poslední commit ve větvi, s tím rozdílem, že je tvořen seznam commitů *ke zpracování* a jsou do něj zařazeny jen commity, které se nenachází na seznamu zpracovaných commitů. Ze seznamu commitů ke zpracování je tvořen seznam změn, které je potřeba zanést do kurzu. Commity jsou řazeny chronologicky podle jejich atributu `authored date`.

4.3.2 Komunikace s REST API aplikace Moodle

Webhooky vytvořené v aplikaci GitLab předávají pluginu informaci o události typu push voláním metody `block_gitlabi_gitlab_repo_push`, kterou plugin vystavuje pomocí REST API aplikace Moodle. Při komunikaci je předáván jediný parametr – identifikátor záznamu s informacemi o importu v hlavní databázové tabulce pluginu.

4.4 Tvorba bezpečných HTML fragmentů

V pluginu je pro tvorbu bezpečných a validních HTML fragmentů z uživatelského vstupu použita knihovna HTML Purifier. [49] Důvodem pro zvolení této knihovny bylo její snadné použití spolu s možností velmi silné kontroly nad výstupem – zejména možnosti přesně definovat povolené tagy, povolené atributy, vstupní a výstupní kódování. Pro zpracování vstupu však nakonec bylo použito její výchozí nastavení, které se ukázalo jako odpovídající potřebám tohoto pluginu, což však nevylučuje možnost toto nastavení v budoucnu podle potřeb změnit.

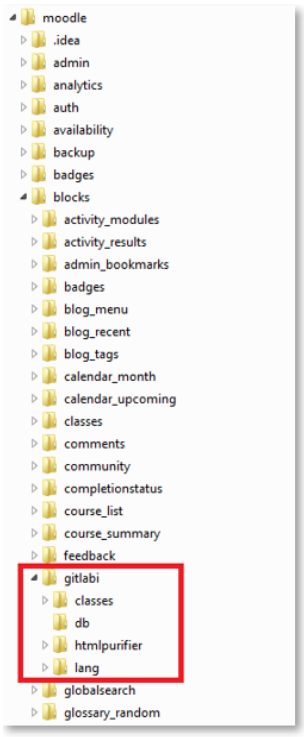
Kapitola 5
Instalace a konfigurace pluginu

Aby bylo možné implementovaný plugin v aplikaci Moodle používat, je nutné ho nainstalovat a nakonfigurovat.

5.1 Instalace

Prvním krokem instalace je nahrání souborů pluginu do aplikace Moodle. To lze provést dvěma způsoby – ručním umístěním těchto souborů do správné lokace, nebo použitím rozhraní, které aplikace Moodle nabízí.

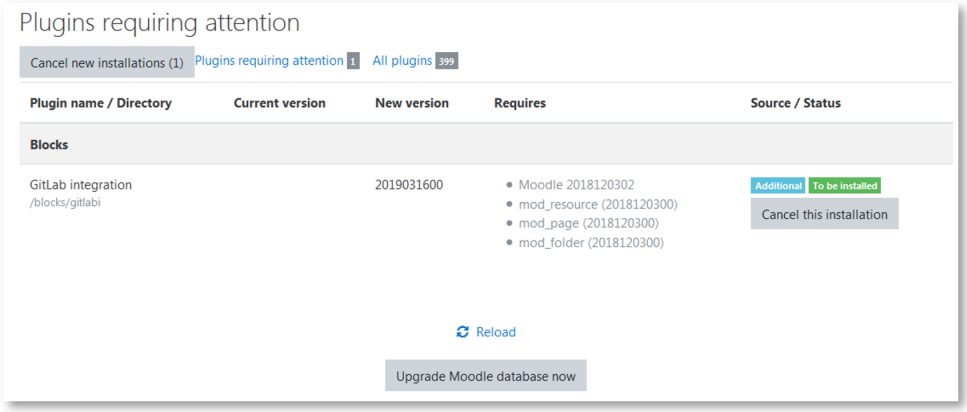
Ruční metoda spočívá v umístění souborů pluginu do podadresáře aplikace Moodle `<moodle root>/blocks/gitlabi`. Správná adresářová struktura podadresáře `blocks` s nahranými soubory pluginu je zobrazena na obrázku 5.1.



Obrázek 5.1. Správné umístění pluginu GitLab integration v adresářové struktuře aplikace Moodle.

Alternativním způsobem nahrání souborů pluginu je využití rozhraní, které aplikace Moodle nabízí. Pro tento způsob je nejprve nutné vytvořit adresář `gitlabi`, do něj umístit veškeré soubory pluginu a následně tento adresář umístit do zip archivu, který zmíněné rozhraní aplikace Moodle požaduje.

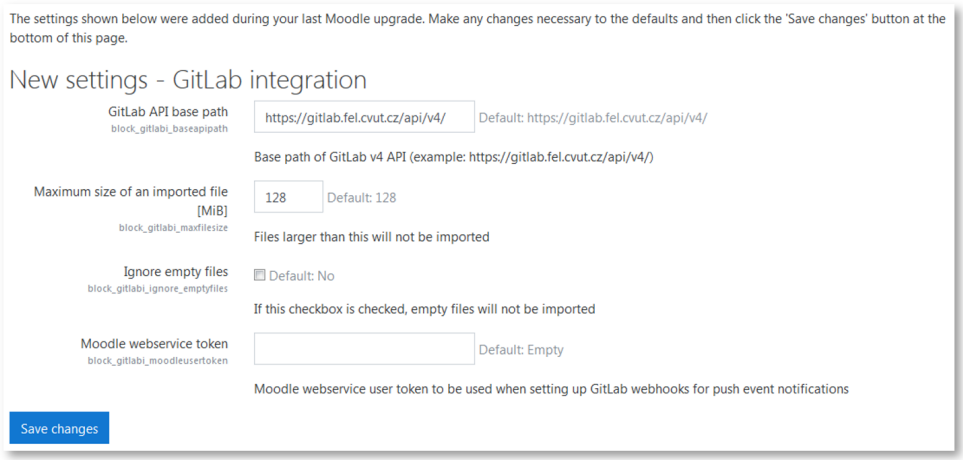
Po nahrání souborů pluginu libovolným ze zmíněných způsobů bude správce v aplikaci Moodle vybídnut k dokončení instalace – tuto obrazovku lze vidět na obrázku 5.2.



Obrázek 5.2. Rozhraní aplikace Moodle pro dokončení instalace pluginu.

5.2 Konfigurace

Po dokončení instalace umožní Moodle správci provést úvodní konfiguraci pluginu, jak je zobrazeno na obrázku 5.3. Při této konfiguraci je nutné ponechat volbu *Moodle web-service token* nevyplněnou, jelikož potřebný webservice uživatel může být vytvořen až dodatečně. Problematice tohoto nastavení se věnuje část 5.2.1.

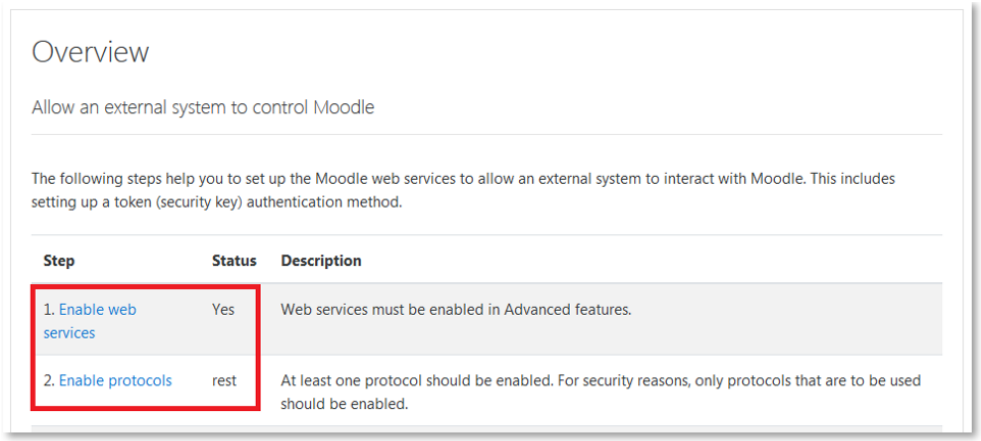


Obrázek 5.3. Úvodní konfigurace pluginu GitLab integration.

5.2.1 Moodle webservice token

Webservice slouží webhooku z aplikace GitLab ke spouštění aktualizačních importů. Aby tato součást fungovala, je nutné, aby byla aplikace Moodle správně nakonfigurovaná, konkrétně musí být povoleno použití webservices a protokolu REST. To znamená, že v administraci aplikace Moodle na stránce *Site administration* → *Plugins* → *Web services* → *Overview* musí být aktivována volba *Enable web services* a dále musí být aktivovaný protokol *rest*. Podobu této stránky v případě správné konfigurace je možné vidět na obrázku 5.4.

Dalším krokem pro zprovoznění webservice implementovaného pluginu je vytvoření uživatelského tokenu pro použití jeho služby. Nejprve je potřeba mít připraveného uživatele s patřičnou rolí. Webservice pluginu v tomto případě vyžaduje uživatele se systémovou rolí, která má následující capabilities (výsledek oprávnění na těchto capabilities v kontextu *system* musí být *allow*):



Obrázek 5.4. Správná konfigurace webservices v aplikaci Moodle.

- `moodle/course:update`,
- `moodle/course:view`.

Následuje vytvoření samotného tokenu pro tohoto uživatele. Na stránce *Site administration* → *Plugins* → *Web services* → *Manage tokens* se nachází tlačítko *Add*, po jehož stisknutí se zobrazí stránka pro vytvoření nového tokenu. Zde je potřeba vybrat správného uživatele z předchozího kroku (pole *User*) a správnou službu (*Service*) – v tomto případě `block_gitlabi_gitlab_repo_push` a kliknout na tlačítko *Save changes*. Ze stránky, která se zobrazí, je možné zkopírovat nově vytvořený token do pole *Moodle webservice token* v konfiguraci pluginu.

5.2.2 Možnosti konfigurace pluginu

V této části jsou vysvětleny jednotlivé konfigurační možnosti pluginu, kterými jsou:

- **GitLab API base path:** Základní cesta k API aplikace GitLab, kterou bude plugin používat.
- **Maximum size of an imported file:** Soubory, jejichž velikost překračuje zadanou hodnotu, nebudou importovány. Hodnota se zadává v MiB.
- **Ignore empty files:** Pokud je tento checkbox zaškrtnutý, budou během importu ignorovány (a tedy neimportovány) prázdné soubory (soubory s velikostí 0 B). Toto nastavení nemá vliv na interpretaci souboru `hidden.txt`, který může být prázdný a přesto bude příslušná sekce při jeho přítomnosti označena jako skrytá.
- **Moodle webservice token:** token webservice uživatele, pomocí kterého budou spouštěny aktualizační importy. Pokud toto pole nebude mít vyplněnou hodnotu, uživatelům pluginu nebude umožněno používat import s automatickým sledováním změn. Další informace o tom, jak toto nastavení používat, je možné nalézt v části 5.2.1.

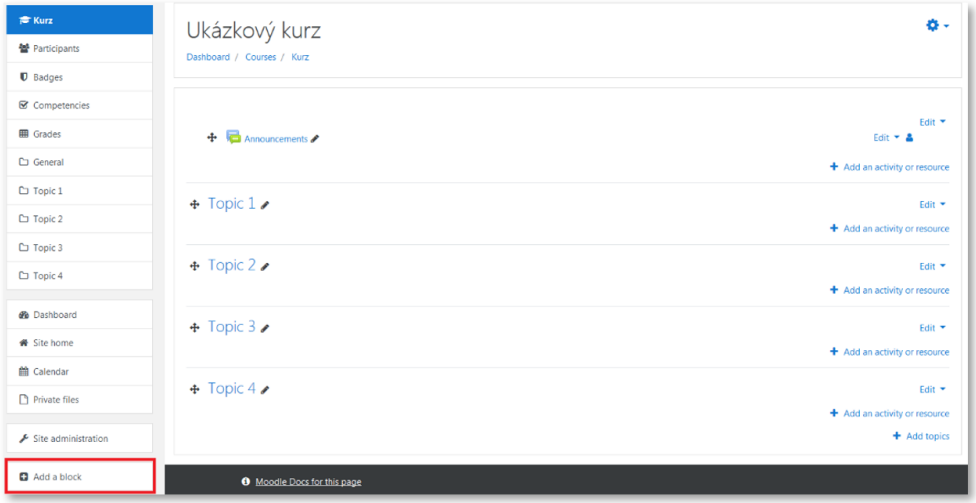
Kapitola 6
Používání pluginu

Po té, co správce aplikace Moodle provede instalaci a konfiguraci pluginu, mohou ho uživatelé začít používat.

Před samotným importem je nutné do příslušného kurzu přidat instanci bloku pluginu, ve které se nachází jeho uživatelské rozhraní. Poté je nutné nastavit personal access token z aplikace GitLab. Následně už je možné provést samotný import obsahu kurzu.

6.1 Zobrazení uživatelského rozhraní pluginu v kurzu

Hlavní uživatelské rozhraní pluginu se nachází v bloku, jehož instanci je nejprve třeba do kurzu přidat. To je možné učinit tlačítkem *Add block* (zvýrazněno na obrázku 6.1) po aktivaci editačního módu a následným výběrem možnosti *GitLab integration*.

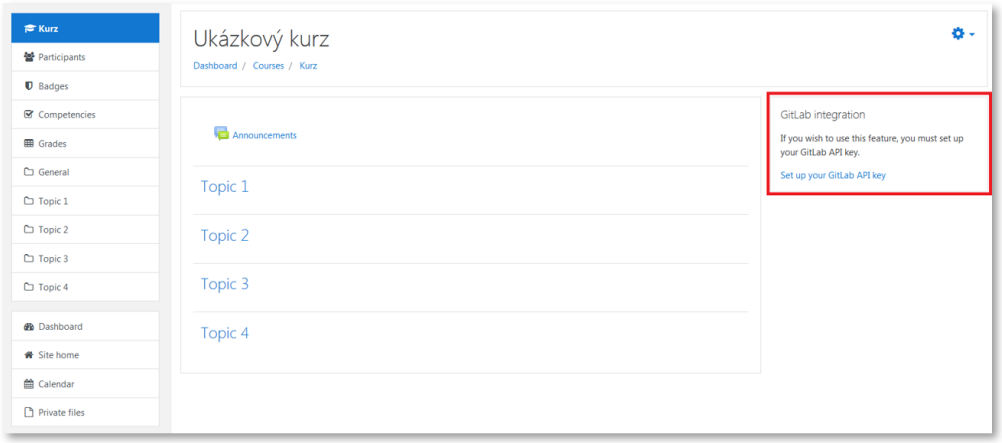


Obrázek 6.1. Tlačítko *Add block* pro přidání instance bloku pluginu do kurzu v aplikaci Moodle.

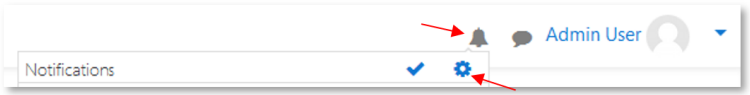
Po přidání instance bloku se zobrazí uživatelské rozhraní (zobrazeno na obrázku 6.2), pomocí kterého lze plugin ovládat.

6.2 Povolení notifikací

Pokud si uživatel přeje od pluginu dostávat notifikace o stavu importů, je nutné tuto funkci povolit, pro správné fungování pluginu to však není nutné. Toto nastavení lze provést kliknutím na ikonku zvonku vlevo od uživatelského jména v pravém horním rohu uživatelského rozhraní aplikace Moodle a následným kliknutím na ozubené kolo, jak je zobrazeno na obrázku 6.3.

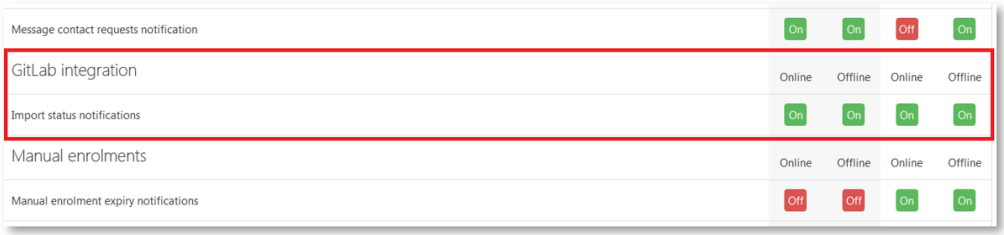


Obrázek 6.2. Instance bloku GitLab integration v kurzu v aplikaci Moodle.



Obrázek 6.3. Postup k zobrazení nastavení notifikací v aplikaci Moodle.

Na stránce, která se otevře, nesmí být zaškrtnutý checkbox *Disable notifications* (globální zákaz notifikací). Dále je nutné najít v seznamu položku *GitLab integration* a pod ní položku *Import status notifications* – zde je možné povolit či zakázat notifikace od pluginu GitLab integration. Levá dvě tlačítka slouží pro nastavení notifikací přímo v aplikaci Moodle, pravá dvě pak pro nastavení e-mailových notifikací. Nastavení se všemi povolenými notifikacemi (v aplikaci Moodle i na e-mail, pokud je uživatel online i offline) od pluginu GitLab integration je možné vidět na obrázku 6.4.



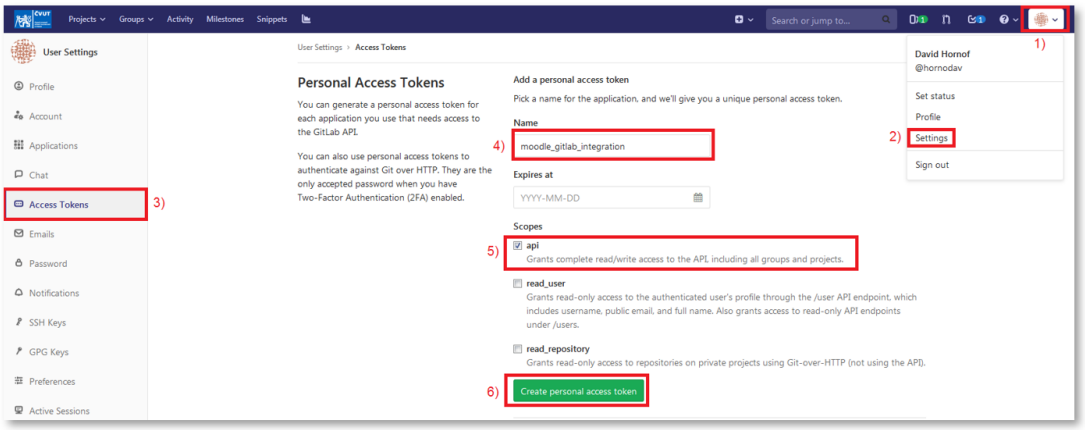
Obrázek 6.4. Nastavení notifikací pluginu GitLab integration.

6.3 Nastavení personal access tokenu z aplikace GitLab

Nezbytným krokem před importem kurzu je nastavení personal access tokenu z aplikace GitLab (v uživatelském rozhraní pluginu označovaný jako *GitLab API key*). Tento token využívá plugin pro autentifikaci uživatele, která je nutná pro přístup k jeho repozitářům. Plugin vyžaduje *scope* (rozsah) tokenu `api`. Tento krok však není nutné vykonávat pro každý kurz, plugin si nastavený token pamatuje pro účet uživatele aplikace Moodle.

Pro vytvoření tokenu v aplikaci GitLab lze po přihlášení využít následujícího postupu (postup je ilustrován na obrázku 6.5):

1. Kliknout na avatar uživatele vpravo nahoře.
2. Kliknout na položku *Settings*.
3. Kliknout na položku *Access Tokens* v levém menu.
4. Vyplnit název tokenu (může být libovolný, slouží uživateli pro následnou identifikaci).
5. Povolení *scope* `api`.
6. Kliknout na tlačítko *Create personal access token*.



Obrázek 6.5. Postup k vytvoření personal access tokenu v aplikaci GitLab.

Po dokončení posledního kroku aplikace GitLab obnoví danou stránku a uživateli se zobrazí nově vygenerovaný token. Tuto hodnotu (token) je třeba zkopírovat do pole *New API key* dostupného po kliknutí na tlačítko *Set up your GitLab API key* v uživatelském rozhraní pluginu GitLab integration. Po kliknutí na tlačítko *Save changes* plugin ověří platnost vloženého tokenu a pokud je tato kontrola úspěšná, token uloží. Od této chvíle je možné provádět import obsahu kurzu. Také je možné zadaný token změnit, nebo případně odstranit.

6.4 Import obsahu kurzu

Plugin nabízí dvě možnosti importu obsahu z aplikace GitLab: *jednorázový import* a *import s automatickým sledováním změn*. Jednorázový import pouze importuje obsah z aplikace GitLab, oproti tomu import s automatickým sledováním změn kromě importu nastaví v aplikaci GitLab webhook, který zajistí reakci na události push odehrávající se ve zdrojové větvi, díky čemuž ve chvíli, kdy dojde k takové události (push do zdrojové větve), bude spuštěn aktualizační import, který aplikuje provedené změny do Moodle kurzu.

Je důležité poznamenat, že před samotným importem dojde k odstranění veškerého existujícího obsahu z cílového kurzu. Pokud tento kurz není prázdný, je vhodné před importem jeho obsah zálohovat.

6.4.1 Pravidla importu

Převod souborů z GitLab repozitáře do obsahu kurzu v aplikaci Moodle probíhá na základě pravidel definovaných v části 3.4. Tato pravidla se dají shrnout v následujících bodech:

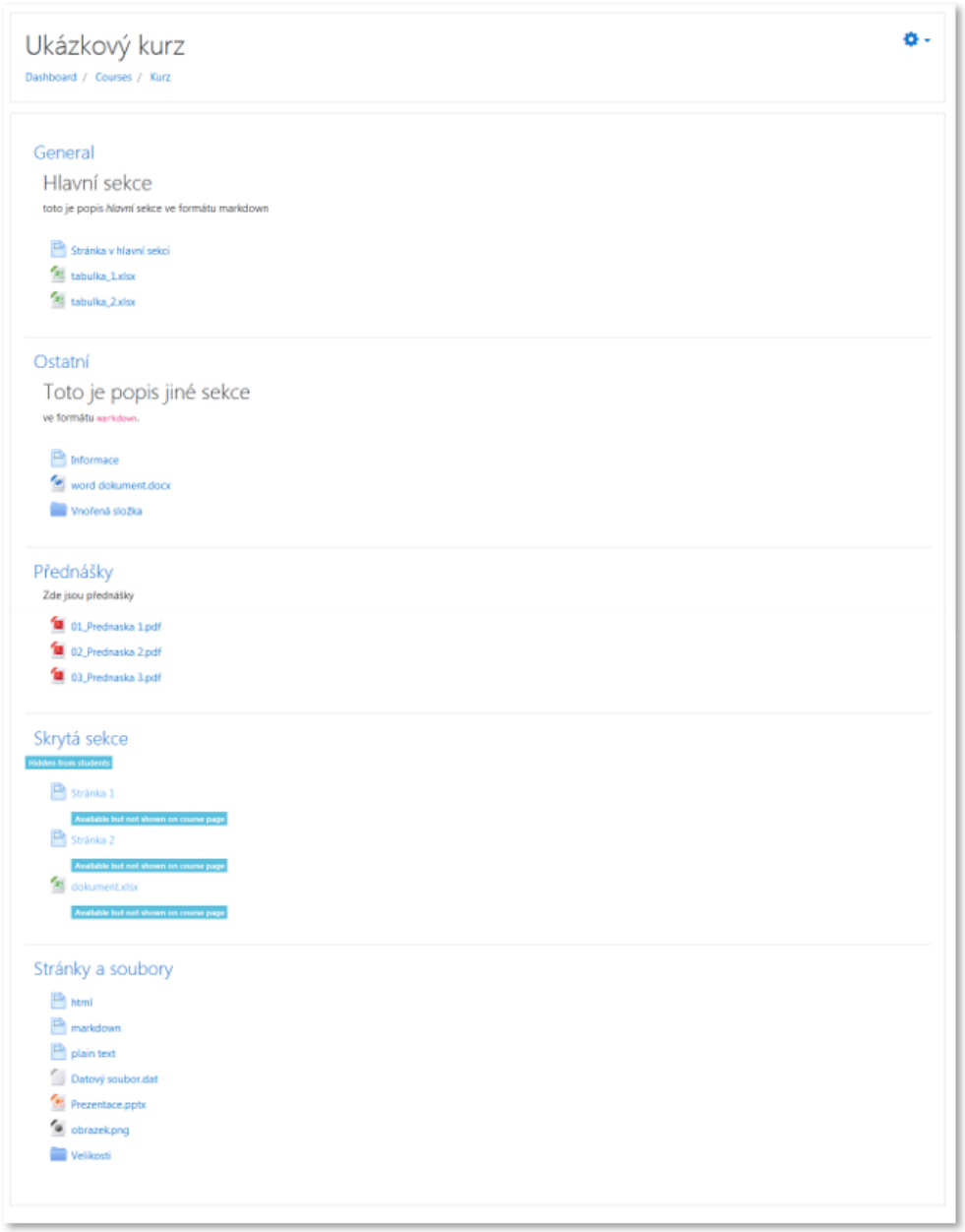
- Soubory v kořenovém adresáři repozitáře jsou importovány jako soubory (instance modulu *resource*) do hlavní sekce kurzu.

- Podadresáře v kořenovém adresáři jsou importovány jako sekce kurzu.
- Soubory s příponami `html`, `md` a `txt` nejsou importovány jako soubory, nýbrž jako stránky (instance modulu *page*) v příslušném formátu.
- Soubory `index.html`, `index.md` a `index.txt` nejsou importovány jako soubory ani jako stránky, nýbrž jako popis příslušné sekce.
- Pokud se v podadresáři první úrovně nachází soubor `hidden.txt`, není tento soubor importován, ale odpovídající sekce se označí jako skrytá před studenty.
- Podadresáře druhé úrovně jsou importovány jako složky (instance modulu *folder*) a jsou do nich umístěny veškeré soubory v nich.
- Podadresáře vyšší než druhé úrovně jsou spolu s jejich obsahem ignorovány.

Pravidla pro převod vychází z možností systému Git a snaží se o jednoduchost a předvídatelnost převodu.

Jako příklad převodu může sloužit ukázkový repozitář, jehož adresářová struktura je uvedena níže. Kurz, do kterého byl obsah tohoto repozitáře importován, je zobrazen na obrázku 6.6.

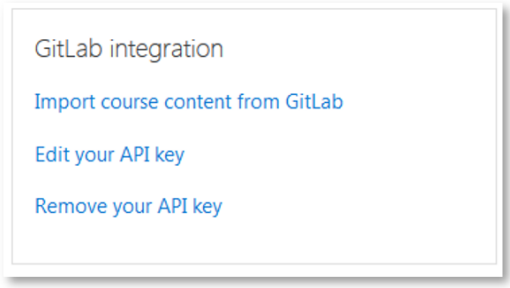
```
.
|-- Ostatní
|   |-- Vnořená složka
|   |   |-- Excel.xlsx
|   |   |-- Powerpoint.pptx
|   |   `-- Word.docx
|   |-- Informace.txt
|   |-- index.md
|   `-- word dokument.docx
|-- Přednášky
|   |-- 01_Prednaska 1.pdf
|   |-- 02_Prednaska 2.pdf
|   |-- 03_Prednaska 3.pdf
|   `-- index.txt
|-- Skrytá sekce
|   |-- dokument.xlsx
|   |-- Stránka 1.txt
|   |-- Stránka 2.txt
|   `-- hidden.txt
|-- Stránky a soubory
|   |-- Velikosti
|   |   |-- 3MiB.dat
|   |   |-- mene_nez_3MiB.dat
|   |   `-- vice_nez_3MiB.dat
|   |-- Datový soubor.dat
|   |-- Prezentace.pptx
|   |-- html.html
|   |-- markdown.md
|   |-- obrazek.png
|   `-- plain text.txt
|-- Stránka v hlavní sekci.html
|-- index.md
|-- tabulka_1.xlsx
`-- tabulka_2.xlsx
```



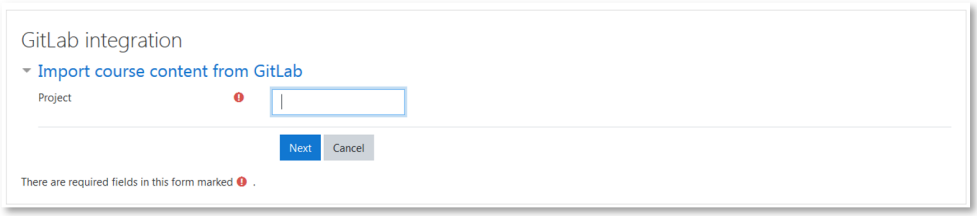
Obrázek 6.6. Kurz v aplikaci Moodle po importu ukázkového repozitáře.

6.4.2 Spuštění importu

Na obrázku 6.7 je zobrazeno uživatelské rozhraní pluginu s nastaveným personal access tokenem z aplikace GitLab. Prvním krokem pro spuštění importu je kliknutí na možnost *Import course content from GitLab*. Následuje zadání názvu projektu z aplikace GitLab, z něhož má být obsah importován (obrázek 6.8). Po kliknutí na tlačítko *Next* plugin vyhledá všechny projekty dostupné uživateli odpovídající zadanému názvu. Pokud je nalezen pouze jeden projekt, plugin vybídne k výběru zdrojové větve (obrázek 6.9). V opačném případě plugin nejprve vybídne k výběru konkrétního projektu ze seznamu nalezených projektů, a až poté vybídne k výběru zdrojové větve.

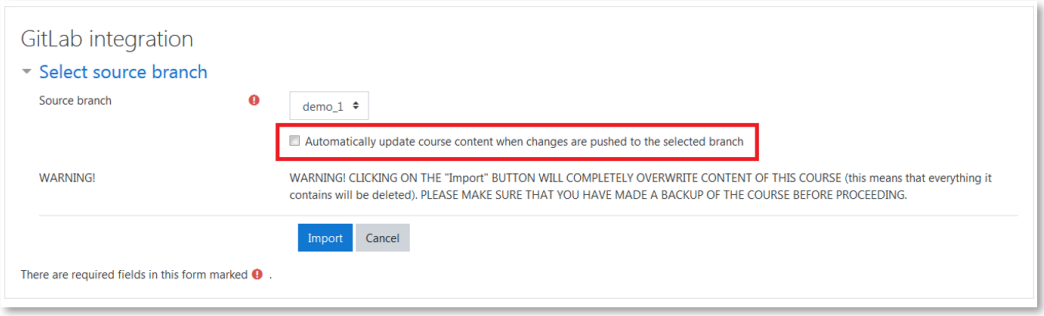


Obrázek 6.7. Uživatelské rozhraní GitLab integration pluginu s nastaveným personal access tokenem z aplikace GitLab.



Obrázek 6.8. Uživatelské rozhraní GitLab integration pluginu pro zadání jména zdrojového projektu z aplikace GitLab.

Při výběru zdrojové větve je možné nastavit import tak, aby sledoval změny, jak je zmíněno v části 6.4. Toto nastavení se provádí checkboxem[1] zvýrazněným na obrázku 6.9. Pokud checkbox není zaškrtnutý, provede se jednorázový import. V opačném případě dojde k nastavení importu s automatickým sledováním změn a úvodnímu úplnému importu obsahu kurzu.



Obrázek 6.9. Uživatelské rozhraní GitLab integration pluginu pro výběr zdrojové větve se zvýrazněným checkboxem pro nastavení importu s automatickým sledováním změn.

Kliknutím na tlačítko *Import* dojde ke spuštění samotného procesu importu, jehož součástí je dříve zmíněné odstranění veškerého existujícího obsahu kurzu.

Import kurzu je časově náročná operace. Pokud to uživatel pluginu dovolil, bude mu zaslána notifikace ve chvíli, kdy bude import dokončen. Stejně tak bude zaslána notifikace ve chvíli, kdy dojde ke spuštění aktualizačního importu. Notifikace je zaslána i v případě, že import selže – v takovém případě bude notifikace obsahovat důvod, proč k selhání došlo.

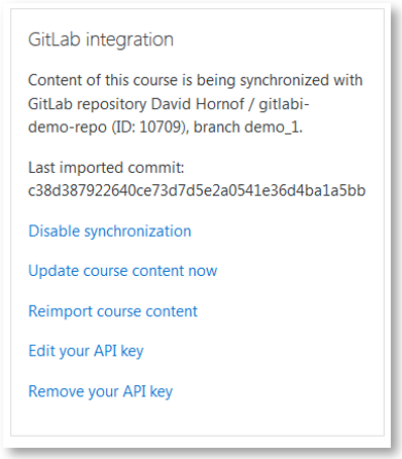
[1] Checkbox nebude zobrazen (a tedy nebude možné nastavit import s automatickým sledováním změn) pokud nebyl nastaven token webservice uživatele – více o nastavení webservice aplikace Moodle je možné nalézt v části 5.2.1.

6.4.3 Dodatečné možnosti importu s automatickým sledováním změn

U kurzu, ve kterém je nastavený import s automatickým sledováním změn, je možné aktualizaci ručně spustit, stejně jako je možné provést kompletní reimport kurzu (zrušení nastaveného importu a jeho následné znovuvytvoření spolu s úplným importem). Tyto možnosti jsou dostupné pouze uživateli, který import nastavil.

Možné je též nastavený import zrušit, čímž dojde k ukončení sledování událostí push ve zdrojové větvi zdrojového repozitáře (dojde k odstranění webhooku). Importovaný obsah kurzu zůstane zachován. Tuto akci může provést kdokoli, kdo má právo upravovat daný kurz.

Na obrázku 6.10 je možné vidět uživatelské rozhraní pluginu u kurzu s nastaveným importem s automatickým sledováním změn.



Obrázek 6.10. Uživatelské rozhraní GitLab integration pluginu u kurzu s nastaveným importem s automatickým sledováním změn.

Kapitola 7
Testování

Testování pluginu probíhalo na několika úrovních. Během vývoje docházelo k průběžnému testování pomocí smoke testů pro ověřování základní funkčnosti implementovávaného řešení. Po dokončení implementace bylo provedeno otestování pluginu na základě připravených testovacích scénářů. Dále proběhlo testování metodou exploratory testingu a otestování skutečným uživatelem pluginu.

7.1 Smoke testy

Smoke testy byly používány při vývoji k rychlému ověřování, zdali je plugin schopný plnit své základní úkoly, zejména správně provést úplný import obsahu kurzu (správně vytvořit instance jednotlivých modulů) a tento obsah následně správně aktualizovat v rámci aktualizačního importu. Tímto způsobem však byly testovány i některé dodatečné funkce, jako například odstranění nastaveného importu s automatickým sledováním změn při odstranění kurzu.

7.2 Testování podle scénářů

Testování podle scénářů tvořilo hlavní akceptační kritérium pluginu – plugin mohl být prohlášen za korektně fungující, pokud úspěšně prošel všemi testovacími scénáři. Autorem práce byly vytvořeny 4 scénáře pokrývající hlavní funkcionality pluginu, konkrétně:

- `GITLABI_UAT_APIK01`: Scénář pro ověření schopnosti přijmout a správně validovat zadaný personal access token z aplikace GitLab, včetně reakce pluginu na chybné tokeny.
- `GITLABI_UAT_CIMP01`: Scénář pro ověření schopnosti provést úplný import – správně vytvořit všechny podporované typy modulů, sekce a správně vytvořené entity seřadit.
- `GITLABI_UAT_CIMP02`: Scénář pro ověření schopnosti provést úplný import stejně jako v předchozím bodě, avšak pro kurz s prázdnou hlavní sekcí. Jelikož každý kurz v Moodle musí obsahovat hlavní sekci (která však může být prázdná), účel tohoto testu spočíval v ověření správného vytvoření této sekce právě v případě, že by měla po importu zůstat prázdná.
- `GITLABI_UAT_UIMP01`: Komplexní scénář pro ověření funkcionalit souvisejících s importem s automatickým sledováním změn. Testována je schopnost nastavení tohoto importu, správného provedení aktualizací na základě různých změn ve zdrojovém repozitáři a také správné zrušení takto nastaveného importu.

Úplnou podobu těchto scénářů je možné nalézt v příloze C tohoto dokumentu. Testování podle scénářů probíhalo v nezávislém testovacím prostředí.

7.3 Exploratory testing

Kromě testů provedených autorem byl plugin také otestován v testovacím prostředí aplikace Moodle Fakulty elektrotechnické testerem z Centra znalostního managementu. Toto testování probíhalo pomocí metody exploratory testingu, při které měl tester k dispozici pouze uživatelskou příručku a testování prováděl na základě své vlastní iniciativy.

7.4 Testování skutečným uživatelem

Plugin byl také otestován vyučujícím z katedry Radioelektrotechniky v testovacím prostředí Moodle FEL. Jednalo se o pokročilého uživatele systému Git, kterého, podle jeho vlastních slov, „nebaví klikat to v Moodle“. Podle jeho vyjádření by chtěl plugin využívat primárně pro usnadnění sdílení menších celků zdrojových kódů v C++ se studenty.

Testování probíhalo bez předem připraveného scénáře, během testu měl uživatel k dispozici uživatelskou příručku. K pluginu vznesl následující připomínky:

- Ocenil by možnost exportu z Moodle zpět do repozitáře v GitLabu. Tato funkce je teoreticky možná, jedná se však o velmi rozsáhlou záležitost.
- Požaduje možnost automatického importu změn ve zdrojovém repozitáři. Plugin toto umožňuje, v době testování však automatické aktualizační importy nefungovaly kvůli problémům s konfigurací testovacího prostředí.
- Ocenil by možnost přejmenovat sekce bez přejmenování složek ve zdrojovém repozitáři. Tato funkce bude do pluginu doplněna před nasazením do produkčního provozu.
- Objevil chybu týkající se zpracování souborů ve formátu Markdown. Tato chyba byla následně opravena.

Celkově však uživatel vyjádřil s pluginem spokojenost.

Kapitola 8
Současný stav a budoucí rozvoj pluginu

Implementovaný plugin je v současnosti nasazený v testovacím prostředí aplikace Moodle Fakulty elektrotechnické, kde proběhlo jeho testování. Nasazení pluginu do produkčního prostředí je plánováno na září 2019.

8.1 Provoz pluginu

V rámci nasazení do produkčního provozu bude potřeba dát pozor zejména na správnou konfiguraci, a to jak aplikace Moodle, tak pluginu samotného – zejména nastavení webservice uživatele. Po uvedení do provozu bude následovat údržba pluginu spolu s poskytováním podpory jeho uživatelům. Součástí poskytování podpory uživatelům bude též sběr jejich zpětné vazby, která bude sloužit jako hlavní podklad pro budoucí rozvoj pluginu.

8.2 Budoucí rozvoj

Jako další krok rozvoje pluginu se nabízí zejména překlad uživatelského rozhraní do českého jazyka pro konzistenci se zbytkem uživatelského rozhraní systému Moodle FEL, které je dostupné v českém i anglickém jazyce. Díky *String API* aplikace Moodle přímo podporující lokalizaci pluginů do jiných jazyků je možné tento krok provést poměrně snadno.

Mezi další možnosti rozvoje v budoucnu patří úprava pravidel pro převod obsahu z aplikace GitLab na základě potřeb uživatelů nebo rozšíření převodu o další moduly s ohledem na možnosti systému Git – jako kandidát se nabízí například modul kniha. Jinou možností rozvoje je také rozšíření podporovaných textových formátů.

Hlavní riziko v budoucnosti pluginu představuje možnost změny API aplikace GitLab. Vzhledem k návrhu pluginu by v takovém případě stačilo upravit wrappery popsané v části 4.3, které slouží k napojení na toto API.

Méně pravděpodobná rizika zahrnují změny v aplikaci Moodle – ať už v databázovém schématu nebo ve výchozích pluginech, jejichž instance jsou vytvářeny. Změny ve výchozích pluginech by nezpůsobovaly větší potíže, jelikož by stačilo upravit adaptér napojující implementovaný plugin na datovou vrstvu aplikace Moodle. Podobně by tomu bylo i v případě změny databázového schématu.

Kapitola 9
Závěr

Cílem této práce bylo vytvořit podpůrný systém umožňující naplnit kurz v systému Moodle obsahem z Git repozitáře hostovaném v prostředí aplikace GitLab. Motivací pro to bylo podpořit a usnadnit přesun studijních materiálů do systému Moodle pro předměty, které zde studijní materiály ještě nemají. Z toho přímo vyplývají i přínosy tohoto podpůrného systému: pro studenty Fakulty elektrotechnické to znamená snadnější přístup k těmto materiálům a pro vyučující znamená více možností, jak studijní materiály připravovat a sdílet se studenty.

V práci byla nejprve provedena analýza, na základě které byl systém navržen, implementován a otestován. Systém byl následně nasazen do testovacího prostředí Moodle FEL a je naplánováno jeho nasazení do produkčního prostředí. Na základě toho lze konstatovat, že cíle této práce byly naplněny.

Nejnáročnější částí práce bylo provedení analýzy systému Moodle. Dokumentace značné části jeho jádra a výchozích pluginů, se kterými implementovaný plugin pracuje, je totiž jen stručná, neúplná, či se v ní hovoří pouze o tom, jak a k čemu některé součásti použít při vývoji nového pluginu, ale už ne o tom, co vlastně dělají. Tento problém byl nejvýraznější při analýze databázového schématu Moodle, ke kterému žádná dokumentace prakticky neexistuje. Díky tomu si tedy z této práce odnáším i zkušenost s orientací se v cizím kódu poměrně velkého systému. Následná implementace podle připraveného návrhu proběhla bez větších problémů. Při nasazení do testovacího prostředí Moodle FEL se objevily problémy s konfigurací tohoto prostředí. Tyto problémy zapříčinily, že budoucí uživatel pluginu, který se podílel na testování, nemohl vyzkoušet automatické aktualizační importy. Příčina těchto problémů je však známá a před produkčním provozem pluginu budou odstraněny.

Literatura

[1] *MoodleDocs: Moodle architecture* [online]. [cit. 2018-11-13]. Dostupné z: https://docs.moodle.org/dev/Moodle_architecture

[2] *MoodleDocs: Plugin types* [online]. [cit. 2018-11-13]. Dostupné z: https://docs.moodle.org/dev/Plugin_types

[3] *MoodleDocs: Web services* [online]. [cit. 2018-11-19]. Dostupné z: https://docs.moodle.org/35/en/Web_services

[4] *MoodleDocs: Web service API functions* [online]. [cit. 2018-11-19]. Dostupné z: https://docs.moodle.org/dev/Web_service_API_functions

[5] *MoodleDocs: Web services API* [online]. [cit. 2018-11-19]. Dostupné z: https://docs.moodle.org/dev/Web_services_API

[6] HODSON, Ryan. *Ry's Git Tutorial*. RyPress, 2014. ISBN 978-0-9850-7231-5.

[7] *Git* [online]. [cit. 2018-11-19]. Dostupné z: https://git-scm.com/

[8] CHACON, Scott a Ben STRAUB. *Pro Git: [everything you need to know about the Git distibuted source control tool]*. 2nd ed. New York, NY: Apress, 2014. ISBN 978-1-4842-0076-6.

[9] *GitLab: Source Code Management* [online]. [cit. 2018-11-19]. Dostupné z: https://about.gitlab.com/product/source-code-management/

[10] *GitLab Docs: Webhooks* [online]. [cit. 2018-11-19]. Dostupné z: https://docs.gitlab.com/ce/user/project/integrations/webhooks.html

[11] *GitLab: Comparing Confusing Terms* [online]. [cit. 2019-04-01]. Dostupné z: https://about.gitlab.com/2017/09/11/comparing-confusing-terms-in-github-bitbucket-and-gitlab/

[12] *GitLab Docs: GitLab API* [online]. [cit. 2018-11-19]. Dostupné z: https://docs.gitlab.com/ce/api

[13] *MoodleDocs: Courses* [online]. [cit. 2018-11-08]. Dostupné z: https://docs.moodle.org/35/en/Courses

[14] *MoodleDocs: Course homepage* [online]. [cit. 2018-11-08]. Dostupné z: https://docs.moodle.org/35/en/Course_homepage

[15] *MoodleDocs: Activity modules* [online]. [cit. 2018-11-13]. Dostupné z: https://docs.moodle.org/dev/Activity_modules

[16] *MoodleDocs: Activities* [online]. [cit. 2018-11-08]. Dostupné z: https://docs.moodle.org/35/en/Activities

[17] *MoodleDocs: Resources* [online]. [cit. 2018-11-08]. Dostupné z: https://docs.moodle.org/35/en/Resources

[18] WILD, Ian. *Moodle 3.x Developer's Guide*. Birmingham: Packt Publishing, 2017. ISBN 978-1-7864-6711-9.

[19] *MoodleDocs: Course formats* [online]. [cit. 2018-11-23]. Dostupné z: https://docs.moodle.org/35/en/Course_formats

[20] *MoodleDocs: Context* [online]. [cit. 2018-11-15]. Dostupné z: https://docs.moodle.org/35/en/Context

[21] *MoodleDocs: Roles and modules* [online]. [cit. 2018-11-15]. Dostupné z: https://docs.moodle.org/dev/Roles_and_modules

[22] NEWMAN, Sam. *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, CA: O'Reilly, 2015. ISBN 978-1-4919-5035-7.

[23] *MoodleDocs: Access API* [online]. [cit. 2018-11-15]. Dostupné z: https://docs.moodle.org/dev/Access_API

[24] *MoodleDocs: File API internals* [online]. [cit. 2018-11-15]. Dostupné z: https://docs.moodle.org/dev/File_API_internals

[25] *MoodleDocs: Core APIs* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.moodle.org/dev/Core_APIs

[26] *GitLab Docs: Repositories API* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.gitlab.com/ce/api/repositories.html

[27] *GitLab Docs: Repository files API* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.gitlab.com/ce/api/repository_files.html

[28] *GitLab Docs: GitLab as an OAuth2 provider* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.gitlab.com/ce/api/oauth2.html

[29] *GitLab Docs: Personal access tokens* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.gitlab.com/ce/user/profile/personal_access_tokens.html

[30] *GitLab Docs: Deploy tokens* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.gitlab.com/ce/user/project/deploy_tokens/index.html

[31] *GitLab Docs: GitLab and SSH keys* [online]. [cit. 2018-11-16]. Dostupné z: https://docs.gitlab.com/ce/ssh/

[32] *SSH Key* [online]. [cit. 2018-11-16]. Dostupné z: https://www.ssh.com/ssh/key/

[33] *MoodleDocs: Blocks* [online]. [cit. 2018-11-17]. Dostupné z: https://docs.moodle.org/dev/Blocks

[34] *GitLab Docs: Branches API* [online]. [cit. 2018-11-23]. Dostupné z: https://docs.gitlab.com/ce/api/branches.html

[35] *GitLab Docs: Commits API* [online]. [cit. 2018-11-23]. Dostupné z: https://docs.gitlab.com/ce/api/commits.html

[36] *GitLab Docs: Projects API* [online]. [cit. 2018-11-23]. Dostupné z: https://docs.gitlab.com/ce/api/projects.html

[37] *MoodleDocs: User preferences for plug-ins* [online]. [cit. 2018-11-17]. Dostupné z: https://docs.moodle.org/dev/User_preferences_for_plug-ins

[38] *MoodleDocs: Preference API* [online]. [cit. 2018-11-17]. Dostupné z: https://docs.moodle.org/dev/Preference_API

[39] *OWASP: XSS Filter Evasion Cheat Sheet* [online]. [cit. 2018-11-23]. Dostupné z: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

[40] *MoodleDocs: Task API* [online]. [cit. 2018-11-22]. Dostupné z: https://docs.moodle.org/dev/Task_API

[41] *MoodleDocs: Message API* [online]. [cit. 2018-11-18]. Dostupné z: https://docs.moodle.org/dev/Message_API

[42] *Git: gitattributes documentation* [online]. [cit. 2019-04-18]. Dostupné z: https://git-scm.com/docs/gitattributes

[43] *Usage of UTF-8 for websites* [online]. [cit. 2019-04-18]. Dostupné z: https://w3techs.com/technologies/details/en-utf8/all/all

[44] *MoodleDocs: String API* [online]. [cit. 2019-04-18]. Dostupné z: https://docs.moodle.org/dev/String_API

[45] *MoodleDocs: Plugin files* [online]. [cit. 2019-04-18]. Dostupné z: https://docs.moodle.org/dev/Plugin_files

[46] *MoodleDocs: XMLDB documentation* [online]. [cit. 2019-04-18]. Dostupné z: https://docs.moodle.org/dev/XMLDB_Documentation

[47] *MoodleDocs: Automatic class loading* [online]. [cit. 2019-04-18]. Dostupné z: https://docs.moodle.org/dev/Automatic_class_loading

[48] MILETIC, Darko. *Moodle Security*. Birmingham: Packt Publishing, 2011. ISBN 978-1-8495-1264-0.

[49] *HTML Purifier* [online]. [cit. 2019-04-02]. Dostupné z: http://htmlpurifier.org/

Příloha A
Použité zkratky

API	Application programming interface – aplikační programové rozhraní
ČVUT	České vysoké učení technické v Praze
FEL	Fakulta elektrotechnická
HTML	Hypertext markup language – značkovací jazyk
HTTP	Hypertext transfer protocol – komunikační protokol
JSON	JavaScript object notation – datový formát
KiB	Kibibyte (2^{10} bytů)
MiB	Mebibyte (2^{20} bytů)
REST	Representational state transfer – architektura pro komunikaci klienta se serverem pomocí protokolu HTTP
URL	Uniform resource locator
XML	Extensible markup language – značkovací jazyk
XSS	Cross-site scripting – druh útoku na webové aplikace

Příloha B
Obsah přiloženého CD

- `plugin/gitlabi` – adresář obsahující soubory implementovaného pluginu.
- `plugin/gitlabi.zip` – zip archiv obsahující implementovaný plugin, pomocí kterého lze provést instalaci pluginu do aplikace Moodle (více informací lze nalézt v kapitole 5).
- `thesis/tex` – adresář obsahující zdrojové soubory tohoto dokumentu ve formátu TEX.
- `thesis/thesis.pdf` – tento dokument ve formátu PDF.
- `demo_repo` – adresář obsahující soubory ukázkového repozitáře.
- `test_cases_repos` – adresář obsahující soubory testovacích repozitářů pro jednotlivé testovací scénáře.

Příloha C
Testovací scénáře

C.1 Test rozhraní pro zadávání personal access tokenu z aplikace GitLab

Název testu: Test rozhraní pro zadávání personal access tokenu z aplikace GitLab

Identifikátor testu: `GITLABI_UAT_APIKO1`

Účel testu: Otestovat funkčnost a schopnost rozhraní pro zadávání personal access tokenu z aplikace GitLab vyrovnat se s běžnými chybami.

Popis testu: V testu jsou nejprve testovány chybné vstupy: nezadaný token a neplatný token. Následně je testován platný vstup (platný a správně nastavený token) a jeho následné odstranění.

Vstupní podmínky:

- V aplikaci Moodle je testovaný plugin nainstalován a nakonfigurován.
- V aplikaci Moodle je vytvořený testovací kurz, ve kterém má uživatel provádějící test právo provádět změny. V tomto kurzu není nastaven import s automatickým sledováním změn.
- Do testovacího kurzu byl přidán blok GitLab integration (blok s uživatelským rozhraním testovaného pluginu).
- Uživatel provádějící test **nemá** v testovaném pluginu nastavený personal access token z aplikace GitLab.

Testovací data:

- Přihlašovací údaje do aplikace Moodle uživatele provádějícího test.
- Platný personal access token z aplikace GitLab s nasteveným scope `api`.

Poznámky:

V uživatelském rozhraní pluginu je personal access token z aplikace GitLab nazýván jako *API key* (*API klíč*).

Kroky testu:

1. Přihlaste se do aplikace Moodle a přejděte na testovací kurz.
   Očekávaný výsledek: Zobrazí se testovací kurz spolu s viditelným uživatelským rozhraním testovaného pluginu, ve kterém je zobrazena hláška o nutnosti nastavení API klíče a jediná volba *Set up your GitLab API key*.
2. Klikněte na volbu *Set up your GitLab API key*.
   Očekávaný výsledek: Zobrazí se formulář pro zadání API klíče z aplikace GitLab.

3. Klikněte na tlačítko *Save changes*.
   Očekávaný výsledek: Zobrazí se chybová hláška oznamující uživateli, že je nutné vyplnit hodnotu v poli *New API key*.
4. Do pole *New API key* zadejte hodnotu `test` a klikněte na tlačítko *Save changes*.
   Očekávaný výsledek: Zobrazí se chybová hláška oznamující uživateli, že zadaný API klíč je neplatný.
5. Do pole *New API key* vložte připravený platný personal access token z aplikace GitLab s nastaveným scope `api` a klikněte na tlačítko *Save changes*.
   Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že zadaný API klíč byl úspěšně uložen a dojde k přesměrování zpět na kurz. Po přesměrování se v uživatelském rozhraní testovaného pluginu zobrazují možnosti *Import course content from GitLab*, *Edit your API key* a *Remove your API key*.
6. Klikněte na volbu *Remove your API key*.
   Očekávaný výsledek: Zobrazí se formulář pro potvrzení zvolené akce.
7. Klikněte na tlačítko *Yes*.
   Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že API klíč byl úspěšně odstraněn a dojde k přesměrování zpět na kurz. Po přesměrování je v uživatelském rozhraní testovaného pluginu zobrazena hláška o nutnosti nastavení API klíče a jediná volba *Set up your GitLab API key*.

C.2 Test úplného importu – úplný obsah

Název testu: Test úplného importu – úplný obsah

Identifikátor testu: `GITLABI_UAT_CIMP01`

Účel testu: Otestovat funkcionalitu úplného importu.

Popis testu: Testován je úplný import neprázdné hlavní sekce s popisem, ostatních sekcí s popisem i bez popisu a jejich obsahu včetně vnořené složky, dále pak import skryté sekce.

Vstupní podmínky:

- V aplikaci Moodle je testovaný plugin nainstalován a nakonfigurován.
- V aplikaci Moodle je vytvořený testovací kurz, ve kterém má uživatel provádějící test právo provádět změny. V tomto kurzu není nastaven import s automatickým sledováním změn.
- V nastavení pluginu GitLab integration je povolen import prázdných souborů.
- Do testovacího kurzu byl přidán blok GitLab integration (blok s uživatelským rozhraním testovaného pluginu).
- Uživatel provádějící test povolil v aplikaci Moodle pluginu GitLab integration zasílání notifikací.
- Uživatel provádějící test má v testovaném pluginu nastavený platný personal access token z aplikace GitLab.
- Testovací repozitář se nachází v aplikaci GitLab, odkud bude importován. Uživatel má v tomto repozitáři roli alespoň *Reporter*.

Testovací data:

- Přihlašovací údaje do aplikace Moodle uživatele provádějícího test.
- Testovací repozitář s následující strukturou:

```
.
|-- sekce1
|   |-- index.html
|   |-- jina_stranka.txt
|   `-- test2.pdf
|-- sekce2
|   |-- slozka
|   |   `-- test3.pdf
|   `-- hidden.txt
|-- index.txt
|-- stranka.md
|-- test1.pdf
`-- ztest.pdf
```

Kroky testu:

1. Přihlaste se do aplikace Moodle a přejděte na testovací kurz.
   Očekávaný výsledek: Zobrazí se testovací kurz spolu s viditelným uživatelským rozhraním testovaného pluginu, ve kterém jsou možnosti *Import course content from GitLab*, *Edit your API key* a *Remove your API key*.
2. Klikněte na volbu *Import course content from GitLab*.

Očekávaný výsledek: Zobrazí se formulář pro zadání názvu projektu, ze kterého má být proveden import.

3. Zadejte správný název projektu a klikněte na tlačítko *Next*.
   Očekávaný výsledek: V případě, že zadanému jménu odpovídá více projektů, zobrazí se formulář pro výběr konkrétního projektu – v takovém případě vyberte správný projekt a klikněte na tlačítko *Next*. V opačném případě nebo po dokončení předchozího subkroku se zobrazí formulář pro výběr zdrojové větve.
4. Vyberte správnou zdrojovou větev a ujistěte se, že na stránce **není** žádný zaškrtnutý checkbox. Následně klikněte na tlačítko *Import*.
   Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že import byl úspěšně spuštěn.
5. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.
   Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.
6. Přejděte do testovacího kurzu a zkontrolujte jeho obsah.
   Očekávaný výsledek: V kurzu se nachází importovaný obsah odpovídající testovacímu repozitáři, konkrétně:

   - Hlavní sekci má popis odpovídající obsahu souboru `index.txt` v kořenovém adresáři repozitáře.
   - V hlavní sekci se nachází stránka `stranka`, jejíž obsah odpovídá souboru `stranka.md` v kořenovém adresáři repozitáře.
   - V hlavní sekci se nacházejí soubory `test1.pdf` a `ztest.pdf`. Tyto soubory jsou seřazeny v abecedním pořadí (tedy `test1.pdf` je před `ztest.pdf`).
   - V kurzu se kromě hlavní sekce nacházejí sekce `sekce1` a `sekce2`.
   - Sekce `sekce2` je skrytá před studenty.
   - Sekce `sekce1` má popis odpovídající obsahu souboru `index.html` ve složce `sekce1` v repozitáři.
   - V sekci `sekce1` se nachází stránka `jina_stranka`, jejíž obsah odpovídá souboru `jina_stranka.txt` ve složce `sekce1` v repozitáři.
   - V sekci `sekce1` se nachází soubor `test2.pdf`.
   - Sekce `sekce2` nemá žádný popis.
   - V sekci `sekce2` se nachází složka `slozka`, která obsahuje soubor `test3.pdf`.

C.3 Test úplného importu – prázdná hlavní sekce

Název testu: Test úplného importu – prázdná hlavní sekce

Identifikátor testu: `GITLABI_UAT_CIMP02`

Účel testu: Otestovat funkcionalitu úplného importu pro kurz, který po importu bude mít prázdnou hlavní sekci.

Popis testu: Testován je úplný import prázdné hlavní sekce spolu s importem jiných sekcí s různým obsahem.

Vstupní podmínky:

- V aplikaci Moodle je testovaný plugin nainstalován a nakonfigurován.
- V aplikaci Moodle je vytvořený testovací kurz, ve kterém má uživatel provádějící test právo provádět změny. V tomto kurzu není nastaven import s automatickým sledováním změn.
- V nastavení pluginu GitLab integration je povolen import prázdných souborů.
- Do testovacího kurzu byl přidán blok GitLab integration (blok s uživatelským rozhraním testovaného pluginu).
- Uživatel provádějící test povolil v aplikaci Moodle pluginu GitLab integration zasílání notifikací.
- Uživatel provádějící test má v testovaném pluginu nastavený platný personal access token z aplikace GitLab.
- Testovací repozitář se nachází v aplikaci GitLab, odkud bude importován. Uživatel má v tomto repozitáři roli alespoň *Reporter*.

Testovací data:

- Přihlašovací údaje do aplikace Moodle uživatele provádějícího test.
- Testovací repozitář s následující strukturou:

```
.
|-- sekce1
|   |-- index.html
|   |-- jina_stranka.txt
|   `-- test2.pdf
`-- sekce2
    |-- slozka
    |   `-- test3.pdf
    `-- hidden.txt
```

Kroky testu:

1. Přihlaste se do aplikace Moodle a přejděte na testovací kurz.
   Očekávaný výsledek: Zobrazí se testovací kurz spolu s viditelným uživatelským rozhraním testovaného pluginu, ve kterém jsou možnosti *Import course content from GitLab*, *Edit your API key* a *Remove your API key*.
2. Klikněte na volbu *Import course content from GitLab*.
   Očekávaný výsledek: Zobrazí se formulář pro zadání názvu projektu, ze kterého má být proveden import.
3. Zadejte správný název projektu a klikněte na tlačítko *Next*.

Očekávaný výsledek: V případě, že zadanému jménu odpovídá více projektů, zobrazí se formulář pro výběr konkrétního projektu – v takovém případě vyberte správný projekt a klikněte na tlačítko *Next*. V opačném případě nebo po dokončení předchozího subkroku se zobrazí formulář pro výběr zdrojové větve.

4. Vyberte správnou zdrojovou větev a ujistěte se, že na stránce **není** žádný zaškrtnutý checkbox. Následně klikněte na tlačítko *Import*.
   Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že import byl úspěšně spuštěn.
5. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.
   Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.
6. Přejděte do testovacího kurzu a zkontrolujte jeho obsah.
   Očekávaný výsledek: V kurzu se nachází importovaný obsah odpovídající testovacímu repozitáři, konkrétně:

   - Hlavní sekce kurzu je prázdná (nezobrazuje se pokud není aktivní režim úprav).
   - V kurzu se nacházejí sekce `sekce1` a `sekce2`.
   - Sekce `sekce2` je skrytá před studenty.
   - Sekce `sekce1` má popis odpovídající obsahu souboru `index.html` ve složce `sekce1` v repozitáři.
   - V sekci `sekce1` se nachází stránka `jina_stranka`, jejíž obsah odpovídá souboru `jina_stranka.txt` ve složce `sekce1` v repozitáři.
   - V sekci `sekce1` se nachází soubor `test2.pdf`.
   - Sekce `sekce2` nemá žádný popis.
   - V sekci `sekce2` se nachází složka `slozka`, která obsahuje soubor `test3.pdf`.

C.4 Test importu s automatickým sledováním změn

Název testu: Test importu s automatickým sledováním změn

Identifikátor testu: `GITLABI_UAT_UIMP01`

Účel testu: Otestovat kompletní funkcionalitu importu s automatickým sledováním změn.

Popis testu: Testovány jsou různé úpravy zdrojového repozitáře spolu se správným chováním možností manuální aktualizace a reimportu kurzu.

Vstupní podmínky:

- V aplikaci Moodle je testovaný plugin nainstalován a nakonfigurován.
- V aplikaci Moodle je vytvořený testovací kurz, ve kterém má uživatel provádějící test právo provádět změny. V tomto kurzu není nastaven import s automatickým sledováním změn.
- V nastavení pluginu GitLab integration je povolen import prázdných souborů.
- Do testovacího kurzu byl přidán blok GitLab integration (blok s uživatelským rozhraním testovaného pluginu).
- Uživatel provádějící test povolil v aplikaci Moodle pluginu GitLab integration zasílání notifikací.
- Uživatel provádějící test má v testovaném pluginu nastavený platný personal access token z aplikace GitLab.
- Testovací repozitář se nachází v aplikaci GitLab, odkud bude importován. Uživatel má v tomto repozitáři roli alespoň *Maintainer*.

Testovací data:

- Přihlašovací údaje do aplikace Moodle uživatele provádějícího test.
- Testovací repozitář s následující strukturou:

```
.
|-- sekce1
|   |-- index.html
|   |-- jina_stranka.txt
|   `-- test2.pdf
|-- sekce2
|   |-- slozka
|   |   `-- test3.pdf
|   `-- hidden.txt
|-- index.txt
|-- stranka.md
|-- test1.pdf
`-- ztest.pdf
```

Kroky testu:

1. Přihlaste se do aplikace Moodle a přejděte na testovací kurz.
   Očekávaný výsledek: Zobrazí se testovací kurz spolu s viditelným uživatelským rozhraním testovaného pluginu, ve kterém jsou možnosti *Import course content from GitLab*, *Edit your API key* a *Remove your API key*.
2. Klikněte na volbu *Import course content from GitLab*.

Očekávaný výsledek: Zobrazí se formulář pro zadání názvu projektu, ze kterého má být proveden import.

3. Zadejte správný název projektu a klikněte na tlačítko *Next*.
   Očekávaný výsledek: V případě, že zadanému jménu odpovídá více projektů, zobrazí se formulář pro výběr konkrétního projektu – v takovém případě vyberte správný projekt a klikněte na tlačítko *Next*. V opačném případě nebo po dokončení předchozího subkroku se zobrazí formulář pro výběr zdrojové větve.
4. Vyberte správnou zdrojovou větev a na stránce zaškrtněte checkbox *Automatically update course content when changes are pushed to the selected branch*. Následně klikněte na tlačítko *Import*.
   Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že import byl úspěšně spuštěn.
5. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.
   Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.
6. Přejděte do testovacího kurzu a zkontrolujte, že v uživatelském rozhraní pluginu jsou informace o importu s automatickým sledováním změn.
   Očekávaný výsledek: Blok GitLab integration vypadá následujícím způsobem:

   - Zobrazuje se hláška oznamující, že obsah kurzu je synchronizován s GitLab repozitářem.
   - Zobrazuje se úplný název repozitáře (včetně ID) a název větve, se kterou je obsah kurzu synchronizován.
   - Zobrazuje se hash posledního importovaného commitu. Tento hash souhlasí s hashem posledního commitu ve zdrojové větvi repozitáře, ze které byl obsah kurzu importován.
   - Jsou dostupné volby *Disable synchronization*, *Update course content now* a *Reimport course content*.

7. Zkontrolujte obsah testovacího kurzu.
   Očekávaný výsledek: V kurzu se nachází importovaný obsah odpovídající testovacímu repozitáři, konkrétně:

   - Hlavní sekce má popis odpovídající obsahu souboru `index.txt` v kořenovém adresáři repozitáře.
   - V hlavní sekci se nachází stránka `stranka`, jejíž obsah odpovídá souboru `stranka.md` v kořenovém adresáři repozitáře.
   - V hlavní sekci se nacházejí soubory `test1.pdf` a `ztest.pdf`. Tyto soubory jsou seřazeny v abecedním pořadí (tedy `test1.pdf` je před `ztest.pdf`).
   - V kurzu se kromě hlavní sekce nacházejí sekce `sekce1` a `sekce2`.
   - Sekce `sekce2` je skrytá před studenty.
   - Sekce `sekce1` má popis odpovídající obsahu souboru `index.html` ve složce `sekce1` v repozitáři.
   - V sekci `sekce1` se nachází stránka `jina_stranka`, jejíž obsah odpovídá souboru `jina_stranka.txt` ve složce `sekce1` v repozitáři.
   - V sekci `sekce1` se nachází soubor `test2.pdf`.
   - Sekce `sekce2` nemá žádný popis.
   - V sekci `sekce2` se nachází složka `slozka`, která obsahuje soubor `test3.pdf`.

8. Klikněte na volbu *Update course content now*.
   Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že aktualizační import byl spuštěn.

9. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.
   Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.
10. Zkontrolujte obsah testovacího kurzu.
    Očekávaný výsledek: Obsah kurzu se nezměnil (je stejný jako v kroku 7).
11. Změňte obsah repozitáře následujícím způsobem:

    - V kořenovém adresáři změňte obsah souboru `index.txt`.
    - V kořenovém adresáři změňte obsah souboru `test1.pdf`.
    - V kořenovém adresáři přejmenujte soubor `ztest.pdf` na `atest.pdf`.
    - Z adresáře `sekce2` odstraňte soubor `hidden.txt`.
    - V podadresáři `slozka` v adresáři `sekce2` změňte obsah souboru `test3.pdf`.
    - Zkopírujte adresář `sekce1` jako `sekce3`.
    - V adresáři `sekce1` vytvořte soubor `hidden.txt`.
    - V adresáři `sekce1` změňte obsah souboru `index.html`.
    - V adresáři `sekce1` změňte obsah souboru `jina_stranka.txt`.
    - V adresáři `sekce1` změňte obsah souboru `test2.pdf`.

    Následně proveďte commit změn do Git repozitáře a push tohoto commitu do zdrojové větve v GitLab repozitáři, ze které byl vytvořen import s automatickým sledováním změn. Vyčkejte na příchod notifikace oznamující začátek aktualizačního importu.
    Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o tom, že byl spuštěn aktualizační import spolu s názvem kurzu, kterého se import týká.
12. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.
    Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.
13. Přejděte na testovací kurz a zkontrolujte, že provedené změny byly importovány.
    Očekávaný výsledek: Provedené změny se projevily, konkrétně:

    - Popis hlavní sekce odpovídá změněnému souboru `index.txt` v kořenovém adresáři.
    - Obsah souboru `test1.pdf` v hlavní sekci odpovídá změněnému souboru `test1.pdf` v kořenovém adresáři.
    - V hlavní sekci se nenachází soubor `ztest.pdf`.
    - V hlavní sekci se nachází soubor `atest.pdf`, který je nad souborem `test1.pdf`. Jeho obsah odpovídá souboru `atest.pdf` v kořenovém adresáři repozitáře.
    - Sekce `sekce2` není skrytá před studenty.
    - Obsah souboru `test3.pdf` ve složce `slozka` v sekci `sekce2` odpovídá změněnému souboru `test3.pdf` v podadresáři `slozka` adresáře `sekce2`.
    - V kurzu se nachází sekce `sekce3`, s obsahem shodným jako sekce sekce1 v kroku 7.
    - Sekce v kurzu jsou zobrazeny v následujícím pořadí: Hlavní sekce, `sekce1`, `sekce2`, `sekce3`.
    - Sekce `sekce1` je skrytá před studenty.
    - Popis sekce `sekce1` odpovídá změněnému souboru `index.html` v adresáři `sekce1`.
    - Obsah stránky `jina_stranka` v sekci `sekce1` odpovídá změněnému souboru `jina_stranka.txt` v adresáři `sekce1`.
    - Obsah souboru `test2.pdf` v sekci `sekce1` odpovídá změněnému souboru `test2.pdf` adresáři `sekce1`.

14. Klikněte na volbu *Reimport course content*.
    Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že byl spuštěn reimport kurzu.
15. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.

Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.

16. Zkontrolujte obsah testovacího kurzu.
    Očekávaný výsledek: Obsah kurzu se nezměnil (je stejný jako v kroku 13).
17. Změňte obsah repozitáře následujícím způsobem:
    - V kořenovém adresáři odstraňte soubor `index.txt`.
    - V kořenovém adresáři odstraňte soubor `test1.pdf`.
    - V kořenovém adresáři změňte obsah souboru `stranka.md`.
    - Odstraňte adresář `sekce1`.
    - Přejměnujte adresář `sekce2` na `sekce4`.
    - V adresáři `sekce3` odstraňte soubor `index.html`.
    - V adresáři `sekce3` přejmenujte soubor `jina_stranka.txt` na `stranka.txt`.
    - V adresáři `sekce3` přejmenujte soubor `test2.pdf` na `test4.pdf`.

    Následně proveďte commit změn do Git repozitáře a push tohoto commitu do zdrojové větve v GitLab repozitáři, ze které byl vytvořen import s automatickým sledováním změn. Vyčkejte na příchod notifikace oznamující začátek aktualizačního importu.
    Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o tom, že byl spuštěn aktualizační import spolu s názvem kurzu, kterého se import týká.
18. Vyčkejte na příchod notifikace oznamující úspěšné dokončení importu.
    Očekávaný výsledek: Uživateli se zobrazí nová notifikace, ve které je oznámení o úspěšně dokončeném importu a název kurzu, do kterého byl import proveden.
19. Přejděte na testovací kurz a zkontrolujte, že provedené změny byly importovány.
    Očekávaný výsledek: Provedené změny se projevily, konkrétně:
    - Hlavní sekce je bez popisu.
    - V hlavní sekci se nenachází soubor `test1.pdf`.
    - Obsah stránky `stranka` v hlavní sekci odpovídá změněnému souboru `stranka.md` v kořenovém adresáři.
    - V kurzu je pouze hlavní sekce a sekce `sekce3` a `sekce4` v tomto pořadí.
    - Sekce `sekce3` je bez popisu.
    - V sekci `sekce3` je jediná stránka, a to `stranka`. Její obsah odpovídá souboru `stranka.txt` v adresáři `sekce3`.
    - V sekci `sekce3` se nachází jediný soubor `test4.pdf`. Jeho obsah odpovídá souboru `test4.pdf` v adresáři `sekce3`.
    - V sekci `sekce4` se nachází pouze složka `slozka`, která obsahuje soubor `test3.pdf`. Obsah tohoto souboru odpovídá souboru `test3.pdf` v podadresáři `slozka` v adresáři `sekce4`.
20. Klikněte na volbu *Disable synchronization*.
    Očekávaný výsledek: Zobrazí se formulář pro potvrzení zvolené akce.
21. Klikněte na tlačítko Yes.
    Očekávaný výsledek: Zobrazí se hláška oznamující uživateli, že synchronizace s GitLab repozitářem byla úspěšně ukončena a po chvíli dojde k přesměrování zpět na kurz. Po přesměrování uživatelské rozhraní testovaného pluginu neobsahuje žádné informace o nastaveném importu s automatickým sledováním změn, ani volby *Disable synchronization*, *Update course content now* a *Reimport course content*. Je však dostupná volba *Import course content from GitLab*.
22. Ve zdrojovém repozitáři proveďte libovolnou změnu.
    Očekávaný výsledek: Změna se nijak neprojeví v testovacím kurzu, uživateli není zaslána žádná notifikace.