ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
PROGRAM GEODÉZIE A KARTOGRAFIE
OBOR GEOMATIKA

DIPLOMOVÁ PRÁCE

NÁVRH WEBOVÉHO ADMINISTRÁTORSKÉHO ROZHRANÍ PRO PLATFORMU GIS.LAB

Vedoucí práce: Ing. Martin Landa, Ph.D.
Katedra geomatiky

červen 2019

Bc. Tereza KULOVANÁ

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Kulovaná Jméno: Tereza Osobní číslo: 440796

Zadávající katedra: Katedra geomatiky

Studijní program: Geodézie a kartografie

Studijní obor: Geomatika

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce: Návrh webového administrátorského rozhraní pro platformu GIS.lab

Název diplomové práce anglicky: GIS.lab Web Administration Console

Pokyny pro vypracování:
Cílem diplomové práce je návrh webového administrátorského rozhraní pro potřeby komplexní open source GISové platformy GIS.lab. Toto rozhraní by mělo rozšířit možnosti správy uživatelů této platformy, umožnit definovat jejich role a oprávnění. Webová aplikace bude navržena s ohledem na její maximální integraci do stávající architektury platformy GIS.lab.

Seznam doporučené literatury:
Pilgrim, M.: Dive Into Python, Createspace Independent, 2009, ISBN: 9781441413024
Holovaty, A., Kaplan-Moss, J.: The Definitive Guide to Django: Web Development Done Right, Second Edition, Apress, 2009, ISBN: 9781430219361
Turnbull, J.: The Docker Book: Containerization Is the New Virtualization, James Turnbull, 2014, ISBN: 9780988820203

Jméno vedoucího diplomové práce: Ing. Martin Landa, Ph.D.

Datum zadání diplomové práce: 22. 2. 2019 Termín odevzdání diplomové práce: 19. 5. 2019
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

Podpis vedoucího práce Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v diplomové práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce“ a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací“.

Datum převzetí zadání Podpis studenta(ky)

ABSTRAKT

Tato diplomová práce si klade za cíl vytvořit webové administrátorské rozhraní pro platformu GIS.lab. Rozhraní umožňuje správu uživatelů a definování jejich rolí a oprávnění. Je tvořeno ze dvou částí - uživatelské konzole a administrátorské konzole. Při tvorbě byl využit framework Django. Data jsou synchronizována mezi lokální databází Djanga a LDAP serverem. Na konec bude webové rozhraní integrováno do GIS.labu jako nová služba přes Docker kontejner.

KLÍČOVÁ SLOVA

GIS.lab, Python, Django, LDAP, Docker, administrace, open source

ABSTRACT

The aim of this diploma thesis is to create a web administration interface for GIS.lab platform. The interface enables user management and defining user roles and permissions. It consists of two parts - user console and admin console. Project uses Django framework. Data are synchronized between local Django database and LDAP server. Eventually, web console is going to be integrated to GIS.lab as a new service using Docker container.

KEYWORDS

GIS.lab, Python, Django, LDAP, Docker, administration, open source

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma „Návrh webového administrátorského rozhraní pro platformu GIS.lab“ jsem vypracovala samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji Ing. Martinovi Landovi, Ph.D. za cenné rady a vedení mé diplomové práce. Velké díky patří i mé rodině a příteli za projevenou podporu a trpělivost.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Open-source platforma GIS.lab slouží k rychlému a jednoduchému nasazení centrálně řízené GIS infrastruktury v lokální síti (LAN), v data centru nebo v cloudové službě. Poskytuje celistvý soubor neplaceného GIS softwaru integrovaného do jednoho systému, který je okamžitě připraven k použití.

Všechny použité technologie jsou plně pod kontrolou správců systému, náklady na nasazení a vlastnictví takovéhoto komplexního řešení jsou sníženy na absolutní minimum. Díky tomu je možné GIS.lab využít v oblastech a podmínkách, kde by aplikace proprietárních technologií nebyla cenově dostupná či technicky možná. Příkladem mohou být rozvojové země či sféra školství a vzdělávacích institucí.

GIS.lab je dostupný ve formě desktopového klienta. Vytváření webových aplikací je přístupné díky samostatné platformě Gisquick, která je automaticky integrovaná i do desktopové verze. GIS.lab Desktop obsahuje širokou škálu funkcionalit, z nichž mezi nejdůležitější patří ukládání prostorově i neprostorově orientovaných dat a jejich sdílení, tvorba a analýza vektorových, rastrových i tabulkových dat či rychlé vytváření kartografických výstupů. Data mohou být ukládána buď v souborovém systému, nebo v PostGIS databázi.

Vývoj GIS.labu ještě ani zdaleka není u konce. Nabízené portfolio se má do budoucna dělit na pět základních balíčků - datasety otevřených dat, přístup k PostGIS databázi, publikaci dat pomocí webových mapových služeb (WMS, WFS), webovou aplikaci a desktopového klienta (více viz kapitola 2.4). Všechny komponenty by měly být dostupné jak společně, tak i samostatně, nezávisle na desktopovém klientovi. Mezi dalšími rozšířeními by mohl být například výpočetní server na bázi WPS (Web Processing Service) a další.

Vývojáři GIS.labu se obecně snaží o co nejvíce uživatelsky přívětivé prostředí, což vedlo k rozhodnutí vytvořit webové administrátorské rozhraní pro snazší správu uživatelů a definování jejich přístupových práv ke zmiňovaným službám. Současný systém administrace přes příkazovou řádku není pro některé správce srozumitelný, navíc neumožňuje žádosti o registraci a o přiřazení přístupových práv přímo ze strany uživatelů. Pro naplnění tohoto požadavku vzniklo zadání této diplomová práce.

Webová konzole bude tvořena dvěma částmi - uživatelskou konzolí a administrátorskou konzolí, jež bude přístupná pouze správcům.

Při návrhu webové aplikace bude vyvinuta snaha o její maximální integraci do stávající architektury platformy GIS.lab. Pro vytvoření webového rozhraní bude zvolen framework Django, který využívá, dnes již od GIS.labu oddělená, platforma Gisquick. Důležitou vlastností tohoto frameworku je i to, že je psaný v jazyce Python, což umožní navázání na existující, ale nedokončenou knihovnu pro administraci uživatelů z roku 2015, která by měla nahradit současné shellové skripty. Pro ověření mezi webovou aplikací a LDAP serverem obsahujícím uživatelské informace bude využita některá z dostupných externích knihoven (např. django-python3-ldap, ldap3).

V teoretické části bude čtenář především podrobněji seznámen s platformou GIS.lab, jejím budoucím rozšířením a protokolem LDAP, který slouží k přístupu k datům, jejich úpravám a ukládání na adresářovém serveru.

1 Rešerše

Framework Django automaticky po instalaci obsahuje vestavěné administrační rozhraní. To slouží ke správě záznamů v lokální databázi. Django umožňuje využít připravené modely *User* a *Group* nebo si vytvářet vlastní. Bylo vyvinuto jako komplexní nástroj pro zobrazování a spravování článků, komentářů, uživatelů a autorů v novinářském prostředí. Administrátorské rozhraní pro správu uživatelů a jejich příslušnosti ke skupinám je jen malou podmnožinou toho, co všechno tento framework umí. Django nabízí obsáhlou dokumentaci [1], která usnadňuje jak orientaci v problematice, tak vlastní vývoj.

Prvotní inspirací pro webovou konzoli byla platforma Gisquick (viz 2.2), která využívá framework Django. Jedná se o webovou mapovou aplikaci, do níž se mohou uživatelé zaregistrovat a po přihlášení zveřejňovat a spravovat své projekty. Po oddělení od GIS.labu jsou uživatelské účty uloženy ve výchozí databázi SQLite. Verze, jež je integrovaná v GIS.labu, ale ověřuje přihlašovací údaje vůči LDAP serveru. Zvolené nastavení připojení k LDAP bylo použito v počáteční fázi vývoje webové konzole GIS.lab.

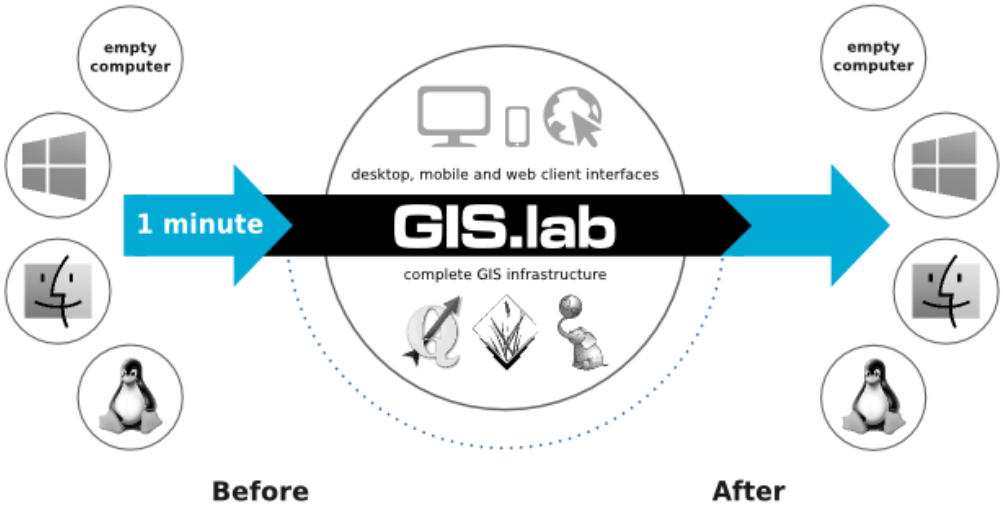
Dalšími zajímavými projekty, které využívají Django pro správu uživatelů je např. chatovací aplikace Zulip (https://zulipchat.com) nebo platforma pro online prodej Oscar (http://oscarcommerce.com).

2 GIS.lab

Obrázek 2.1: GIS.lab logo (zdroj: GIS.lab repozitář)

2.1 Co je to GIS.lab?

GIS.lab je nástroj pro jednoduché a rychlé nasazení (deployment) funkční, centrálně spravované GIS infrastruktury v jednotném prostředí lokální sítě (LAN), v data centru nebo v cloudové službě. Jedná se o technologii, která poskytuje komplexní soubor otevřených GISových softwarových nástrojů integrovaných do jednoho celistvého systému, jenž vyžaduje minimální náklady na pořízení a údržbu.



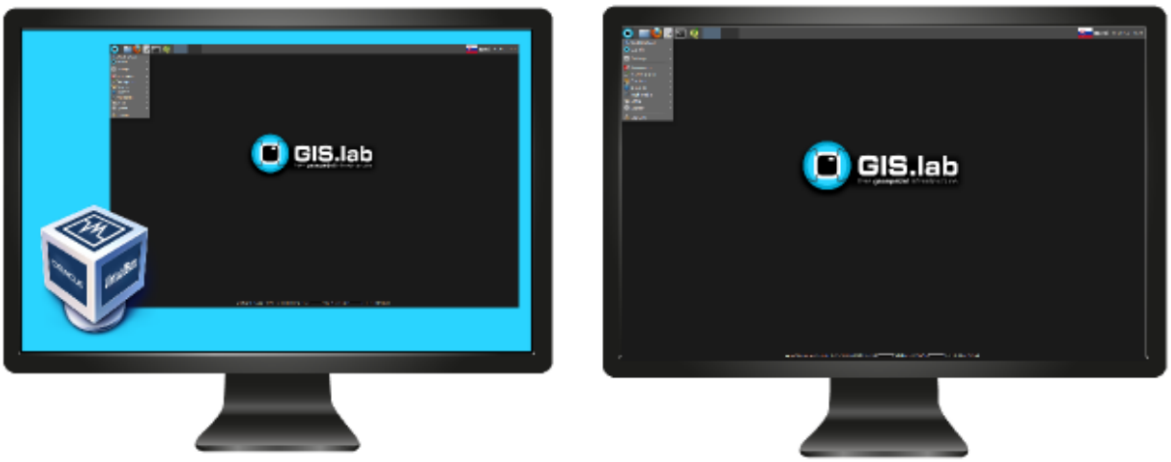
Obrázek 2.2: Schéma jednoduchosti nasazení GIS.lab (zdroj: GIS.lab repozitář)

Svět softwaru založeného na myšlence open-source sestává z velkého množství různorodých osobností a skupin, z nichž každá svůj projekt uchopuje po svém. Propojit tyto různorodé nástroje mezi sebou do funkčního celku je náročný úkol a právě ten se pokouší řešit platforma GIS.lab.

K největším výhodám této platformy patří maximálně automatizovaná instalace či rychlé nasazení pomocí GIS.lab Unit (viz 2.1), jejichž výsledkem je plně funkční a vysoce výkonný nástroj, bez nutnosti dalšího složitého nastavení. Přizpůsobení specifickým potřebám zákazníka je však také možné.[1] Všichni klienti GIS.labu Desktop, uživatelské účty i zálohy jsou centrálně spravované.

Z hlediska GIS infrastruktury a její komplexnosti jsou nejzásadnější ukládání prostorově i neprostorově orientovaných dat (v souborovém systému nebo v PostGIS databázi) a jejich sdílení, tvorba a analýza vektorových, rastrových i tabulkových dat nebo rychlé vytváření kartografických výstupů.

Desktopový klient (tlustý klient) GIS.lab Desktop může být spuštěn v režimu fyzickém či virtuálním. Virtuální režim lze využít na kterémkoliv operačním systému (OS) s tím, že původní OS i GIS.lab jsou přístupné. Fyzický režim umožňuje lepší výkon, který je vykoupen dočasnou nedostupností původního OS. [2]



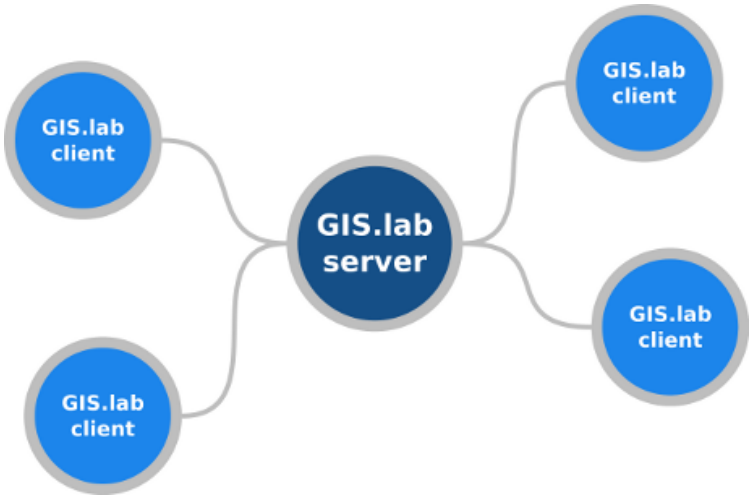
Obrázek 2.3: Virtuální a fyzický režim GIS.lab Desktop (zdroj: GIS.lab repozitář)

Tlustý klient poskytuje desktopové prostředí bez zádrhelů, které by se mohly vyskytovat u odlehčené webové verze. Jeho využití však není primárně zamýšleno v tradičním pojetí desktopu jako jediného klienta, ale spíše jako jakési specializované klientské rozhraní poskytující nástroje ze světa desktopu.

[1] Úpravy pro skupinu GISMentors: https://github.com/GISMentors/gislab-customization

Konfiguraci a nasazení GIS.labu řídí platforma Ansible (viz 3.5). Deployment ve virtuálním režimu umožňuje Vagrant[2] a VirtualBox[3].

GIS.lab se po nasazení sestává z jednoho stroje, který zastává roli hlavního uzlu (server, master), a k němu připojeného množství dalších počítačů (klientů, nodes). Výsledkem je lokální počítačová síť, v podstatě jakýsi cluster. Pro hlavní uzel je vyžadován stroj, který běží na operačním systému Ubuntu či se na něj dá doinstalovat, požadavky na klientské počítače nejsou téměř žádné - nemusí obsahovat operační systém ani pevný disk. Je však potřeba, aby všechny počítače v síti byly připojené pomocí gigabitového síťového kabelu a síťového přepínače (switch). Výpočetní kapacitu je možné sdílet přes všechny stroje. [2]



Obrázek 2.4: GIS.lab architektura (zdroj: GIS.lab repozitář)

GIS.lab Server běží na operačním systému Ubuntu Linux LTS s odlehčeným desktopovým prostředím XFCE. [3] K ověření a správě uživatelů využívá protokol LDAP, přesněji jeho nadstavbu OpenLDAP (viz 3.1.1).

Vedle základního softwarového vybavení standardního OS jsou dostupné dva hlavní GISové programy QGIS a GRASS GIS. [3] QGIS je vhodný pro tvorbu projektů, přípravu a analýzu dat. Umožňuje i jejich následnou publikaci nejen v podobě WMS/WFS služeb, ale i jako webových aplikací pomocí předinstalovaného zásuvného modulu Gisquick. GRASS GIS lze využít pro komplexní datové analýzy.

[2]open-source nástroj pro vytváření a údržbu přesnosného vývojového prostředí pomocí virtualizace

[3]Oracle VM VirtualBox, multiplatformní virtualizační nástroj

Data jsou uložena v souborovém systému či geodatabázi PostGIS. Dostupná je i nadstavba SpatiaLite pro databázi SQLite umožňující ukládání geoprostorových dat. [3]

GIS.lab obsahuje velké množství knihoven a balíčků Python, z nichž zde jsou zmíněni jen vybraní zástupci. GDAL je knihovna určená pro zápis a čtení vektorových a rastrových geodat. Proj.4 slouží k transformaci geoprostorových souřadnic z jednoho souřadnicového systému do dalšího. Pro práci se satelitními daty družic MODIS a Sentinel slouží PyModis a Sentinelsat.

Přizpůsobení základní instalace specifickým požadavkům může být provedeno buď standardními linuxovými příkazy, nebo upravením Ansible Playbooks, které se vyznačují pro člověka snadno čitelným jazykem. Chování systému během vytváření a odstraňování uživatelských účtů je definováno v pěti speciálních skriptech.

- `before-add` - spuštěn před vytvořením účtu
- `after-add` - spuštěn po vytvoření účtu
- `before-delete` - spuštěn před odstraněním účtu
- `after-delete` - spuštěn po odstranění účtu
- `files` - obsah tohoto adresáře je zkopírován do domovského adresáře uživatele před tím, než je spuštěn skript `after-add`

Tyto skripty je také možné upravovat konkrétním potřebám. Příkladem může být nakládání s databází. Ve skriptu `after-add` definujeme, zda uživateli bude vytvořena nová databáze nebo jen přidáno schéma do již existující společné. Ve skriptu `before-delete` zvolíme, jestli před odstraněním účtu dojde k vyprodukování tzv. dumpu, který bude uživateli zaslán na jeho adresu. A nakonec při spuštění `after-delete` bude tato databáze/schéma odstraněna, resp. odstraněno.

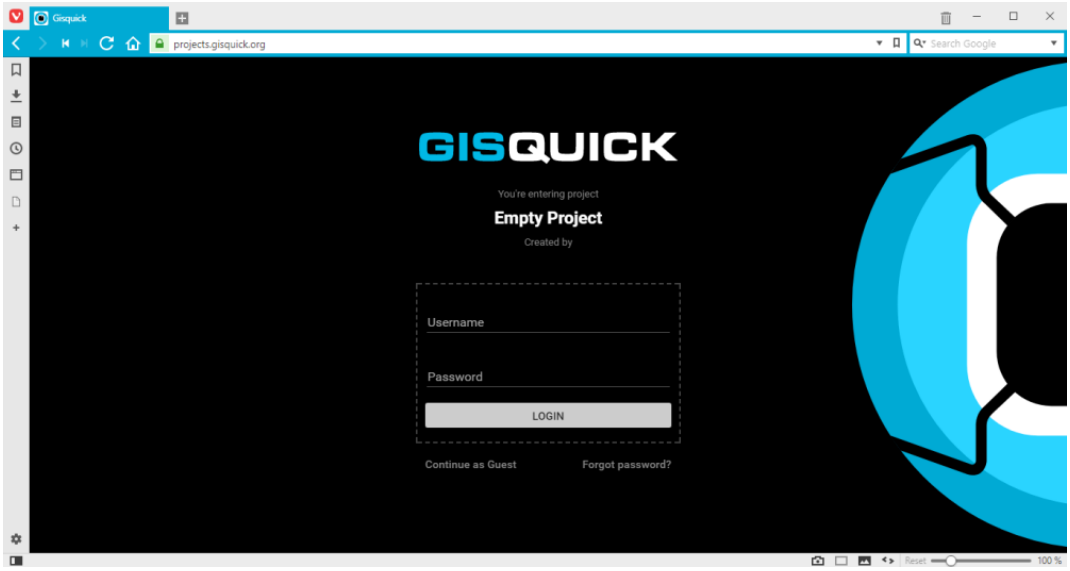
GIS.lab Unit

Zařízení s názvem GIS.lab Unit je přenosné hardwarové řešení obsahující systém GIS.lab připravený k okamžitému zapojení a nasazení ve zvolené síti. Jedná se o krabičku s rozměry přibližně 11 x 11 x 4 cm, procesorem Intel Haswell, SSD diskem a 16 GB RAM. [3] Maximální testované množství klientských počítačů je 20, v rámci výuky na Fakultě stavební ČVUT.

Obrázek 2.5: GIS.lab Unit (zdroj: GISMentors)

S pomocí integrované platformy Gisquick (viz 2.2) podporuje GIS.lab kromě desktopového klienta i webovou publikační platformu.

2.2 Gisquick



Obrázek 2.6: Gisquick - přihlašovací stránka (zdroj: Tereza Kulovaná)

Gisquick je open-source platforma umožňující publikaci geoprostorových dat. Byla vytvořena s cílem snadného sdílení projektů vytvořených v desktopové aplikaci QGIS

na webu. Gisquick sestává ze zásuvného modulu QGIS, QGIS serveru, serverové aplikace založené na frameworku Django a webového klienta. Obsahuje základní sadu nástrojů potřebných pro webovou mapovou aplikaci s plně responzivním uživatelským rozhraním (GUI). [4]

Gisquick byl vyvíjen jako součást GIS.labu, ale v roce 2015 se oddělil jako samostatný projekt. Dle původní představy autorů měl mít Gisquick podobu tenkého klienta GIS.labu, tedy jakési odlehčené verze. Měl nabízet nástroje pro interakci s daty - editaci, provádění výpočtů a analýz. Aktuálně však umožňuje pouze jejich zobrazování.

Dnes je možné ho využívat samostatně, zároveň však zůstává integrován v každé instalaci GIS.labu a rozšiřuje jeho funkcionalitu. Je distribuován pod otevřenou licencí GNU GPL v2.0. [5]

2.3 Aktuální využití

V současné době se GIS.lab využívá především při výuce na Fakultě stavební ČVUT v rámci hodin zaměřených na GIS a dálkový průzkum Země (DPZ) a na Přírodovědecké fakultě Univerzity Karlovy. Také na něm probíhají veřejná školení skupiny GISMentors.

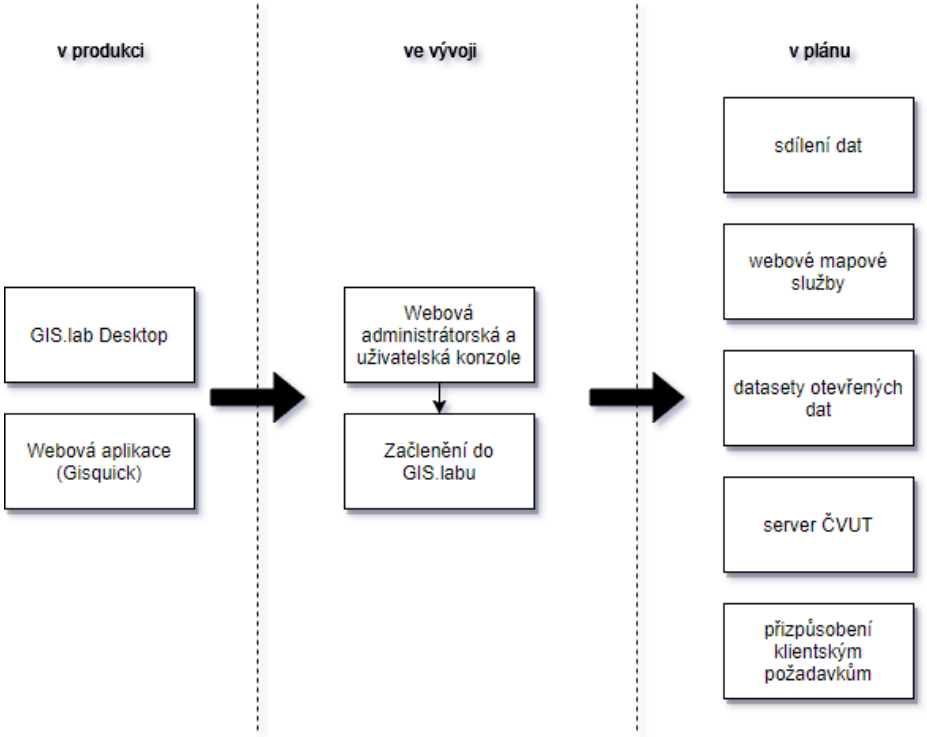
Obrázek 2.7: Počítačová učebna Fakulty stavební ČVUT (zdroj: GISMentors)

Vyučování na stavební fakultě probíhá ve fyzickém režimu GIS.lab Desktop s nasazením pomocí GIS.lab Unit. Každý student má vytvořen vlastní uživatelský účet a dedikované schéma v databázi. Studenti jsou seznámeni s připojením k databázi pomocí PgAdmin, většinou je však využíváno připojení přes QGIS. QGIS je také upřednostňovaným softwarem při zpracování prostorových i neprostorových dat, výhodou je i možnost využívat zásuvný modul Gisquick pro publikaci dat v podobě webových aplikací. Studenti jsou krátce seznámeni s programem GRASS GIS, strukturou GRASS projektů a práce s nimi. GRASS GIS je více využíván v rámci předmětů orientovaných na DPZ, jelikož obsahuje dobře implementované nástroje pro zpracování obrazových dat. K nástrojům GRASS GIS mohou uživatelé přistupovat i přes QGIS UI díky předinstalovanému zásuvnému modulu QGIS GRASS.

Komunikace probíhá přes zabudovanou IRC službu, kterou lze využít mimo jiné pro okamžité a jednoduché sdílení nezbytných příkazů či potřebných webových adres. Pro sdílení souborů jsou určené dvě složky, k nimž mají přístup všichni připojení uživatelé. Pro standardní výměnu mezi klientskými počítači je příhodnější složka *Barrel* s právy čtení i zápisu pro všechny. Data trvalejšího charakteru je vhodné umístit do adresáře *Repository*, odkud je možné data stahovat, ale upravovat je mohou pouze správci.

2.4 Plány na rozšíření

Autoři GIS.labu by rádi v budoucnu rozšířili uživatelskou základnu, proto mají za cíl práci správcům i uživatelům zpříjemnit a nabídnout co nejširší portfolio různých balíčků (tzv. services).



Obrázek 2.8: Vývoj GIS.labu (zdroj: Tereza Kulovaná)

Webová administrátorská a uživatelská konzole

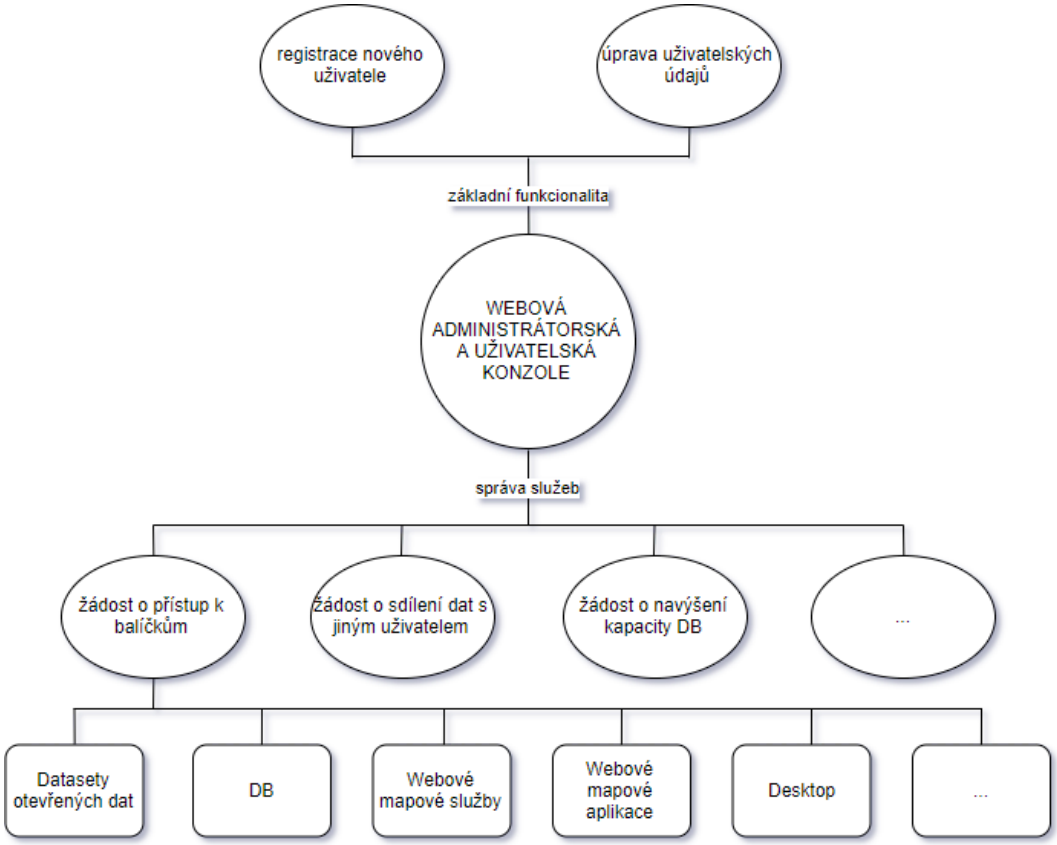
Nejblíže zařazení na seznam služeb je webové rozhraní pro správu uživatelů, které je zpracováváno v rámci této diplomové práce.

Aktuálně používaný systém správy uživatelů funguje na bázi příkazové řádky a je popsán v kapitole 4.1. Tento systém není pro všechny úplně intuitivní, zároveň neumožňuje přístup ze strany uživatelů. Proto je cílem vytvořit webové grafické uživatelské rozhraní (GUI), které bude mít dvě části - administrátorskou a uživatelskou.

Uživatelská konzole bude umožňovat registraci, přístup k osobním údajům a jejich editaci a v neposlední řadě žádosti o zpřístupnění jednotlivých balíčků služeb. Balíčky by v první fázi měly sestávat z pěti složek: datasetů otevřených dat; přístupu k datům (souborový systém, DB); publikace dat pomocí webových mapových služeb; webové aplikace; desktopového klienta. Přístup k prvním třem komponentám by měl být možný i mimo desktopového klienta, což je další důvod pro vytvoření webového rozhraní, ke kterému lze přistupovat z jakéhokoliv počítače. Přes konzoli bude probíhat i sdílení dat s dalšími uživateli či žádost o navýšení kapacity

databáze. Kromě úpravy uživatelských údajů budou všechny změny vyžadovat potvrzení správce.

Administrátorská konzole bude mít v podstatě stejnou funkcionalitu jako uživatelská s tím rozdílem, že administrátor bude moci spravovat všechny uživatele.



Obrázek 2.9: Budoucí podoba webové konzole (zdroj: Tereza Kulovaná)

Sdílení dat

V případě fyzického režimu GIS.lab Desktop jsou veškerá vytvořená data dostupná pouze v rámci klienta, na němž GIS.lab běží, sdílet s dalšími uživateli je lze pouze přes společné adresáře *Repository* a *Barrel*. Data jsou tímto způsobem přístupná všem uživatelům v rámci sítě bez rozdílu.

Pokud by se chtěl uživatel podělit o přístup k některé části své databáze, bude k tomu moci v první fázi využít SQL příkazu GRANT, později pak vyhledat jiného uživatele přes webovou administrátorskou konzoli a přístup mu udělit přes ni.

Virtuální režim GIS.lab Desktop běží nad VirtualBoxem, který umožňuje připojení složek z hostitelského OS. Pokud má uživatel zájem data přenést z fyzického klienta pryč nebo naopak nějaká data nahrát, musí k tomu využít mezistupeň v podobě externího disku, ať už reálného či virtuálního. Ideální by proto byla možnost přistupovat vzdáleně nejen k databázi, ale například i k oddílu na serveru obsahujícímu data ve formátu Esri Shapefile (.shp) či OGC GeoPackage (.gpkg). Tuto variantu bude umožňovat sdílení pomocí služby NFS (Network File System). Tímto způsobem budou moci uživatelé pracovat s daty i při běhu svého standardního OS.

Desktopový klient již NFS využívá pro připojení některých adresářů (Home, Barell, Repository) ze serveru. Cílem tedy je povolit sdílení disku i mimo desktop klienta.

Webové mapové služby

Pro publikaci dat je nyní aktuálně nutné být přihlášen na desktopovém klientovi GIS.lab, zde vytvořit projekt a nakopírovat jej do adresáře *Publish*. Poté je přístupný jako webová služba dle standardu OGC (Open Geospatial Consortium). Cílem je umožnit uživateli vytvořit si projekt ve svém vlastním prostředí a následně ho jen nahrát na server, který by dovolil jeho sdílení. Připojení k serveru za tímto účelem bude umožňovat výše zmiňovaná služba NFS.

Další variantu bude pro uživatele nabízet webová uživatelská konzole, přes níž bude možné projekt na server nahrát a konzole zpátky klientovi vygeneruje správnou cestu k webové mapové službě. Tento přístup bude potřeba zvláště v případě, že uživatel bude mít přístup pouze k balíčku webových mapových služeb.

Datasety otevřených dat

Část dat už je dnes v rámci státní správy České republiky poskytovaná ve formě otevřených dat. [6] Ne všichni uživatelé, kteří by je mohli ke své práci využít, však vědí, kde zmiňovaná data najít, případně jak je dostat do formátu vhodného k dalším analýzám. Právě pro ně bude vhodný další balíček.

Pokud si to správce zvolí, GIS.lab server bude obsahovat stažená geoprostorová data, především z Českého úřadu zeměměřického a katastrálního (ČÚZK) - územní jednotky, Registr územní identifikace, adres a nemovitostí (RÚIAN), apod. Na datové sady budou po stažení aplikovány testy datové integrity, případné nekonzistence

budou odstraněny a výsledek bude začleněn do PostGIS databáze. Přístup k těmto datasetům bude možný i nezávisle na jakémkoli dalším balíčku.

Konkrétní datové sady a tempo jejich aktualizací budou rozhodnuty až při reálné implementaci.

Server ČVUT

Snaha o rozšíření základny uživatelů platformy GIS.lab cílí v nejbližší době především na studenty a zaměstnance ČVUT a to i mimo Fakultu stavební. V prvním kroku již byly získány prostředky, jež umožňují spuštění GIS.labu přes server ČVUT. Pro jednotlivé fakulty či katedry, které by se jej rozhodly využívat, bude připraveno několik základních přizpůsobení, hlavně s ohledem na datasety z veřejného sektoru.

Přizpůsobení klientským požadavkům

Poslední v blízké době plánovanou změnou je umožnit zákazníkům přizpůsobit platformu GIS.lab konkrétním požadavkům již při nasazení pomocí Ansible Playbooks (viz 3.5). Aktuálně jsou využívány pro plně automatizovanou instalaci základní podoby GIS.labu a úpravy pro jednotlivá školení skupiny GISMentors. Podobným způsobem jako pro GISMentors se bude uzpůsobovat prostředí i pro další skupiny, např. fakulty ČVUT.

Ukázky nejvyužívanějších nastavení budou dostupné v Github repozitáři[4].

[4]https://github.com/gislab-npo/gislab/

3 Použité technologie

Třetí kapitola stručně představuje jednotlivé technologie použité při tvorbě webového administrátorského rozhraní.

3.1 LDAP

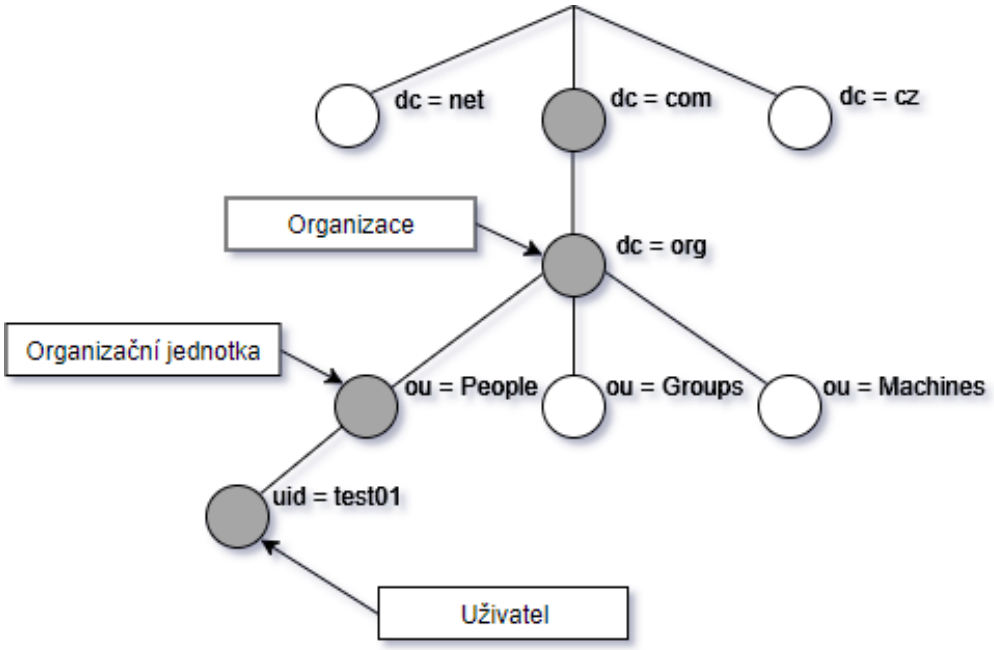
Lightweight Directory Access Protocol neboli LDAP je otevřený, standardizovaný protokol pro přístup k adresářovým službám. Slouží k zobrazování dat, jejich úpravám a ukládání na adresářovém serveru přes Internet Protocol (IP). [7]

Adresářová služba (directory service) je v podstatě specializovaná databáze, která slouží především k procházení a vyhledávání, ke změnám dochází jen zřídka. Obvykle neposkytuje komplikovanější databázové techniky, jako jsou transakce a operace nutné pro zachování datové integrity. Pro zvýšení rychlosti odezvy mohou dokonce obsahovat i duplicitní záznamy.

LDAP je založen na modelu server-klient. Jeden či více LDAP serverů obsahují data ve stromové struktuře (directory information tree, DIT). Klient požádá o záznam odpovídající jeho dotazu. Odpověď dostane buď ze serveru, ke kterému je připojen, nebo je odkázán na jiný server, kde je informace uložena. Výsledek hledání bude vždy stejný, ať už se připojí ke kterémukoliv serveru. Tento rys je důležitý obzvláště u globálních adresářových služeb. [8]

LDAP model se skládá z jednotlivých záznamů (entries). Záznam sestává z kolekce atributů (povinných a nepovinných) a má přiřazen globálně unikátní identifikátor, tzv. Distinguished Name (DN), který je založen na lokaci záznamu v rámci hierarchického (stromového) uspořádání modelu. DN začíná vlastním jménem záznamu (Relative Distinguished Name) a k němu jsou jako řetěz připojena jména všech jeho předků. Každý atribut má vlastní datový typ a může obsahovat jednu či více hodnot. [7]

Podle uvedené ukázky stromové struktury by jednoznačným pojmenováním pro uživatele se jménem uid=test01 bylo DN uid=test01, ou=People, dc=org, dc=com.



Obrázek 3.1: Příklad stromové struktury LDAP (zdroj: Tereza Kulovaná)

LDAP poskytuje ochranu informací, které jsou na serveru uloženy, pomocí mechanismu autentizace, který požaduje po klientovi prokázaní a ověření identity před zobrazením jakýchkoliv dat.

Server LDAP je vhodný pro autentizaci uživatelů a klientských zařízení, pro správu uživatelských informací, apod. Výhodou je, že uživatelé potřebují pro přístup k různým aplikacím v síti (pošta, ftp) pouze jedny přihlašovací údaje. Je nezávislý na operačním systému.

Protokol LDAP vychází ze standardu X.500, jehož je odlehčenou variantou. Někdy je nazýván X.500-lite.[5]

[5]https://www.webopedia.com/TERM/L/LDAP.html

3.1.1 OpenLDAP

Obrázek 3.2: OpenLDAP logo (zdroj: OpenLDAP.org)

OpenLDAP je otevřená (open-source) implementace protokolu LDAP vyvíjená pod hlavičkou OpenLDAP Project. OpenLDAP je distribuován pod vlastní licencí OpenLDAP Public License.[6] Vývoj OpenLDAP má počátek v roce 1998 a navazuje na předešlou činnost University of Michigan. [8]

Hlavní komponenty:

- slapd - nezávislý LDAP server
- knihovny implementující LDAP protokol
- klientské nástroje (ldapsearch, ldapadd, ldapmodify,...)

slapd (Standalone LDAP daemon) je nezávislý LDAP adresářový server, který zachytává LDAPpřipojení a odpovídá na operace, které přes tato spojení obdrží. Umožňuje připojení ke globální LDAP adresářové službě nebo může lokální služby obsluhovat sám. Může běžet na mnoha různých platformách.

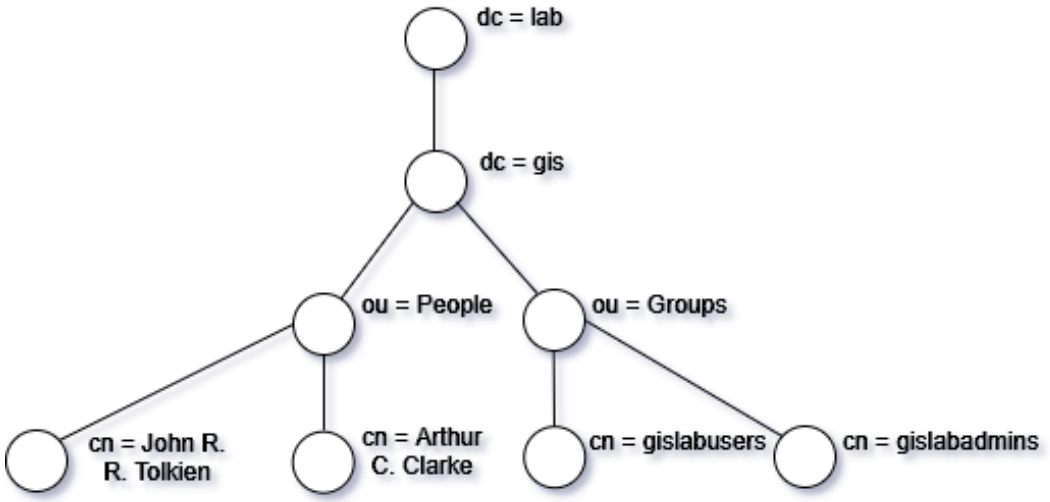
Podporuje silnou autentizaci a bezpečnost dat díky metodě SASL (Simple Authentication and Security Layer) a protokolu TLS (Transport Layer Security). Poskytuje široké možnosti kontroly přístupu k informacím v databázi - přístup může být udělen mj. na základě přihlašovacích údajů, IP adresy nebo DN. *slapd* nabízí celou škálu databázových backendů - MDB, hierarchický, vysoce výkonný transakční databázový backend; SHELL, backend pro shellové skripty; a jednoduchý backend PASSWD a jiné. [8]

Konfigurace *slapd* probíhá přes konfigurační soubor, příklad takové změny je uveden v kapitole 4.2.

GIS.lab využívá OpenLDAP 2.4. V rámci webové aplikace probíhá synchronizace s Djangem, z adresářové struktury OpenLDAP jsou konkrétně využívány organizační jednotky *People* a *Groups*. Náhled struktury GIS.labu je v ukázce omezen jen

[6] http://www.openldap.org/software/release/license.html

na záznamy potřebné k vytvoření DN těchto organizačních jednotek, které v realitě obsahují mnohem více záznamů.



Obrázek 3.3: Ukázka stromové struktury GIS.labu (zdroj: Tereza Kulovaná)

3.1.2 django-python3-ldap

Knihovna *django-python3-ldap* zajišťuje autentizaci uživatelů s LDAP serverem a synchronizaci LDAP uživatelů s lokální databází Django. Podporuje vlastní modely Django upravené specifickým potřebám. Ve výchozím nastavení je nakonfigurována podpora přihlášení přes OpenLDAP, po úpravě nastavení umožňuje připojení k adresářové službě Microsoft Active Directory. Vychází z knihovny *ldap3* a funguje jak pro Python 2, tak pro Python 3. [9]

Při prvním přihlášení uživatele se aplikace pokusí vytvořit připojení k LDAP serveru skrze poskytnuté uživatelské jméno a heslo. V případě úspěšného připojení jsou údaje z LDAP serveru uloženy do databáze Django a při každém dalším přihlášení jsou aktualizovány.

Synchronizaci všech uživatelů zároveň zajišťuje příkaz `manage.py ldap_sync_users`. Tuto akci lze provést jednorázově, pro pravidelnou synchronizaci může být automatizovaně spouštěn na pozadí skrze softwarového démona Cron. [9]

Použití této knihovny je relativně snadné. Po instalaci stačí v konfiguračním souboru Django *settings.py* nastavit potřebné proměnné (např. URL adresu LDAP serveru) a při příštím spuštění serveru již synchronizace funguje.

3.1.3 ldap3

Knihovna *ldap3* je založena na aplikaci protokolu LDAP v3. Poskytuje operace potřebné pro připojení k LDAP serveru, pro vyhledání záznamů, jejich vytvoření, úpravu a odstranění. Podporuje verze Python 2 i 3. [10]

3.2 Python

Obrázek 3.4: Python logo (zdroj: Python.org)

„*Python je vysokoúrovňový, interpretovaný programovací jazyk. Podporuje procedurálně i objektově orientované programování, je výkonný, zároveň má velmi jednoduchou a čistou syntax. V ostatních jazycích je odsazování řádků doporučeno z hlediska přehlednosti, u Pythonu je základním stavebním kamenem a je povinné.* [11]

Dnes je Python vyvíjen jako open source projekt pod záštitou neziskové organizace Python Software Foundation (PSF). Je distribuován pod licencí PSF, která je kompatibilní s GPL. Je možné ho nainstalovat na běžné platformy jako Windows, Unix nebo Mac OS, pro Linux je většinou součástí základní instalace. Při vyvarování se systémově závislých funkcí je přenositelný mezi platformami bez jakýchkoli změn.

Python má široké využití, od jednoduchých programů po rozsáhlé aplikace. Právě pro tyto možnosti, univerzálnost, přehlednost kódu a výkonnost z něj udělaly programovací jazyk, který je mezi začátečníky ve velké oblibě. Během krátké doby v něm funkční skript zvládne napsat každý.“ [12]

Python v současnosti existuje ve dvou hlavních verzích - Python 2 a Python 3. Python 3.0 byl vydán v roce 2008[7] a není zpětně kompatibilní s verzí Python 2. Hlavním důvodem pro takto zásadní změnu bylo rozhodnutí Guido van Rossuma, zakladatele jazyku Python, očistit Python 2.x od mnoha problémů pořádně v jednom kroku. [13]

Největšími změnami jsou upravení některých tříd a oddělení abstrakcí řetězec a posloupnost bytů. Textový řetězec (str) je nově převeden ve výchozím nastavení

[7] https://www.python.org/downloads/

na typ unicode, což vytváří konzistentnější a spolehlivější prostředí. Změny doznalo chování operátoru dělení / celých čísel. Výsledkem je číslo s plovoucí desetinnou čárkou, na rozdíl od předchozí verze, která vracela celé číslo (ve verzi 3.x dostupné pod operátorem //). Mezi nejviditelnější novinky pro běžného uživatele je přechod od příkazu print k funkci print(). [14]

Python 3 je obecně přívětivější k učení nových uživatelů a je považován za budoucnost tohoto jazyka. Stále však existuje mnoho programů, jež fungují na poslední verzi Python 2.7 vydané v roce 2010[8], která již nedostává žádné velké aktualizace. Důvody jsou různé - daný projekt byl vyvíjen v Pythonu 2 a nejsou dostatečné kapacity na jeho přechod na novější verzi; na některých operačních systémech není Python 3 nainstalovaný a uživatelé nemají vždy práva si ho sami doinstalovat; existuje potřeba využívat externí knihovnu, která podporuje pouze Python 2 a není triviální ji převést do Pythonu 3.

Poslední vydanou stabilní verzí je Python 3.7.[9]

3.3 Django

Obrázek 3.5: Django logo (zdroj: Djangoproject.com)

Django je vysokoúrovňový webový framework napsaný v jazyce Python. Je udržovaný organizací Django Software Foundation (DSF). Djagno je bezplatné a vydané pod open-source licencí BSD. Název získalo po jazzovém kytaristovi Djangovi Reinhardtovi. [15]

Hlavním cílem Djanga je usnadnit tvorbu komplexních, databází řízených webových aplikací. Pro tento účel se řídí zásadou oddělení zodpovědností (angl. Separation of concerns) a volně navazuje na architekturu Model-view-controller (MVC). [15]

[8] https://www.python.org/downloads/
[9] https://www.python.org, květen 2019

MVC architektura sestává ze tří volně propojených komponent:

- model (model) - reprezentace dat, k nimž aplikace přistupuje
- view (pohled) - uživatelské rozhraní
- controller (řadič) - reakce na žádosti a zajištění změn v pohledu nebo v modelu

Poslední vydanou stabilní verzí v době zpracování bylo Django 2.1 a veškerý další popis uvedený níže platí pro tuto verzi.

Frameworky mají snahu co největší množství práce automatizovat a tak při vytvoření projektu pomocí příkazu:

```
django-admin startproject nazev_projektu
```

vznikne automaticky jeho základní kostra, která slouží jako podstata již fungující Django aplikace. [1]

```
project
├── mysite
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── db.sqlite3
└── manage.py
```

__init__.py

Soubor *__init__.py* dává Pythonu najevo, že s adresářem, v němž se soubor nachází, má být zacházeno jako s balíčkem modulů Pythonu. Jedná se o prázdný soubor, který se obvykle nijak nemění.

settings.py

Soubor *settings.py* skrývá nastavení Django projektu, např. jakým způsobem má probíhat autentizace, kde jsou umístěné další potřebné soubory či informace o použitých databázích a registrovaných aplikacích.

urls.py

urls.py obsahuje URL cesty pro vytvořený Django projekt, v podstatě se jedná o jakýsi rejstřík stránky. Implicitně obsahuje cestu k vestavěné administrátorské konzoli. Při propojení s Django aplikacemi jsou sem přidány další URL adresy.

wsgi.py

WSGI je specifikace popisující komunikaci mezi webovým serverem a webovou aplikací nebo frameworkem v jazyce Python. Jedná se o primární nástroj nasazení programů v Djangu. Soubor *wsgi.py* se v základu skládá z jednoduché WSGI konfigurace, již je možno podle potřeby dále upravovat.

manage.py

Nástroj pro příkazový řádek *manage.py* umožňuje spravovat Django projekt. Tento soubor není po vytvoření nijak upravován.

Pro vývoj aplikace lze použít odlehčený webový server Djanga, na němž může autor okamžitě začít budovat aplikaci, aniž by byla vyžadována konfigurace produkčního serveru. Vývojový server provádí kontrolu kódu a automaticky se po každé uložené změně znovu načte, bez nutnosti restartu. [1] Tento server se spouští příkazem:

```
python manage.py runserver 0:8080
```

Standardně se vývojový server spouští na interní IP adrese a portu 8000. V případě, že je třeba zobrazovat webové stránky mimo stroj, na němž server běží, lze nastavit viditelnost a odlišný port serveru přidáním parametru `0:8080`. V takové situaci je stránka dostupná odkudkoliv při zadání adresy nastavené v proměnné `ALLOWED_HOSTS` v souboru *settings.py* a zvoleného portu.

Pro práci s databází jsou nezbytné dva základní příkazy:

```
python manage.py makemigrations
```

který vytváří jednotlivé migrační soubory založené na změnách provedených v modelech Djanga a

```
python manage.py migrate
```

který tyto změny aplikuje do databáze. V případě, že databáze ještě neexistuje, tak ji automaticky vytvoří. [1]

Migrace v Djangu funguje podobně jako verzovací systémy - `makemigrations` odpovídá příkazu `commit` a `migrate` pak obdobně jako `push` tyto změny propíše do databáze.

Poslední z významných funkcí *manage.py* je spouštění jednotkových testů.

db.sqlite3

Výchozí databází v Djangu je relační databáze SQLite. [1] Obsahuje informace o jednotlivých modelech a vztazích mezi nimi.

SQLite nemá vhodně implementovánu podporu změn, Django ji proto nahrazuje postupem, kdy vytvoří novou tabulku s novým schématem, data z původní tabulky překopíruje do nové, starou smaže a novou přejmenuje podle prvotní. Proto se nedoporučuje tuto databázi používat v produkci, obzvlášť v případě většího množství dat a častých změn.

Kromě SQLite Django oficiálně podporuje databáze PostgreSQL, MySQL a Oracle. [1]

3.3.1 Webové aplikace

Do projektu lze přidávat webové aplikace, jedna aplikace může být součástí více projektů. Pro její vytvoření lze užít konzolový nástroj:

```
python manage.py startapp nazev_aplikace
```

Implicitní struktura aplikace je vždy stejná:

- `__init__.py` - určuje aplikaci jako balíček Pythonu
- `admin.py` - umožňuje registraci datových modelů
- `apps.py` - definuje název aplikace
- `models.py` - umožňuje tvorbu datových modelů
- `tests.py` - umožňuje tvorbu jednotkových testů
- `views.py` - umožňuje definování pohledových funkcí

Každou aplikaci, která má být součástí projektu, je třeba zaregistrovat v souboru *settings.py* (viz 3.3) v položce `INSTALLED_APPS`.

3.4 Docker

Obrázek 3.6: Docker logo (zdroj: Wikimedia Commons)

Docker je platforma pro vývoj, nasazení a běh aplikací. Poskytuje jednotné rozhraní pro izolaci procesů do standardizovaných balíčků, tzv. kontejnerů. Kontejnery jsou tvořeny vlastním softwarem, knihovnami a konfiguračními soubory. Jsou na sobě navzájem nezávislé, ale mohou mezi sebou komunikovat skrze definované kanály. Na rozdíl od virtuálních strojů neobsahují kontejnery operační systém, ale sdílejí kernel (jádro OS[10]) s hlavním operačním systémem. Díky tomu je režie při jejich spuštění výrazně nižší a mají mnohem menší velikost. [16]

Kontejnery jsou vytvořeny z tzv. obrazů (images). Image je spustitelný balíček, který obsahuje všechno potřebné pro běh aplikace (kód, knihovny, proměnné prostředí, konfigurační soubory). Využívá se ke skladování a sdílení aplikací. Z jednoho obrazu je možné spustit jakékoliv množství kontejnerů. [17]

Docker existuje v placené formě i jako open-source software. Funguje v prostředí Linuxu, novějších verzích Windows a ve vybraných cloudových službách. [17] Skládá se ze tří komponent:

- software - Docker démon, proces, který řídí Docker kontejnery
- objekty - entity sloužící k sestavení aplikace v Dockeru (např. obrazy, kontejnery, služby)
- registr - repozitář Docker obrazů

[10] „Jádro operačního systému je v informatice část operačního systému, která je zavedena do operační paměti při startu počítače a je jí předáno řízení.“(Wikipedie)

3.5 Ansible

Ansible je jednoduchý automatizační nástroj, který umožňuje konfiguraci systémů, víceuzlové nasazení aplikací a orchestraci jiných pokročilejších úloh. Orchestrace je řízena z jednoho řídícího stroje, jenž přistupuje ke spravovaným uzlům přes SSH nebo PowerShell. Ansible využívá architekturu, jež běží bez agentů, tedy na uzly jsou jen dočasně nainstalovány a spuštěny tzv. moduly pomocí SSH. Ve chvíli, kdy Ansible uzly neřídí, uzel nespotřebovává žádné prostředky, jelikož na něm není nainstalován démon, který by sám software spouštěl. [18]

K zápisu znovupoužitelného popisu stavu uzlů jsou vytvářeny Ansible Playbooks. Jedná se o soubory psané v jazyce YAML, který se vyznačuje pro člověka snadnou čitelností.

4 Administrátorské rozhraní

V této kapitole bude představena současná správa uživatelských účtů, tvorba webového rozhraní, která má aktuální postup nahradit a budoucí rozvoj této webové konzole.

4.1 Současná správa uživatelských účtů

Aktuálně probíhá správa uživatelských účtů spouštěním shellových skriptů[11]. Všechny příkazy je nutné spouštět pod právem *sudo* (jako administrátor). Některé příkazy lze spustit s parametry, které mohou být buď povinné, nebo nepovinné.

Seznam názvů dostupných skriptů pro GIS.lab administraci vrací příkaz gislab-help. Každý jednotlivý příkaz z tohoto soupisu pak lze spustit s příznakem -h, který zobrazí podrobnější nápovědu. Uživatel se dozví, co skript vykoná a za jakých podmínek, jakým způsobem příkaz zapsat do konzole a získá přehled parametrů, které může či musí použít.

Níže jsou popsány skripty pro vytváření uživatelů a skupin, jejich mazání a pro výpis existujících entit, protože souvisí s funkčností, kterou nabízí nové webové rozhraní. Vedle toho existují i další, např. pro zálohování dat nebo upgrade GIS.lab systému.

gislab-adduser

-g křestní jméno (povinné)

-l příjmení (povinné)

-m email (povinné)

-d popis (nepovinné)

-p heslo (nepovinné)

-s přidat uživateli status administrátora (nepovinné)

-a přidat uživatele do vybraných skupin (nepovinné)

```
$ sudo gislab-adduser -g Doug -l Adams -m adams@g.lab -p pswd adams
```

[11] https://github.com/gislab-npo/gislab/tree/314fe436e1b65783d65e61000ca6d3f8ba873b2f/system/admin

Tento příkaz má několik příznaků, z nichž část je povinná, část nepovinná. Parametr `-p` musí být použit jako poslední, těsně před názvem uživatele (username), může však být aplikován v různých obměnách. První variantou je `-p pswd`, která nastaví heslo na hodnotu pswd. Pokud je použit parametr bez argumentu (`-p`), uživatel je následně dotázán na heslo. V třetím případě je parametr z příkazu kompletně vynechán a heslo je automaticky vygenerováno. Tedy i přestože parametr patří mezi nepovinné, heslo je vždy po doběhnutí skriptu vytvořeno.

Uživatele lze přidat do více skupin zároveň použitím parametru `-a` a seznamu skupin oddělených od sebe čárkami. Pokud je k účtu přiřazen status administrátora (superuser), tak takový uživatel může na klientských počítačích spouštět operace pod právem *sudo*.

gislab-moduser

-a přidat uživatele do vybrané skupiny

-A odebrat uživatele z vybrané skupiny

-s přidat uživateli status administrátora

-S odebrat uživateli status administrátora

-m změnit email

-p změnit heslo

-d změnit popis

```
$ sudo gislab-moduser -a db, desktop -m adams@gis.lab adams
```

Upraví jeden či více atributů. Pokud nebyly předtím definovány, tak je vytvoří. Stejně jako při vytváření uživatelského účtu je možné přidávat a odebírat členství ve skupinách hromadně, jsou-li v seznamu a oddělené čárkami. V případě, že je parametr `-p` uveden bez argumentu, je heslo vygenerováno automaticky.

gislab-deluser

-b zazálohovat uživatelská data (nepovinné)

-f vynutit proběhnutí tohoto příkazu (nepovinné)

```
$ sudo gislab-deluser -f adams
```

Smaže uživatelský účet včetně příslušnosti ke skupinám. Pokud je příkaz spuštěn s parametrem -f, proběhne vše okamžitě, v opačném případě musí uživatel ještě jednou potvrdit, že si skutečně přeje účet odstranit.

gislab-addgroup

-d popis (nepovinné)

```
$ sudo gislab-addgroup -d databaze db
```

Vytvoří skupinu (v tomto příkladě *db*).

gislab-delgroup

-f vynutit proběhnutí tohoto příkazu

```
$ sudo gislab-delgroup db
```

Smaže skupinu, pokud je prázdná. Existují-li uživatelé, kteří patří do mazané skupiny, vypíše se chybová hláška a skupina zůstane v systému. Nejdříve je potřeba skupinu vyčistit přes gislab-moduser a pak příkaz zopakovat.

Obdobně jako při mazání uživatelského účtu je možné vynutit proběhnutí příkazu bez dalších dotazů. Pokud však zůstali nějací uživatelé ve skupině, je vrácena stejná chyba, která byla popsána výše.

gislab-listusers

-g vypsat pouze uživatele zvolené skupiny

```
$ sudo gislab-listusers -g db
```

Bez příznaku vypíše seznam všech uživatelů. U každého uživatele jsou uvedeny všechny jeho atributy. S příznakem -g nazev_skupiny vypíše list uživatelů příslušících této skupině, včetně jejich atributů. Heslo je zobrazeno v šifrované podobě.

```
dn: uid=gislab,ou=People,dc=gis,dc=lab
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: gislab
uidNumber: 3000
gidNumber: 3001
homeDirectory: /mnt/home/gislab
loginShell: /bin/bash
cn: Administrator GIS.lab
sn: GIS.lab
givenName: Administrator
mail: gislab@gis.lab
userPassword:: e1NTSEF9Y0JBWUdMcVgxNTdweVJreXdxZzJRaUlpTE9CaHNaSTU=
description: Main admin
```

gislab-listgroups

```
$ sudo gislab-listgroups
```

Kromě nápovědy nemá žádné parametry. Vypíše seznam všech skupin a jejich atributů, včetně identifikátorů *uid* uživatelů, kteří do dané skupiny patří.

```
dn: cn=db,ou=Groups,dc=gis,dc=lab
objectClass: posixGroup
cn: db
gidNumber: 3101
memberUid: gislab
memberUid: tolkien
memberUid: hitchhiker
memberUid: django_admin
```

Jak gislab-listusers, tak gislab-listgroups vracejí při větším množství záznamů velmi dlouhý seznam, který vypisují do konzole. Zobrazit jen požadovanou část informací umožňuje program grep.

```
\$ sudo gislab-listusers | grep uid:

uid: uid=user01
uid: uid=user02
```

4.2 Webové administrátorské a uživatelské rozhraní

Technologie použité při vývoji byly zvoleny na základě těch již aplikovaných v rámci platformy GIS.lab, resp. Gisquick. Pro zpracování byl zvolen programovací jazyk Python 3. V první řadě se jedná o budoucnost tohoto jazyka, na rozdíl od Pythonu 2. Také to umožní navázat na koncept existující knihovny pro správu uživatelů z roku 2015. Navíc je to jazyk, v němž je napsán framework Django, který byl použit také při tvorbě webové aplikace Gisquick. Ta byla původně součástí GIS.labu a tímto způsobem zůstane celá široká základna v jedné technologii.

Pro přístup k LDAP serveru bylo třeba vyvíjet konzoli přímo v GIS.labu. Ten běžel ve virtuálním vývojovém prostředí Vagrant, v počátcích na virtuálním serveru Výpočetního a informačního centra ČVUT (gislab-vm.fsv.cvut.cz), později na školním počítači (http://b802-01.fsv.cvut.cz).

Django bylo nainstalováno ve virtuálním prostředí, aby nebyl ovlivněn zbytek GIS.labu. Při vývoji webové konzole byl využit odlehčený webový server Djanga (vývojový server) a implicitní databáze SQLite. Pro vytvoření základní struktury projektu a aplikace *users* byly použity vestavěné příkazy Djanga.

Balíčků pro autentizaci Djanga vůči LDAP serveru existuje velké množství. První ze zvažovaných knihoven byla *django-auth-ldap*, která provádí autentizaci uživatelů a při jejich prvním přihlášení vytvoří účet v databázi Djanga. Nakonec byla zvolena knihovna *django_python3_ldap*, jež navíc obsahuje možnost vytvořit všechny uživatelské účty zároveň spuštěním příkazu manage.py ldap_sync_users. Kromě toho tuto knihovnu využívá i platforma Gisquick[12], která byla velkou inspirací v počátcích vývoje. *django_python3_ldap* navazuje na knihovnu *ldap3*.

Pro synchronizaci z Djanga do LDAP byly provedeny pokusy využít existující knihovny *django-ldapdb* a *django-ldap-user-registration*. Ani jednu se nepodařilo

[12]https://github.com/gislab-npo/gislab/blob/master/system/roles/service-gisquick/files/static/django/settings_custom.py

zprovoznit pro potřeby GIS.labu a proto byla vytvořena vlastní třída SyncDjangoLDAP, která také staví na *ldap3*.

LDAP server v původní konfiguraci neumožňuje, aby samotní uživatelé měnili údaje někoho jiného ani své vlastní. Proto byl vytvořen modifikační soubor:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcAccess
olcAccess: to *
  by dn.exact="cn=admin,dc=gis,dc=lab" manage
  by self write
  by * read
```

který do nastavení serveru přidal právo pro správce *cn=admin,dc=gis,dc=lab* upravovat všechny uživatele v databázi, pro všechny uživatele upravovat své vlastní údaje a pro ostatní právo čtení. Stejným způsobem bylo pozměněno již existující nastavení práv k úpravě hesla, kde byly rozšířeny kompetence správce, aby mohl upravovat všechny:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
delete: olcAccess
olcAccess: {0}
-
add: olcAccess
olcAccess: {0}to attrs=userPassword
  by dn.exact="cn=admin,dc=gis,dc=lab" manage
  by self write
  by anonymous auth
  by * none
```

Modifikační soubory byly využity při spuštění operace ldapmodify, která slouží k úpravám konfigurace LDAP serveru:

```
sudo ldapmodify -Q -Y EXTERNAL -f /cesta/k/modifikacnimu/souboru
```

Změny byly aplikovány restartováním *slapd*:

```
sudo service slapd restart
```

V situaci, kdy jsou upravovány údaje v LDAP skrze webovou konzoli, je vytvořeno připojení s přihlašovacími údaji správce *cn=admin,dc=gis,dc=lab*, proto je dostačující nastavit nejvyšší práva pouze pro něj.

Webové rozhraní má dvě hlavní části - uživatelskou a administrátorskou. Administrátorská konzole vychází z vestavěné správcovské konzole Django a je upravena potřebám GIS.labu. Uživatelská konzole je vytvořená samostatně.

4.2.1 Struktura projektu

Adresářová struktura projektu:

```
web_admin_console
├── web_console_project
│   ├── __init__.py
│   ├── ldap_auth.py
│   ├── settings.py
│   ├── settings_custom.py
│   ├── urls.py
│   └── wsgi.py
├── static
│   └── styles.css
├── templates
│   ├── registration
│   │   └── login.html
│   ├── base.html
│   ├── home.html
│   ├── password_change.html
│   ├── signup.html
│   └── user_change.html
├── users
│   ├── templatetags
│   │   ├── __init__.py
│   │   └── auth_extras.py
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── ldap_sync.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── db.sqlite3
└── manage.py
```

web_console_project

Základní struktura projektu v Djangu byla vygenerována automaticky po spuštění `django-admin startproject project` (viz 3.3.1).

Doplňující nastavení je definováno v novém souboru **settings_custom.py**, který je naimportován na konci původního souboru *settings.py*. V tomto souboru je v první řadě definována URL adresa, na které je webová konzole dostupná a zaregistrovány používané aplikace - tedy vlastní aplikace *users* popsaná níže a knihovna

django_python3_ldap, která řídí komunikaci a synchronizaci s LDAP serverem. Dále jsou zde určeny cesty k nalezení šablon a statických souborů.

Rovněž je v tomto souboru definován vlastní uživatelský model a autentizační backend využívající LDAP server namísto *ModelBackend*, který je pro Django implicitní. Pro správnou funkčnost knihovny *django_python3_ldap* jsou nakonfigurovány potřebné proměnné, mezi hlavními URL LDAP serveru, úroveň, na níž se mají vyhledávat v LDAP uživatelé či namapování ekvivalentních atributů mezi LDAP a Djangem. Kromě toho je na konci volána vlastní funkce ze souboru *ldap_auth.py*, jež provádí synchronizaci skupin do Djanga.

Soubor **ldap_auth.py** sestává z jediné funkce custom_sync_user_relations(). Používaná knihovna *django_python3_ldap* při synchronizaci předpokládá existenci atributu *MemberOf* u uživatelského záznamu v databázi na LDAP serveru. Ten obsahuje informace o skupinách, jichž je uživatel členem. GIS.lab server tento parametr u uživatelů nemá a nastavení nebylo možné změnit, data o příslušnosti jsou ukládána pouze na straně skupin. Proto byla napsána vlastní funkce, která synchronizaci provádí.

Nejdříve je vytvořeno anonymní připojení k LDAP serveru a získán seznam existujících skupin. Ty jsou porovnány se skupinami v Djangu a v případě nesrovnalostí je Django aktualizováno podle stavu na LDAP serveru. Stejným způsobem je provedena synchronizace členství, kdy jsou postupně procházeny všechny skupiny a je testováno, zda je uživatel jejich členem na LDAP serveru. Po doběhnutí této funkce jsou z pohledu skupin Django a LDAP konzistentní.

Pseudokód 1 Funkce custom_sync_user_relations

```
1: vytvoř připojení k LDAP serveru
2: uživatel = synchronizovaný uživatel
3: ldap_skupiny = existující skupiny na LDAP serveru
4: django_skupiny = existující skupiny v Djangu
5: for skupina_D z django_skupiny do
6:     if skupina_D není v ldap_skupiny then
7:         odstraň skupina_D z Djanga
8:     end if
9: end for
```

```
10: for skupinaL z ldapSkupiny do
11:   skupina_D = vytvoř nebo získej stejnou skupinu v Djangu
12:   if uživatel je ve skupina_L then
13:     přidej uživatel do skupina_D
14:   else
15:     odstraň uživatel ze skupina_D
16:   end if
17:   if uživatel je ve skupině gislabadmins then
18:     přidej uživatel status administrátora
19:   end if
20: end for
```

Soubor **urls.py** v základní podobě obsahuje pouze URL adresu k vestavěné administrátorské konzoli Djanga. K ní jsou přidány cesty ke správě uživatelů a k vlastní aplikaci *users* a definována šablona *home.html* jako domovská stránka.

users

Tento adresář se skládá ze souborů tvořících Django aplikaci s názvem *users*, základní struktura byla vytvořena příkazem python manage.py startapp users.

V souboru **models.py** je definován vlastní uživatelský model CustomUser, který dědí z třídy AbstractUser a rozšiřuje ji o nový textový atribut description.

forms.py definuje třídy vlastních formulářů, které všechny dědí z vhodných tříd Djanga a přidávají potřebnou funkcionalitu. CustomUserCreationForm slouží k registraci nových uživatelů, CustomUserChangeForm je použit při změně osobních údajů. Oba formuláře mají nastaven model na CustomUser a definován seznam polí, která se zobrazí při registraci, resp. úpravě údajů. V třídě FieldsRequiredMixin, která je předána oběma jako parametr, jsou určeny povinné a nepovinné atributy tak, aby odpovídaly atributům na LDAP serveru. Povinné jsou, vedle uživatelského jména a hesla, křestní jméno, příjmení a emailová adresa. Nepovinné pak zůstává pouze pole popisu. U registračního formuláře je navíc přetížena metoda *save()* tak, aby byl při vytváření nový uživatel uložen i na LDAP server.

Posledním formulářem je CustomAdminPasswordChangeForm, jenž slouží ke změně hesla. U něj je také přetížená funkce *save()* tak, aby nové heslo bylo upraveno i na serveru.

V souboru **admin.py** jsou veškeré změny, které upravují standardní nastavení administrátorské konzole Djanga. Sestává ze dvou tříd, jedné pro správu uživatelů a druhé pro správu skupin (rolí).

V CustomUserAdmin je připojen vlastní model CustomUser a vlastní formuláře pro vytváření uživatele, změnu údajů a hesla. Implicitní nastavení je upraveno. Zobrazují se jen pole a filtry, které jsou pro administrátorskou konzoli potřebné. Při manipulaci s uživatelským účtem přes administrátorskou konzoli se nevolají metody formuláře, ale metody třídy UserAdmin. Proto jsou zde přetíženy metody save_model(), která vytváří či upravuje uživatele, a delete_model(), jež uživatele maže. Po úpravě promítají tyto změny také na LDAP server.

Podobně upravená je i třída CustomGroupAdmin. Ta sice ponechává původní model Group, ale z atributů je zobrazen jenom název. Metody save_model() a delete_model() jsou přetížené stejně jako u administrace uživatelů a změny jsou promítány na LDAP server.

Na konci souboru jsou obě vlastní třídy zaregistrovány a popisy v Django administraci jsou upraveny, aby lépe odpovídaly potřebám GIS.lab konzole.

Soubor **ldap_sync.py** je zcela nově vytvořen a obsahuje vlastní třídu SyncDjangoLDAP, jež provádí synchronizaci změn z Djanga do LDAP. Při její inicializaci je vytvořeno připojení k LDAP serveru s přihlašovacími údaji pro hlavního GIS.lab administrátora se jménem *cn=admin,dc=gis,dc=lab*, heslo je uloženo v textovém souboru. V rámci destruktoru je připojení zrušeno.

Pseudokód 2 Metoda __init__()

```
1: vytvoř připojení k LDAP serveru
```

Pseudokód 3 Metoda __del__()

```
1: zruš připojení k LDAP serveru
```

Pro synchronizaci je využita externí knihovna ldap3, konkrétně operace ADD, MODIFY a DELETE. Metoda change_user() zjistí, která data byla editována, a provede tuto změnu i v LDAP. Upravuje nejenom vlastní atributy uživatele, ale i členství ve skupinách. Jen změna hesla má jinou metodu, change_password(), protože se jedná o atribut, který má vlastní formulář pouze pro heslo.

Pseudokód 4 Metoda change_user(uživatel, změněná_data)

```
1: if křestní_jméno v změněná_data then
2:    změň křestní_jméno uživatele v LDAP
3: end if
4: if příjmení v změněná_data then
5:    změň příjmení uživatele v LDAP
6: end if
7: if křestní_jméno nebo příjmení v změněná_data then
8:    změň celé_jméno uživatele v LDAP
9: end if
10: if email v změněná_data then
11:    změň email uživatele v LDAP
12: end if
13: if popis v změněná_data then
14:    změň popis uživatele v LDAP
15: end if
16: if role v změněná_data then
17:    role_Django = všechny role v Djangu
18:    for role z role_Django do
19:       člen_Django = je uživatel členem role v Djangu?
20:       člen_LDAP = je uživatel členem role v LDAP?
21:       if uživatel je člen_Django a není člen_LDAP then
22:          přidej uživatele do skupiny v LDAP
23:       else if uživatel není člen_Django a je člen_LDAP then
24:          odstraň uživatele ze skupiny v LDAP
25:       end if
26:       if uživatel je ve skupině gislabadmins then
27:          přidej uživatel status administrátora
28:       end if
29:    end for
30: end if
```

Pseudokód 5 Metoda change_password(uživatel, nové_heslo)

```
1: změň heslo na nové_heslo pro uživatele v LDAP
```

Nového uživatele ukládá metoda save_user(). Některé atributy jsou zvoleny jako konstanty (např. identifikační číslo), protože se jedná o interní hodnoty LDAP a v Django neexistuje jejich ekvivalent. Do budoucna by tento problém mělo vyřešit propojení s Python knihovnou pro správu (viz 4.2.3). Odstranění účtu řídí delete_user(), jež nejdřív odstraní všechny vztahy uživatele a pak jej teprve smaže.

Pseudokód 6 Metoda save_user(uživatel, heslo)

```
1: vytvoř uživatele v LDAP
```

Pseudokód 7 Metoda delete_user(uživatel)

```
1: role_Django = všechny role v Djangu
2: for role z role_Django do
3:     if uživatel členem role v LDAP then
4:         odstraň uživatele z role v LDAP
5:     end if
6: end for
7: odstraň uživatele z LDAP
```

Skupiny lze pouze vytvořit přes save_group() či odstranit v metodě delete_group(). Není umožněno provádět žádné úpravy, protože jediným existujícím atributem je název role. Při tvorbě je zvolena konstanta pro identifikační číslo skupiny, propojení s Python knihovnou tuto situaci vyřeší (viz výše). Před odstraněním jsou nejdříve zrušeny veškeré vztahy k uživatelům. V tomto ohledu se webová konzole chová odlišně od současné správy přes administrátorské příkazy. Příkaz gislab-delgroup nedovoluje smazat skupinu, pokud není prázdná.

Pseudokód 8 Metoda save_group(role, změněná_data)

```
vytvoř roli v LDAP
```

Pseudokód 9 Metoda delete_group(role)

```
členové = všichni uživatelé, kteří patří do role
for člen z členové do
  odstraň člena z role v LDAP
end for
odstraň roli z LDAP
```

Soubor **views.py** obsahuje logiku propojující formuláře a modely se zobrazením přes uživatelskou konzoli. Jedná se o tři pohledy (třídy), jeden pro registraci (SignUp), druhý pro úpravu uživatelských údajů (ChangeUser) a třetí pro změnu hesla (ChangePassword). U všech je definováno, který formulář mají použít, ve které šabloně je zobrazen a kam uživatele přesměrovat po správném vyplnění a odeslání. U pohledu editace údajů je přetížená funkce form_valid() tak, aby navíc volala třídu SyncDjangoLDAP a promítala změny i do LDAP.

Posledním souborem v adresáři *users* je **urls.py**, jenž definuje URL adresy ke třem výše zmiňovaným pohledům - registraci, změně údajů a úpravě hesla.

templates, templatetags, static

Složka **templates** obsahuje šablony (templates), psané v jazyce HTML, které umožňují vypisovat vybraná data z modelů do prohlížeče. Cesta k adresáři musí být registrována v souboru *settings.py* v proměnné DIRS u položky TEMPLATES.

Pro přístup k proměnným a některým funkcím Pythonu slouží složené závorky. V případě proměnných se jedná o závorky dvojité: {{ variable }}.

Django funguje na principu dědičnosti. Bázovou šablonou je **base.html**. Vložením textu {% extends "base.html" %} na začátek jiné šablony, např. *child.html*, je nejdříve načtena šablona *base.html*, definující základní bloky, a až následně je k nim přidán obsah *child.html*. Tímto způsobem jsou omezeny duplicity v jednotlivých šablonách.

U tohoto projektu jsou například v bázové šabloně definovány navigační prvky společné pro níže uvedené šablony, které z ní všechny dědí. Na každé stránce se v horní části zobrazuje logo GIS.labu, které po kliknutí přesměruje uživatele na hlavní stránku *home.html*. V případě přihlášeného uživatele se navíc zobrazuje tlačítko pro odhlášení.

signup.html obsahuje jednoduchý interaktivní formulář s informacemi, která pole jsou povinná pro platné vyplnění a jaké podmínky musí splňovat. V situaci, kdy je registrace provedena správně, je uživatel přesměrován na přihlašovací stránku, v opačném případě je zobrazena chybová hláška a potřebná data je nutné opravit, resp. doplnit.

Template **login.html** sestává z jednoduchého formuláře pro vyplnění uživatelského jména a hesla. Při neúspěšném pokusu je vypsána chyba, po zdárném vyplnění je uživatel přesměrován na domovskou stránku *home.html*.

První šablona, s níž se uživatel při zobrazení hlavní stránky setká, je **home.html**. Ta ukazuje rozdílné výsledky nepřihlášenému uživateli a přihlášenému. V prvním případě je dostupný rozcestník, který jej navede na stránky přihlášení či k registraci. V druhém případě, tedy pokud je již přihlášen, se zobrazí jeho osobní informace a aktivní role. Přímo zde nemůže nic upravovat, ale pomocí tlačítka Edit je přesměrován na stránku s úpravou osobních údajů.

Šablona **user_change.html** obsahuje formulář pro editaci osobních údajů, který umožňuje upravit jeden či více záznamů naráz. Heslo se zde nezobrazuje, ale přes odkaz lze pokračovat na stránku *password_change.html*, kde je možné provést jeho změnu.

Složka **static** umístěná v hlavním adresáři projektu obsahuje soubor *styles.css*, jenž popisuje způsob zobrazení elementů, které jsou součástí jednotlivých šablon uživatelské konzole. To je umožněno načtením tohoto souboru pomocí {% load static %} do bázové šablony. K vytvoření designu byly použity kaskádové styly (CSS).

Django umožňuje vytvořit si vlastní filtry. Ty jsou specifikovány v souboru **auth_extras.py** uloženém v adresáři aplikace *users/templatetags* a k šablonám jsou připojeny pomocí {% load auth_extras %}. Filtru foo lze předat hodnotu proměnné *var* a argument *arg*. Poté, co je filtr zaregistrován jako *django.template.Library.filter()* a definována žádaná funkce, je možné jej zavolat v šabloně příkazem:

{{ var|foo:"arg" }}

Pro potřeby zobrazení aktivních rolí uživatele v šabloně *home.html* byly vytvořeny dva filtry. První zjišťuje všechny existující role v databázi Djanga a na něj navazuje druhý, který určuje, zda je uživatel jejich členem. Výsledky jsou pak zobrazeny v GUI.

db.sqlite3

Pro vývoj byla využita implicitní databáze Djanga SQLite. Z hlediska uživatelů a rolí jsou důležité především tři tabulky *auth_group*, *users_customuser* a *users_customuser_groups*.

Tabulka **auth_group** obsahuje pouze názvy existujících rolí.

auth_group		
name	id	name
type	integer	varchar(80)

Tab. 4.1: Atributy tabulky auth_group

Tabulka **users_customuser** se skládá z osobních údajů uživatelů včetně zašifrovaného hesla, data vytvoření, posledního přihlášení a interních statusů Djanga.

users_customuser						
name	id	password	last_login	is_superuser	username	first_name
type	integer	varchar(128)	datetime	bool	varchar(150)	varchar(30)

Tab. 4.2: Atributy tabulky users_customuser 1/2

users_customuser						
name	last_name	email	is_staff	is_active	date_joined	description
type	varchar(150)	varchar(254)	bool	bool	datetime	text

Tab. 4.3: Atributy tabulky users_customuser 2/2

Tabulka **users_customuser_groups** propojuje obě výše zmíněné tabulky, tj. příslušnost uživatelů k jednotlivým skupinám.

users_customuser_groups			
name	id	customuser_id	group_id
type	integer	integer	integer

Tab. 4.4: Atributy tabulky users_customuser_groups

4.2.2 Spuštění projektu

Ve finální verzi bude webová konzole zaintegrována do GIS.labu pomocí Docker kontejneru jako samostatná služba. Při odevzdání je projekt ještě ve vývojové fázi. Pro jeho spuštění je třeba mít nainstalovanou platformu GIS.lab alespoň ve virtuální režimu. Projekt si uživatel může sestavit sám či může využít připravený skript, který je dostupný v repozitáři. [13]

Shellový skript **setup_script.sh** připraví projekt ke spuštění, jediné co musí uživatel udělat, je upravit proměnnou PROJ_PATH, která obsahuje cestu k adresáři, do něhož má být projekt uložen. Poté jen stačí spustit skript příkazem:

```
./setup_script.py
```

Nejdříve je vytvořen adresář projektu, poté je nainstalován Python 3 s programem pip a virtuálním prostředím. V dalším kroku je spuštěno virtuální prostředí a v něm je nainstalováno Django a knihovna *django_python3_ldap*. Pak je vytvořena základní struktura projektu a aplikace. Repozitář diplomové práce je naklonován do dočasného adresáře, odkud jsou všechny upravené či nově vytvořené soubory překopírovány na správnou lokaci v projektu. Tento dočasný adresář je následně smazán. Na konec je provedena migrace databáze.

Tím je projekt téměř připraven. Webové rozhraní bude dostupné na adrese, která je uvedena v nastavení *settings_custom.py* v proměnné ALLOWED_HOSTS. Proto musí uživatel tuto cestu upravit tak, aby odpovídala adrese serveru. Pro spuštění projektu je třeba přejít do složky projektu a aktivovat virtuální prostředí:

```
source virenv/bin/activate
```

[13] https://github.com/ctu-geoforall-lab-projects/dp-kulovana-2019

Poté již lze spustit vývojový server:

```
python manage.py runserver 0:8080
```

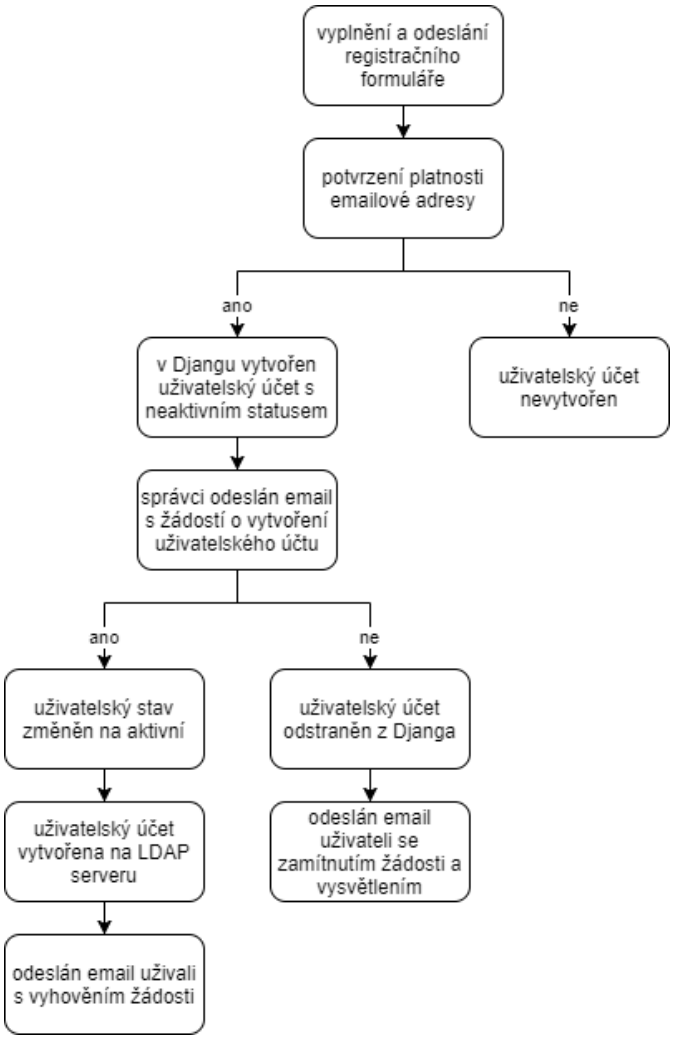
Při zobrazení stránky uvedené v proměnné ALLOWED_HOSTS a portu 8080 (např. http://b802-01.fsv.cvut.cz:8080) se uživatel dostane na domovskou stránku webové konzole.

4.2.3 Budoucí vývoj

V době odevzdání probíhá proces registrace tak, že po odeslání validního formuláře je okamžitě vytvořen uživatelský účet v Djangu i na LDAP serveru.

Obrázek 4.1: Vytváření uživatele - současný stav (zdroj: Tereza Kulovaná)

V konečné formě by mělo mezi jednotlivými činnostmi probíhat potvrzení přes email. Konkrétně přímo po registraci bude uživateli odeslán email na vyplněnou adresu, který bude nutné před následujícími akcemi potvrdit. Poté bude vytvořen účet v Djangu s neaktivním statusem, který uživateli zabraňuje se do systému přihlásit. Správce obdrží email s žádostí o vytvoření uživatelského účtu, jenž bude přímo obsahovat volby pro potvrzení a zamítnutí. Pokud bude požadavek zamítnut, administrátor vyplní zdůvodnění tohoto rozhodnutí, účet bude z Djanga smazán a uživatel obdrží vysvětlující zprávu. V případě vyhovění žádosti se účet stane aktivním a vytvoří se jeho ekvivalent v LDAP. Uživatel bude o kladném rozhodnutí spraven emailem.



Obrázek 4.2: Vytváření uživatele - finální stav (zdroj: Tereza Kulovaná)

Ověřování platnosti emailové adresy bude doplněno i do části uživatelské konzole, jež umožňuje editaci osobních údajů.

Vývoj knihovny pro správu uživatelů psané v jazyce Python má počátek v roce 2015. Její vznik byl iniciován především proto, aby nahradila stávající shellové skripty, protože vývojářům GIS.labu je bližší Python a následné úpravy pro ně budou tímto způsobem snadnější. Neméně důležitá je i možnost propojení s webovým rozhraním, a tak byly práce na této knihovně nedávno po delší odmlce obnoveny.

Aktuálně jsou zpracovány skripty pro tvorbu, úpravu a mazání uživatelských účtů a správu známých zařízení v síti. Před propojením s webovou konzolí je bude potřeba ještě dokončit a vytvořit nové pro správu skupin (rolí).

Přes opakované pokusy se nepodařilo zprovoznit vypisování informačních zpráv (tzv. logů) pro úroveň DEBUG, které by usnadnily vývoj. Aktuálně jsou dostupné

pouze hlášky pro úroveň INFO a vyšší. Před dalším postupem bude třeba tuto problematiku hlouběji prozkoumat a vyřešit.

V rámci vlastních funkcí jsou vypisovány informační logy v podobě prostého textu. Ty dostanou vhodnější formátování a budou doplněny o datum a čas, kdy daná činnost proběhla. Pro lepší informovanost o dění budou přidány chybové hlášky. Veškeré informace, které se aktuálně ukazují v konzoli, budou ukládány do souborů.

Administrátorská konzole, která vychází z konzole Djanga, obsahuje informační zprávy, jež se správci zobrazují ve webovém prohlížeči. Tato funkcionalita bude doplněna i pro uživatelskou konzoli.

Synchronizace skupin do Djanga je realizována funkcí custom_sync_user_relations(). V rámci úprav kódu se stane členskou metodou nové třídy.

O role, které budou uživatele opravňovat k využití jednotlivých balíčků GIS.labu, si bude moci uživatel zažádat sám. Nyní se v uživatelské konzoli klient pouze dozví, které role jsou pro něj aktivní. Ve finální verzi si bude moci uživatel vybrat zvolené role během registrace či si o jejich změnu zažádat přes uživatelskou konzoli. Vyhovění či zamítnutí požadavku provede administrátor a uživatel bude informován emailem. Podobně bude implementována i žádost o navýšení kapacity databáze a další služby popsané v kapitole GIS.lab (2.4).

Pokud se uživatel při instalaci platformy GIS.lab rozhodne, že chce mít dostupné webové administrační rozhraní, bude rozhraní zahrnuto jako další služba (service). Z připravených konfiguračních souborů vznikne Docker obraz, na jehož základě bude vytvořen Docker kontejner. Díky tomu webový server administrátorské konzole poběží v izolovaném prostředí.

Pro zápis konfigurace bude využit Ansible playbook[14]. Soubor pro konfiguraci bude vycházet z aktuálně používaného skriptu *setup_script.sh*.

[14]inspirace bude čerpána z integrace Gisquicku: https://github.com/gislab-npo/gislab/blob/python-lib/system/roles/service-gisquick/tasks/main.yml

Závěr

Tato diplomová práce si kladla za cíl vytvořit webové administrátorské rozhraní pro potřeby platformy GIS.lab. Zadání se podařilo splnit v primární, neodladěné formě. Během zpracování se vyskytlo několik dalších požadavků, které bude třeba do finální podoby zakomponovat.

V době odevzdání práce obsahuje administrátorská konzole základní funkcionalitu: přidávání nových uživatelů, jejich úpravu a odebírání, správu příslušnosti k rolím (skupinám), vytváření a mazání těchto rolí. Pro běžného uživatele je prozatím dostupná registrace, přihlášení pod svým účtem do uživatelské konzole, která zobrazuje osobní informace a role, jichž je uživatel členem. Své osobní informace, včetně hesla, může uživatel měnit.

Komunikace mezi webovou aplikací a LDAP serverem funguje obousměrně, ale vyžaduje do budoucna ještě odladění, doplnění některých okrajových případů a v první řadě zajištění pravidelné automatické synchronizace. Nejzásadnější problematické situace v aktuální verzi nastávají v případě odstranění uživatele na LDAP serveru - změna se nyní okamžitě neprojeví v databázi webové aplikace a pokud byl uživatel v momentu smazání přihlášen, může i nadále upravovat svoje údaje.

Role může aktuálně uživateli přiřazovat jen administrátor, záměrem je umožnit uživateli vybrat si potřebné role už při registračním procesu, případně následně zažádat o změnu přes uživatelskou konzoli.

V současnosti jsou všechny změny propisovány do databáze okamžitě. Cílovým stavem, po vyplnění registračního formuláře a ověření emailové adresy uživatele, je poslat emailové upozornění administrátorovi s žádostí o vytvoření nového účtu. Teprve po potvrzení správcem funkční uživatelský účet skutečně vznikne. Stejným potvrzovacím procesem budou procházet i žádosti o přiřazení nové role.

Důležitým prvkem, který bude třeba přidat, je ověřování platnosti emailové adresy během procesu vytváření nového uživatele i při případné následné změně emailu samotným uživatelem. Pro lepší přehlednost budou pro uživatele také přidány informativní zprávy o úspěšné změně osobních údajů či hesla.

Administrátorská konzole je ponechána v původním designu Djanga, design uživatelské konzole byl inspirován vzhledem webové platformy Gisquick, avšak v konečné verzi dozná ještě dalších úprav.

Podstatným bodem z hlediska vývoje je zprovoznění plnohodnotného výpisu logů a doplnění dalších informativních zpráv o probíhajících procesech ve webové konzoli. Veškerá současná řešení i finální podobu bude třeba řádně ověřit pomocí automatizovaných testů.

Otázkou, na níž bude třeba teprve najít odpověď, je, jakým způsob bude vhodné se postavit ke schopnosti administrátorů měnit osobní údaje a případně i heslo dalších administrátorů. Nyní mají všichni správci přístup k údajům všech ostatních osob v databázi a u každé z nich mohou upravovat cokoliv. Bezpečnost je dalším prvkem, který bude vyžadovat hlubší prozkoumání, především s ohledem na předávání hesla mezi webovou aplikací a LDAP serverem a také následné šifrování hesla na LDAP serveru.

V roce 2015 vznikl prvotní návrh Python knihovny, která by měla nahradit stávající shellové skripty při tvorbě uživatelů (více viz kapitola 4.2.3). Její vývoj byl na jistou dobu pozastaven, ale jedním z ambicioznějších cílů této práce bylo její dokončení a propojení s webovou aplikací. To se před termínem odevzdání nepodařilo, ale práce na knihovně byly obnoveny a v krátké době snad budou úspěšně završeny.

Integrace do stávající architektury platformy GIS.lab proběhne přes Docker kontejner. Při instalaci platformy GIS.lab si bude moci uživatel zvolit, zda si přeje mít dostupné i webové administrační rozhraní či ne. Pokud ano, z konfiguračních souborů vznikne Docker obraz, na jehož základě bude vytvořen kontejner. Konzole, včetně webového serveru, poběží v izolovaném prostředí. Pro zápis konfigurace bude využit Ansible playbook.

Webové administrátorské a uživatelské rozhraní tedy nyní obsahuje základní funkcionalitu s tím, že bude snaha je v brzké době dokončit a zařadit mezi služby platformy GIS.lab.

Seznam zkratek

GIS	Geografický informační systém (Geographic information system)
GUI	Grafické uživatelské rozhraní (Graphical user interface)
LDAP	Lightweight Directory Access Protocol
LAN	Local Area Network
OS	Operační systém (Operating system)
UI	Uživatelské rozhraní (User interface)
IP	Internet Protocol
MVC	Model-view-controller
WMS	Webová mapová služba (Web map service)
WFS	Web feature service
URL	Jednotná adresa zdroje (Uniform Resource Locator)
WSGI	Web Server Gateway Interface
HTML	Hypertext Markup Language
DPZ	Dálkový průzkum Země
NFS	Network File System
DN	Distinguished Name
NFS	Network File System

Literatura

[1] *Django documentation* [online]. [cit. 2019-05-05]. Dostupné z: https://docs.djangoproject.com/en/2.1/.

[2] *GIS.lab 0.7 documentation* [online]. [cit. 2019-04-23]. Dostupné z: https://gislab.readthedocs.io/en/latest/.

[3] *GIS.lab Atom* [online]. [cit. 2019-04-25]. Dostupné z: http://gislab-npo.github.io/gislab/.

[4] *Gisquick* [online]. [cit. 2019-04-28]. Dostupné z: http://gisquick.org.

[5] *Gisquick 1.0 documentation* [online]. [cit. 2019-04-28]. Dostupné z: gisquick.readthedocs.io/en/latest/.

[6] *Otevřená data* [online]. Ministerstvo vnitra ČR. [cit. 2019-04-27]. Dostupné z: https://data.gov.cz/datov%C3%A9-sady.

[7] *Learn About LDAP* [online]. [cit. 2019-05-01]. Dostupné z: https://ldap.com/learn-about-ldap/.

[8] *OpenLDAP Software 2.4 Administrator's Guide* [online]. [cit. 2019-05-01]. Dostupné z: http://www.openldap.org/doc/admin24/.

[9] *etianen/django-python3-ldap* [online]. [cit. 2019-04-28]. Dostupné z: https://github.com/etianen/django-python3-ldap.

[10] CANNATA, Giovanni. *ldap3 Documentation Release 2.5* [online]Aug 2018. [cit. 2019-04-29]. Dostupné z: https://buildmedia.readthedocs.org/media/pdf/ldap3/stable/ldap3.pdf.

[11] PILGRIM, Mark. *Dive Into Python.* Createspace Independent2009. ISBN 9781441413024.

[12] KULOVANÁ, Tereza. *Zásuvný modul QGIS pro terénní radiační průzkum* [online]. Praha, 2017. Bakalářská práce. ČVUT v Praze, Fakulta stavební, Katedra geomatiky. Vedoucí práce Ing. Martin Landa, Ph.D. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/70669/

F1-BP-2017-Kulovana-Tereza-tereza-kulovana-bp-2017.pdf?sequence=1&isAllowed=y.

[13] *Python2orPython3 - Python Wiki* [online]. [cit. 2019-05-15]. Dostupné z: https://wiki.python.org/moin/Python2orPython3.

[14] *Nick Coghlan's Python Notes - Nick Coghlan's Python Notes 1.0 documentation* [online]. [cit. 2019-05-15]. Dostupné z: http://python-notes.curiousefficiency.org/en/latest/index.html.

[15] ADRIAN HOLOVATY, Jacob KAPLAN-MOSS. *The Definitive Guide to Django: Web Development Done Right, Second Edition.* Apress2009. ISBN 9781430219361.

[16] TURNBULL, James. *The Docker Book: Containerization Is the New Virtualization.* Turnbull Press2014. ISBN 9780988820203.

[17] *Docker Documentation* [online]. [cit. 2019-05-20]. Dostupné z: https://docs.docker.com.

[18] *Ansible Documentation* [online]. [cit. 2019-05-21]. Dostupné z: https://docs.ansible.com/ansible/.

A User guide

GIS.lab web console is as an application for management of GIS.lab users and roles. It consists of two sections - user console (A.2) and admin console (A.3). Data are synchronized between LDAP server and local Django database. It is currently available only as a Django project in virtual environment of GIS.lab Desktop. Before installation of web console, please connect to your existing GIS.lab server or install it on the basis of this instruction: https://gislab.readthedocs.io/en/latest/installation/index.html.

A.1 Installation

1. Download script called setup_script.sh from repository https://github.com/ctu-geoforall-lab-projects/dp-kulovana-2019 to your GIS.lab server.

2. Go to the directory where the script is located.

3. Open the script and assign the variable PROJ_PATH to a full pathname of a directory where you want the web console project to be located (e.g. PROJ_PATH=/mnt/home/gislab/web_console).

4. Run command:

```
./setup_script.sh
```

5. Go to the project directory (e.g. /mnt/home/gislab/web_console), open file web_console_project/settings_custom.py and assign the variable ALLOWED_HOSTS to your GIS.lab server URL.

6. Activate virtual environment by running command:

```
source virenv/bin/activate
```

7. Sync your existing LDAP users and groups to Django database by running:

```
manage.py ldap_sync_users
```

8. Start up the server with:

```
python manage.py runserver 0:8080
```

9. Go to the homepage at ALLOWED_HOSTS:8080 where ALLOWED_HOSTS stands for assigned URL.

10. Stop server with:

```
CTRL+C
```

11. Deactivate virtual environment by running command:

```
deactivate
```

A.2 User console

There is a user console available on the homepage (e.g. http://b802-01.fsv.cvut.cz:8080). Its objective is to allow users to registrate and to edit their personal information.

As a non-authenticated user, you see two buttons which redirect you to either login page or a registration.

Figure A.1: User console - home page (zdroj: Tereza Kulovaná)

Username, first name, last name, email address and *password* are mandatory fields for successful registration, *description* is optional. For sign up push the SIGN UP button. If the form is not valid, the error, with information of what went wrong, shows. In case of a successful validation, you are redirected to login page.

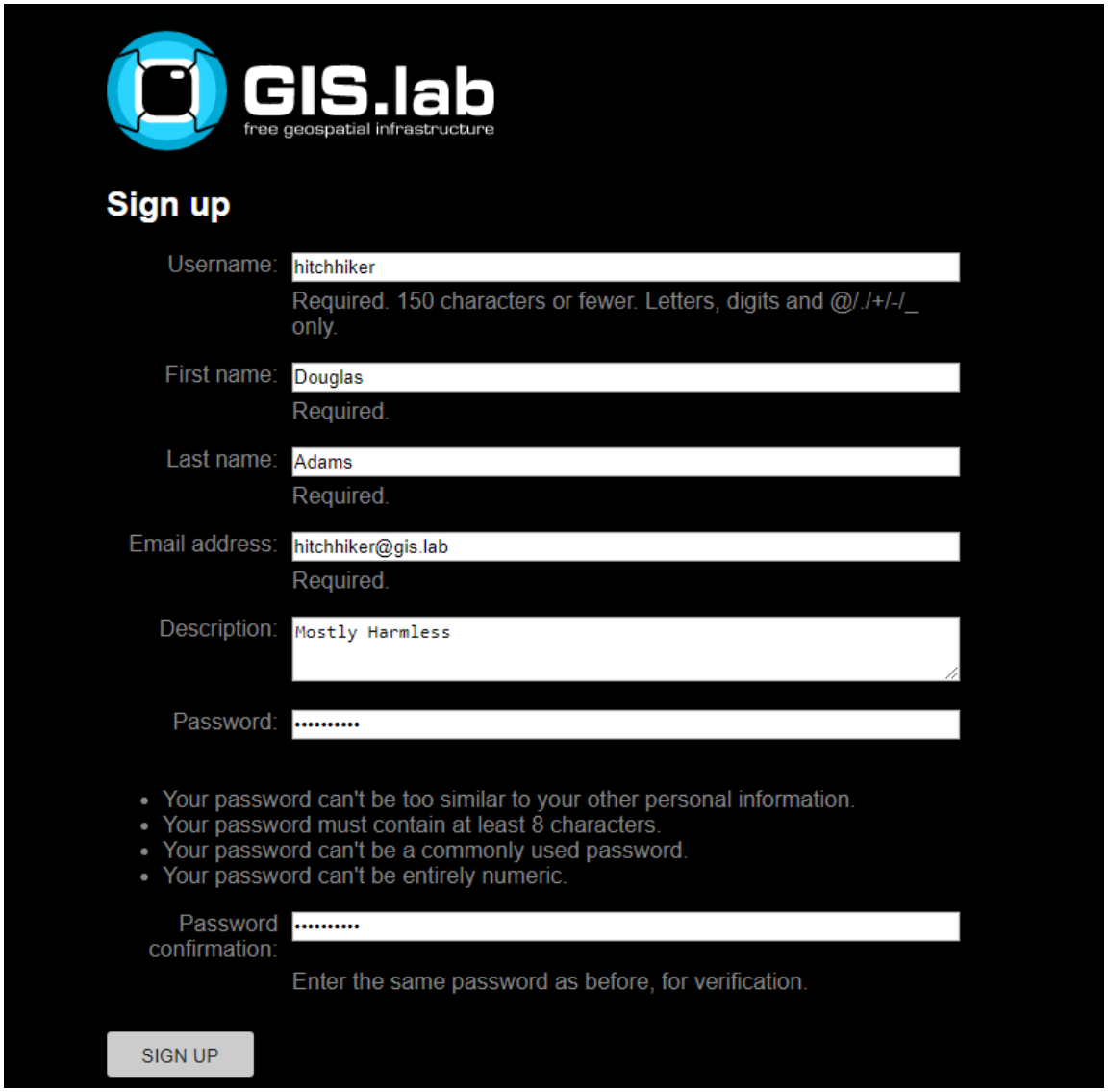


Figure A.2: User console - registration (zdroj: Tereza Kulovaná)

On login page you need to fill in your credentials. After successful authentication, you are redirected to the page with your personal information.

Figure A.3: User console - login page (zdroj: Tereza Kulovaná)

For authenticated user, homepage displays their personal details and active roles. Edit button allows you to change your personal details.

Figure A.4: User console - home page (authenticated user) (zdroj: Tereza Kulovaná)

From all of your atributes, you are not allowed to change username field. For changing password, you need to click on this form link in the text and you will be redirected to the correct page.

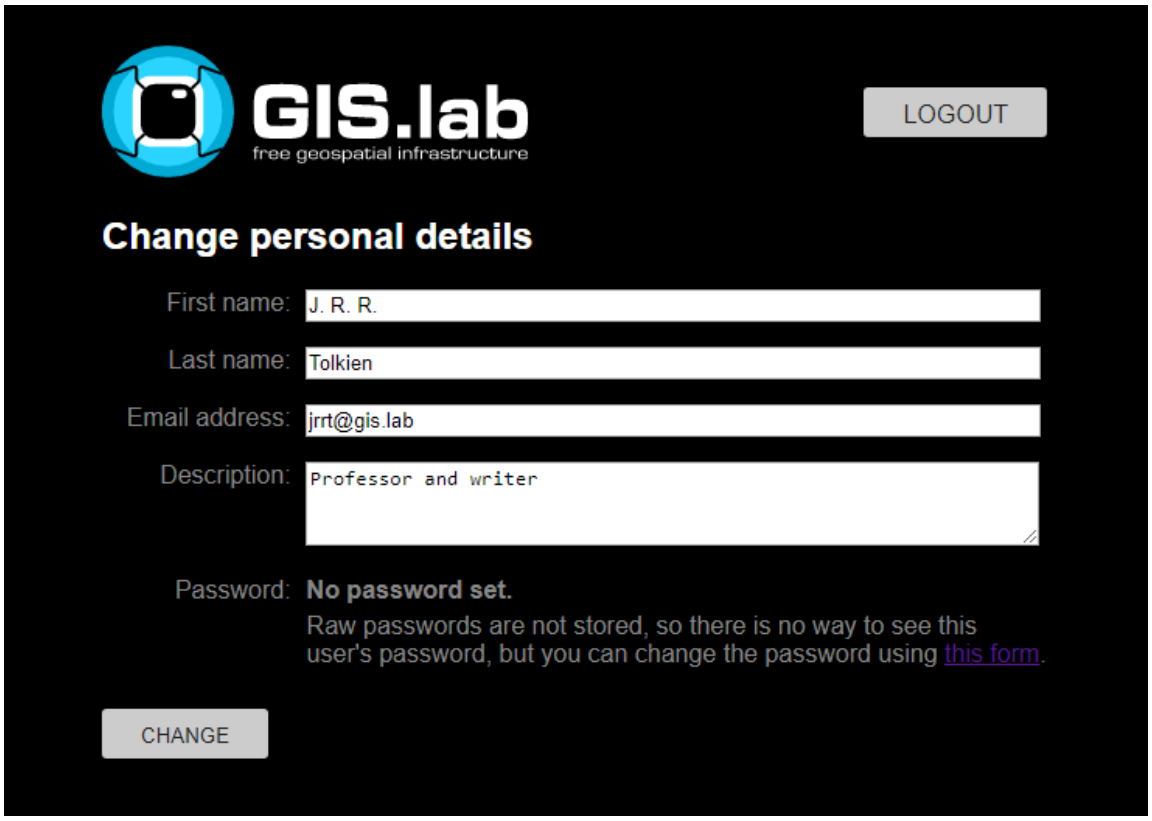


Figure A.5: User console - edit personal details (zdroj: Tereza Kulovaná)

Write your password twice for verification and push the CHANGE button. In case of a successful change, your credentials are updated and you are redirected to the homepage.

Figure A.6: User console - password change (zdroj: Tereza Kulovaná)

Logout is available by LOGOUT button in the upper right corner.

A.3 Admin console

To access the admin console, you need to write */admin* behind your homepage URL (e.g. http://b802-01.fsv.cvut.cz:8080/admin). Fill in your credentials to log in to the admin console.

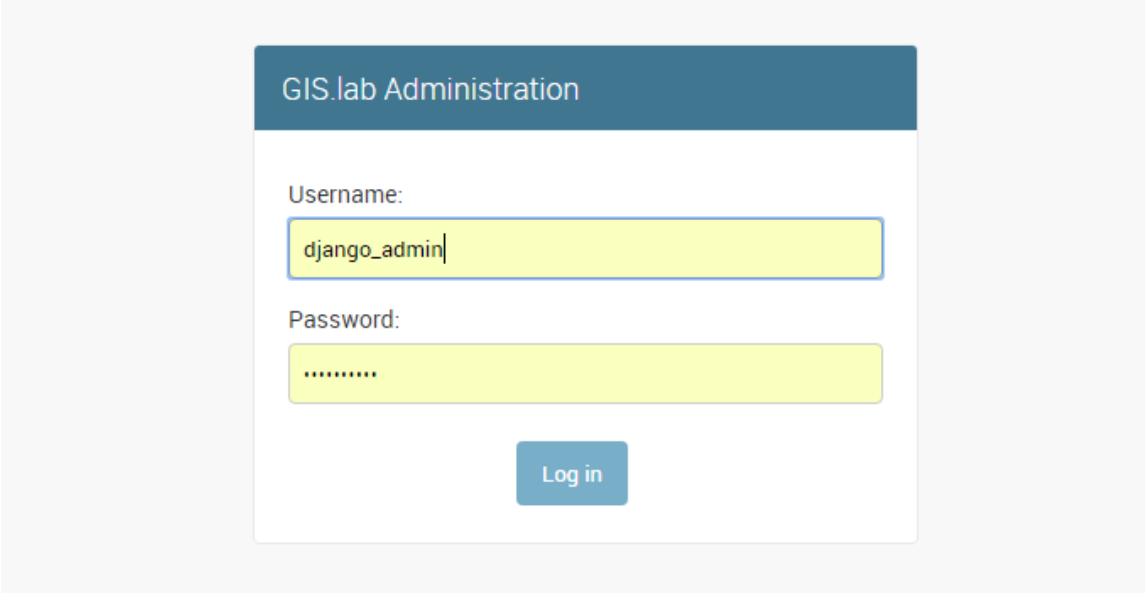


Figure A.7: Admin console - login page (zdroj: Tereza Kulovaná)

There is a list of recent actions on the welcome page of GIS.lab administration, as well as links to the list of users and the list of groups (roles).

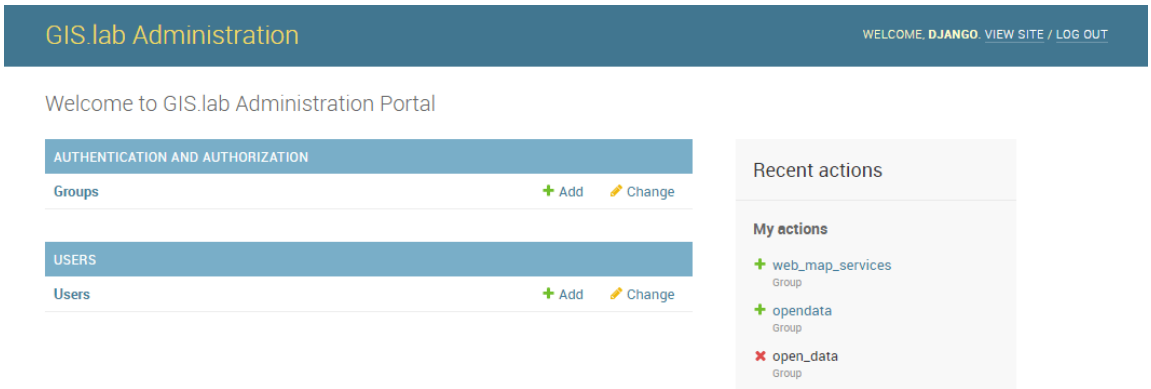


Figure A.8: Admin console - home page (zdroj: Tereza Kulovaná)

User list displays all users in database and their *username, first and last name, email address, description* and *superuser status*. You can filter users by superuser

status or by group membership. New user can be added by ADD USER button in the upper right corner. You can change group membership and personal details of a user by clicking on their username. If you want to delete user, click on their username as well.

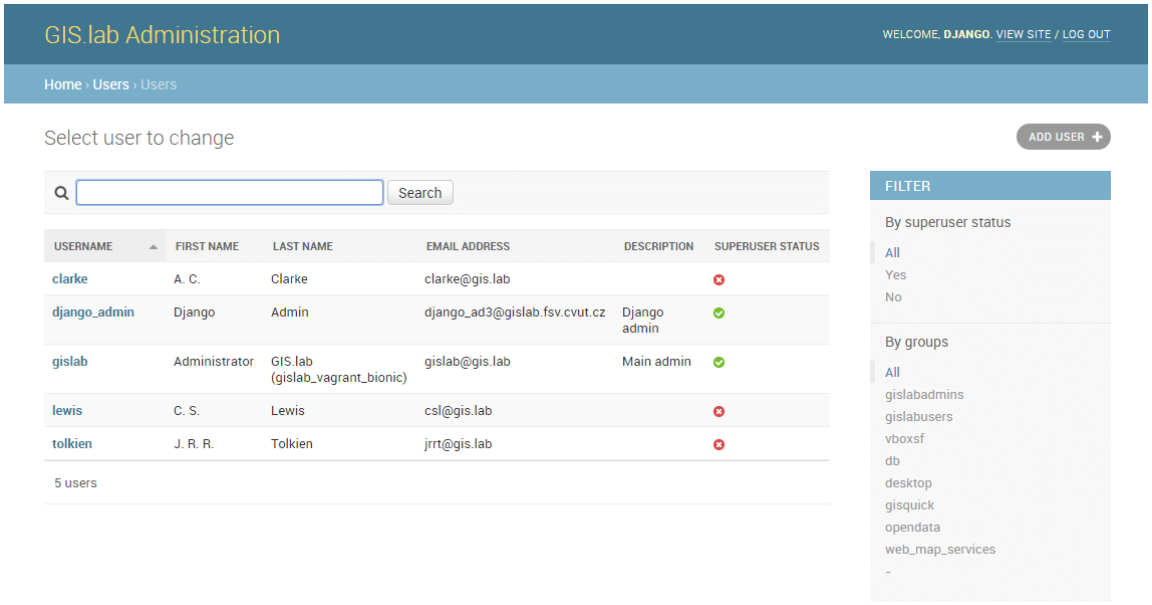


Figure A.9: Admin console - users (zdroj: Tereza Kulovaná)

On user details page you can change user's personal info or group memberships. For changing password, you need to click on this form link in the text and you will be redirected to correct page. To move groups between Available groups and Chosen groups, highlight selected group(s) and click on right/left arrow. You can move all groups in the same time by clicking Choose all/Remove all. Push SAVE button in the bottom right corner to reflect the changes.

You can delete user from database by pushing Delete button in the bottom left corner. The decision has to be confirmed.

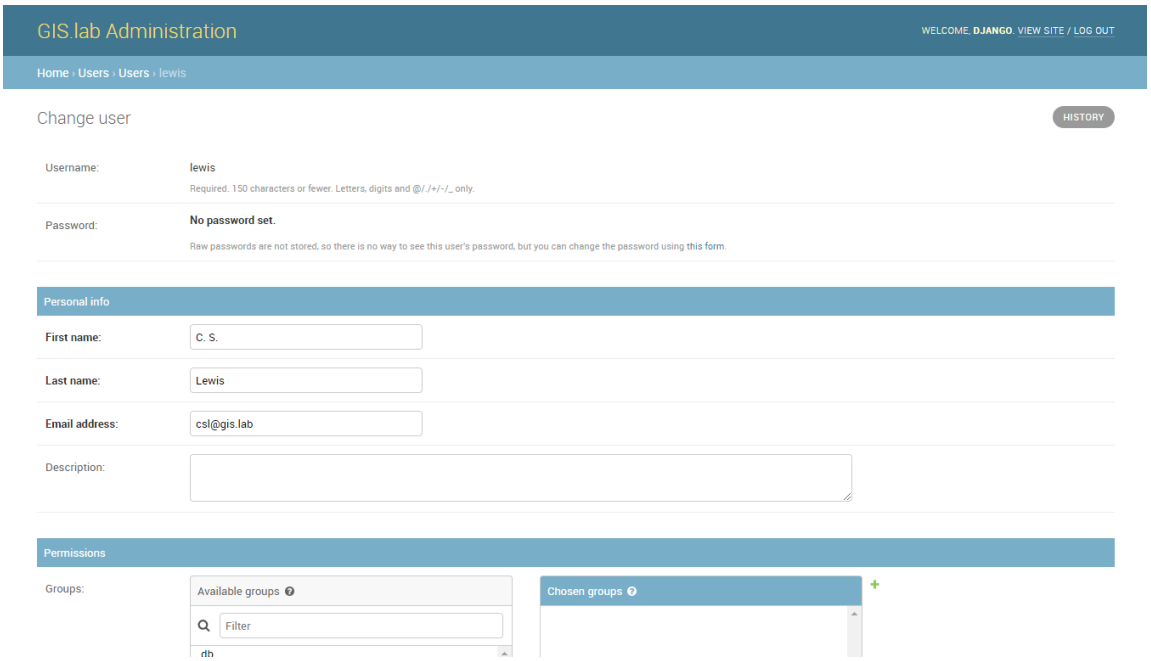


Figure A.10: Admin console - user details 1/2 (zdroj: Tereza Kulovaná)

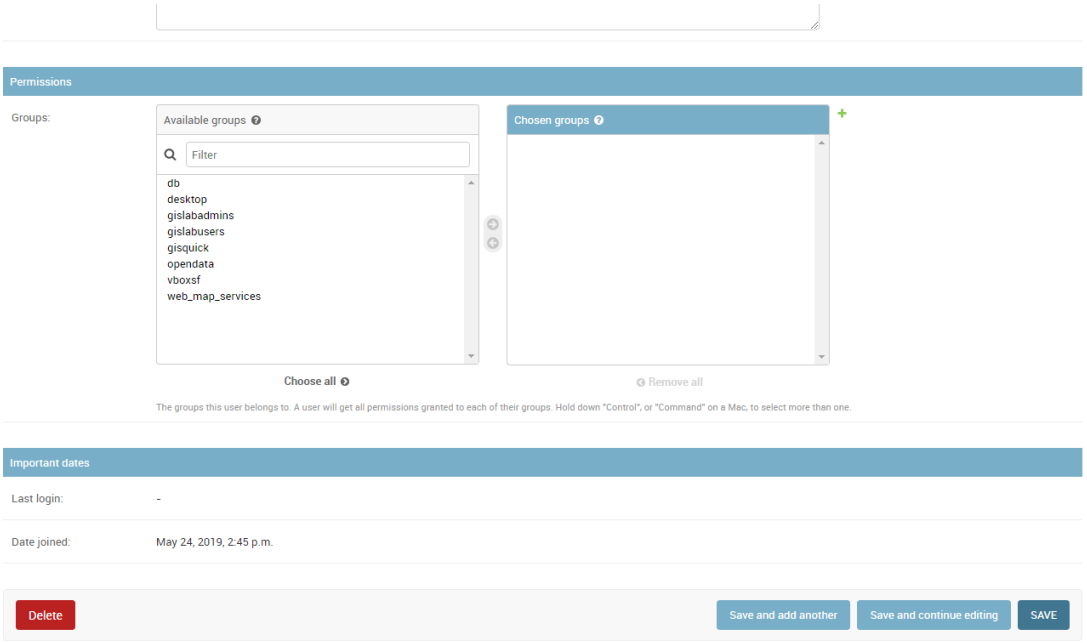


Figure A.11: Admin console - user details 2/2 (zdroj: Tereza Kulovaná)

Write new password twice for verification and push the CHANGE button. In case of a successful change, user's credentials are updated.

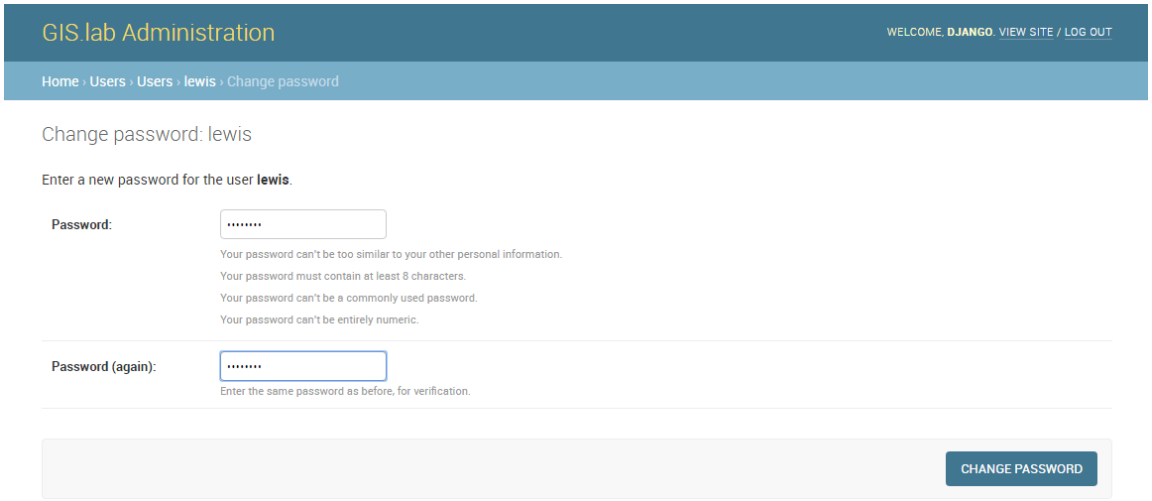


Figure A.12: Admin console - change password (zdroj: Tereza Kulovaná)

Username, first name, last name, email address and password are mandatory fields for successful registration, *description* is optional. For creating new user push the SAVE button. If the form is not valid, the error, with information of what went wrong, is displayed. In case of a successful validation, you are redirected to page with users list.

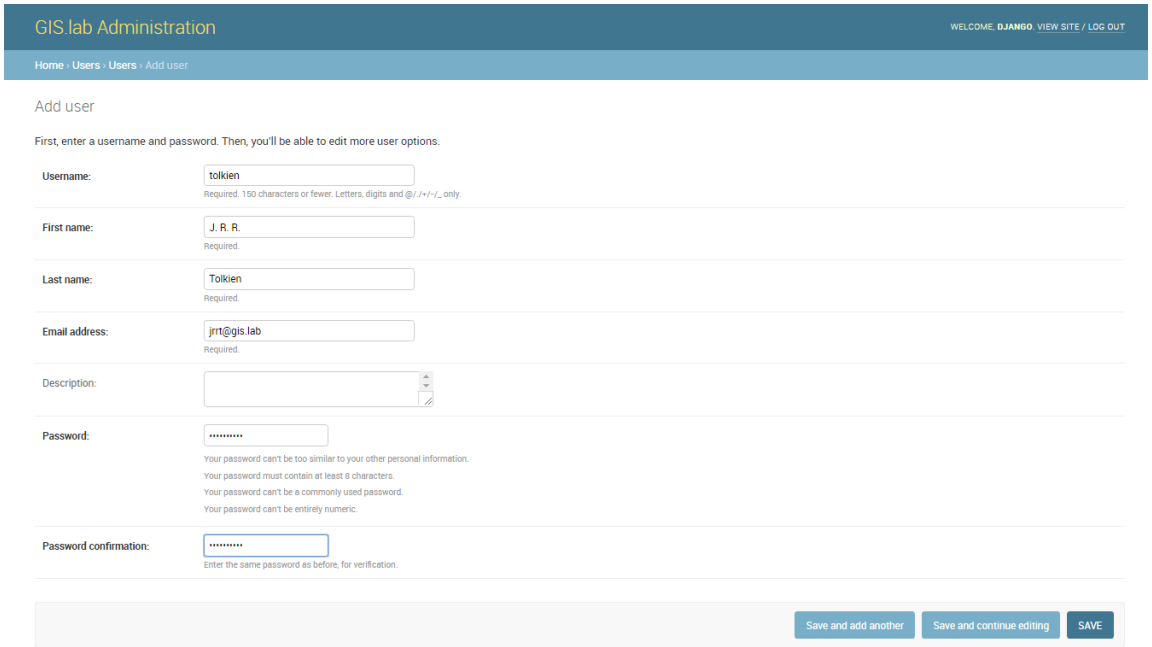


Figure A.13: Admin console - create new user (zdroj: Tereza Kulovaná)

Group list displays names of all groups (roles) in database. New group can be added by ADD GROUP button in the upper right corner. You can delete group by clicking on its name.

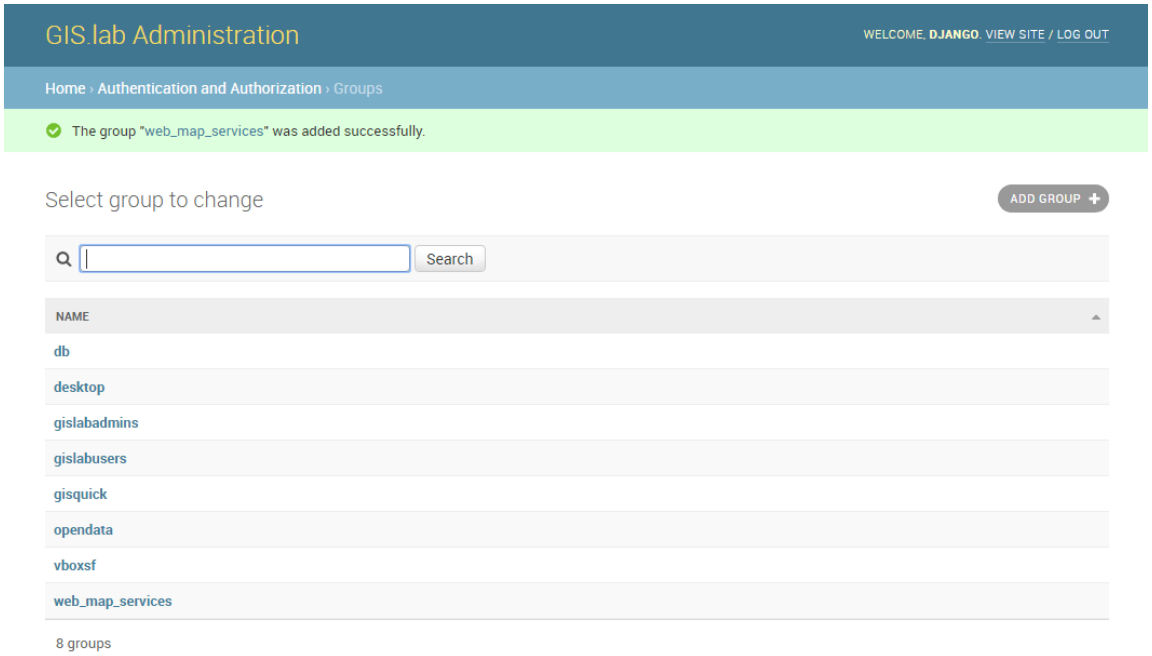


Figure A.14: Admin console - groups (zdroj: Tereza Kulovaná)

On group details page you can delete group from database by pushing Delete button in the bottom left corner. The decision has to be confirmed.

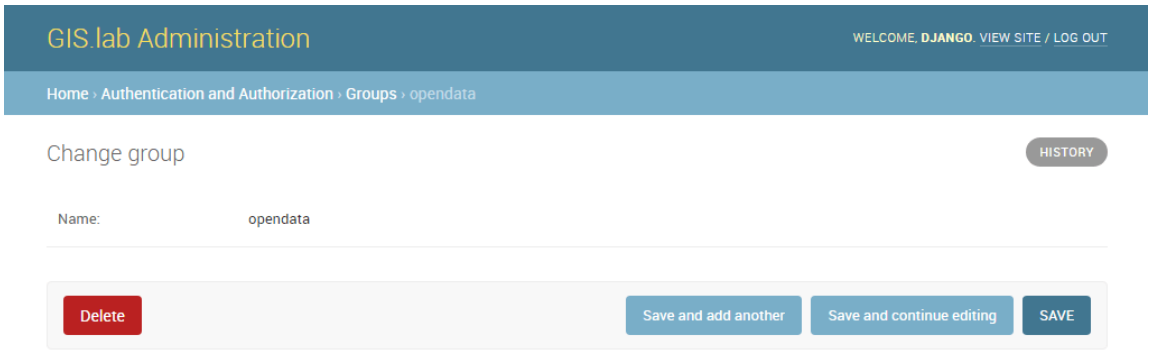


Figure A.15: Admin console - group details (zdroj: Tereza Kulovaná)

Group object has only one atribute, *name*. For creating new group push the SAVE button. If the group already exists, the error is displayed. In case of a successful validation, you are redirected to the page with list of groups.

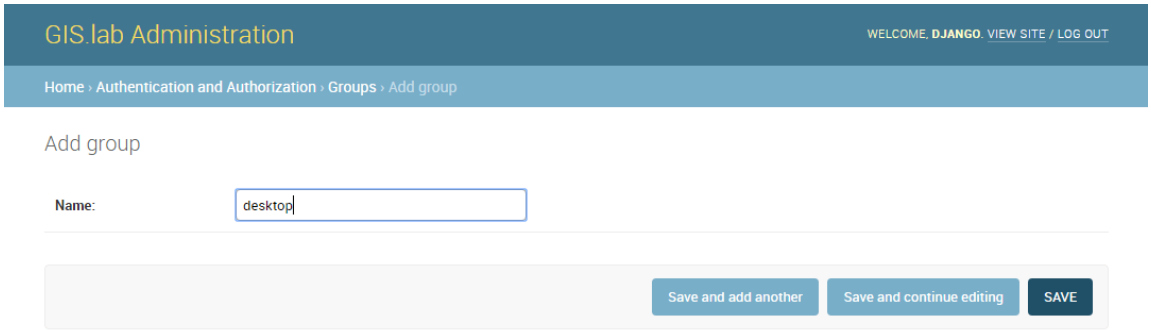


Figure A.16: Admin console - create new group (zdroj: Tereza Kulovaná)

Logout is available by LOG OUT link in the upper right corner. Link VIEW SITE will redirect you to user console.

B Obsah CD

.

assignment	zadání práce
src	zdrojový kód
text	text práce ve formátu PDF