Masarykova univerzita
Fakulta informatiky

Webový rezervační systém laboratoře LEMMA

Diplomová práce

Dalibor Jelínek

Brno, podzim 2021

Masarykova univerzita
Fakulta informatiky

Webový rezervační systém laboratoře LEMMA

Diplomová práce

Dalibor Jelínek

Brno, podzim 2021

Na tomto místě se v tištěné práci nachází oficiální podepsané zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Dalibor Jelínek

Vedoucí práce: Doc. RNDr. Petr Sojka, Ph.D.

Poděkování

Rád bych zde upřímně poděkoval vedoucímu práce doc. RNDr. Petru Sojkovi, Ph.D za velmi vstřícný přístup, trpělivost a cenné rady. Dále bych rád poděkoval celé své rodině, přítelkyni a kolegům z Coworkers.ai za veškerou podporu a pochopení při psaní této práce.

Shrnutí

Tato diplomová práce se zabývá návrhem a implementací rezervačního systému audiovizuální techniky laboratoře LEMMA s důrazem na UX a udržitelnost. V rámci práce vznikla webová aplikace postavená na technologiích Vue.js, Django a Docker podporující automatické nasazení pomocí GitLab pipelines. Textová část popisuje důkladnou analýzu procesů, požadavků a případů užití. Dále dokumentuje vybrané problémy řešené v rámci implementace, postup nasazení na produkční server a uvedení systému do provozu.

Klíčová slova

Python, JavaScript, LEMMA, Vue.Js, Django, REST, GitLab, Docker, Continuous Delivery, SPA, UX

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Rezervační systém je obecně tisíckrát řešené a vyřešené téma. V současné době není problém najít mnoho existujících nástrojů určených k rezervování zdrojů (1, 2, 3). Přesto i dnes existují situace, kdy je adaptace existujících řešení nákladnější nebo komplikovanější, než vývoj vlastního systému. Jedním takovým případem je rezervační systém audiovizuální techniky laboratoře LEMMA FI (Laboratoř Elektronických a MultiMediálních Aplikací Fakulty Informatiky) (3). LEMMA se zabývá zpracováním a publikací rozsáhlých textových a multimediálních kolekcí dat včetně produkce a postprodukce filmů. Zejména pro účely produkce filmových děl laboratoř disponuje kolekcí techniky, kterou propůjčuje na základě požadavků konkrétních projektů. Projekty typicky vznikají zejména v souvislosti s předměty zabývajícími se filmovou tvorbou vyučovanými na Fakultě informatiky Masarykovy univerzity (FI MU). Z toho plyne kumulace požadavku v určitém časovém úseku a také množství nových uživatelů každý rok.

Pro uspokojivé naplnění těchto požadavků je nezbytné poskytnout uživatelům možnost standardizovaného přístupu ke zdrojům takovým způsobem, který bude intuitivní a přímočarý a bude eliminovat konflikty výpůjček, čímž umožní efektivní plánování projektů. Zároveň je také nutné zajistit kontrolu nad vypůjčenými zdroji a jejich stavem.

Tuto úlohu dlouhodobě plní již existující rezervační systém, který v různých verzích a úpravách doprovází laboratoř LEMMA téměř od jejího vzniku v roce 1999. Tento systém byl vyvíjen a udržován zejména studenty Fakulty informatiky formou závěrečných prací. Za svou bezmála dvacetiletou existenci několikrát změnil svou podobu, nicméně jádro systému zůstává poslední dekádu stejné, což s sebou nese nejen z hlediska udržitelnosti nemalé komplikace. Současný systém je také úzce provázaný s videoarchivem, který nyní plní funkci archivace a prezentace video výstupů projektů. Těsná vazba mezi těmito systémy a skutečnost, že i společně s webovou prezentací laboratoře sdílí jeden produkční server, komplikuje údržbu a další vývoj.

Cílem této práce je analyzovat funkce a nedostatky současného systému, zmapovat procesy související s půjčováním audiovizuální techniky a na základě výsledků analýzy implementovat nový rezer-

vační systém za pomocí moderních technologií a postupů. Výstupem práce bude webová aplikace skládající se z klientské části formou SPA (single-page application) a serverové části s REST rozhraním. Pro zajištění snadných budoucích úprav a údržby systému budou obě části navrženy pro běh a vývoj v samostatných kontejnerech pomocí nástroje Docker a proces nasazení nových verzí systému bude maximálně automatizovaný.

Teoretická část práce je rozdělena do 4 kapitol, přičemž první se zabývá analýzou současných požadavků, návrhem architektury systému a návrhem uživatelského rozhraní. Druhá kapitola je věnována výběru technologií pro vývoj nového systému a dokumentuje vybrané oblasti vývoje a řešené problémy. Třetí kapitola popisuje konfiguraci automatického nasazení systému a migraci dat z předchozího systému. Závěrečná kapitola pak vyhodnocuje splnění nastavených požadavků a cílů a nastiňuje možností budoucího vývoje a údržby tohoto systému.

2 Analýza

2.1 Historie dosavadních řešení a současný stav

Potřeba efektivního sdílení prostředků a kontroly jejich využívání přirozeně vede k nasazení systému na alokaci těchto zdrojů. Nejinak je tomu v případe laboratoře LEMMA. Po prvních pokusech využití systému vyvinutého Jiřím Filipovičem pro účely laboratoře SITOLA v roce 2006 se první verze vlastního rezervačního systému pro účely laboratoře LEMMA ujala trojice autorů Jan Řezník, Tomáš Bukal a Vojtěch Vild (4). Ti vytvořili webovou aplikaci postavenou na technologiích PHP a MySQL (5).

Na jejich práci navázal v roce 2008 Pavel Jelínek se svou bakalářskou prací pod názvem Rozšíření a údržba rezervačního systému laboratoře LEMMA, v rámci které původní systém rozšířil o možnosti lepší kontroly nad využíváním zdrojů ze strany uživatelů a získávaní dat pro účely vykazování činnosti laboratoře. Oproti předchozí verzi umožňovala tato rozšíření rozlišovat, zda je vytvořená rezervace vyzvednuta, případně zda jsou zdroje po skončení rezervace vráceny. Dále poskytla možnost vytvářet kolekce zdrojů, které mohly být rezervovány jako jedna položka, detekci nadměrných rezervací, prohlížení historie rezervací a v neposlední řadě zahrnovala úpravy uživatelského rozhraní pro lepší uživatelský zážitek. Tato verze byla postavena na robustnější platformě za využití technologií Java, Apache, Tomcat a PostgreSQL (5).

Dalšího rozšíření se systém dočkal o dva roky později v rámci bakalářské práce Radima Urbáška. Ten k rezervačnímu systému doplnil modul webového video archivu, jenž umožnil jednotným způsobem dokládat a prezentovat audiovizuální díla, která za pomocí techniky zapůjčené prostřednictvím rezervací vznikala. Tím jednak přispěl k prezentační úrovni laboratoře, ale zároveň i rozšířil kontrolní potenciál rezervačního systému, jelikož díky tomuto nástroji bylo možné snadno dohledat, k jakému účelu byla zapůjčená technika reálně využita. Práce byla oceněna cenou děkana za vynikající závěrečnou práci. Pro vývoj tohoto modulu byly využity technologie Java EE, PostgreSQL, Stripes, GlassFish a Shiboleth a mnoho dalších nástrojů pro zpracování audia a videa (6).

Na Urbáškovu práci navázal v roce 2015 Peter Rozum, který svou diplomovou práci pojal jako analýzu problémů, které provází vývoj rozsáhlejších informačních systémů, zejména pak pokud tento vývoj probíhá v iteracích v rámci závěrečných prací. Výsledky jeho práce poukázaly na kritický nedostatek dokumentace funkcí i kódu a problémy způsobené absencí systematického přístupu k odstraňování chyb a implementaci nových funkcí. Kromě rozšíření funkcí systému tak byl zaveden i SVN systém pro verzování kódu a webový systém Trac pro vedení dokumentace a mapování chyb, změn a požadavků (3).

Webové technologie se však posouvají extrémním tempem, a tak systém opět přestával vyhovovat současným požadavkům. Aplikaci chyběla například optimalizace pro zobrazení na mobilních zařízeních. Klíčovou roli hrál Adobe Flash player, jehož oficiální podpora již skončila jak ze strany vývojářů, tak ze strany většiny moderních prohlížečů (7), a v neposlední řadě působil vzhled nejednotným a neaktuálním dojmem (8). Proto vznikla diplomová práce Tomáše Kovaříka, jejímž cílem bylo navrhnout a implementovat nad systémem jednotné uživatelské rozhraní, které odpovídá současnému standardu. Kovařík zvolil pro implementaci vizuálního stylu framework Masarykovy univerzity. Převlékl tak systém do barev FI a sjednotil vizuální prvky. Svou práci však stavěl na logické vrstvě předchozí verze Petera Rozuma, a tak optimalizace uživatelského rozhraní ke zvýšení uživatelského zážitku vedla jen částečně. Tato verze je aktuálně nasazena a využívána laboratoří.

Rezervační systém v současné verzi vykazuje řadu funkčních nedostatků a dostatečně nereflektuje procesy, v rámci kterých je využíván. Jedním z příkladu je proces registrace, kdy je uživatel vyzván k zadání projektu, kvůli kterému se chce rezervovat. Tento projekt však v době rezervace nemůže existovat, jelikož pro jeho vytvoření je potřeba registrace. Projekt a důvod registrace se pak navíc stanou součástí uživatelova profilu a ten je může kdykoliv po registraci změnit, což z procesního hlediska také nedává žádný smysl. Běžnou praxí také je, že si uživatelé techniku předávají mezi sebou. Systém však neumožňuje tento transfer evidovat. Tuto funkcionalitu nahrazuje papírový formulář, který předání stvrzuje. Tím ale ztrácí systém přehled o tom, kdo techniku reálně využívá, a není snadné dohledat historii skutečných rezervací.

2.2 Analýza požadavků

Analýza požadavků je do značné míry zjednodušena tím, že se jedná o reimplementaci dlouhodobě fungujícího řešení. Původní systém však za dobu svého vývoje prošel řadou změn a nasbíral mnoho funkcionalit, které se například v současnosti nepoužívají nebo nejsou z hlediska uživatele na první pohled zřejmé a neexistuje k nim odpovídající dokumentace. Výsledky prezentované v této kapitole jsou tak z části výstupem reverzního inženýrství a z části formulací nových požadavků současných uživatelů.

2.2.1 Slovní popis požadavků

Laboratoř LEMMA disponuje audiovizuální technikou, kterou propůjčuje pro účely výuky studentům a pracovníkům Masarykovy univerzity. Cílem je vytvořit webový rezervační systém, který by jeho uživatelům umožnil seznámit se s technikou, která je k dispozici, a rezervovat ji na konkrétní časový interval.

Kontrola využití zdrojů

Cílem systému je též kontrola využívání zdrojů laboratoře. Oprávnění k rezervování zdrojů je udělováno na základě žádosti, kterou je možné odeslat prostřednictvím systému (žádost typicky souvisí se zápisem některého z předmětů vyučovaných pod hlavičkou LEMMA). Žádost zahrnuje důvod a její prerekvizitou jsou řádně vyplněné kontaktní údaje v uživatelském profilu. Žádost je schvalována administrátorem systému. Po schválení požadavku je uživateli zpřístupněna základní sada zdrojů, které může rezervovat. Laboratoř disponuje také nákladnější technikou, pro jejíž vypůjčení je nutné splnit další kritéria. Rezervační systém bude zahrnovat mechanismus ke kategorizaci techniky a zpřístupňování konkrétních kategorií uživatelům, kteří splňují daná kritéria. Dalším kontrolním prvkem je výpočet nákladnosti celkové rezervace s ohledem na cenu rezervované techniky a trvání dané rezervace. U rezervací, jejichž nákladnost překročí stanovenou mez, bude vyžadováno schválení administrátorem. Výpočet nákladnosti je také využit pro výpočet penále při nedodržení termínu vrácení.

Vytváření projektů

Pro efektivní mapování využití zdrojů je nutné každou rezervaci přiřadit ke konkrétnímu projektu. Projekty mohou zakládat uživatelé. Zakladatel projektu vyplní název a popis projektu a může označit další uživatele systému, kteří na tento projekt smí vytvářet rezervace. Pro přehlednost jsou tyto projekty třízeny do skupin. Skupiny projektů zakládá administrátor systému.

Mapování dostupnosti výdejáře

Fyzicky předání a převzetí zdrojů provádí takzvaný výdejář – zaměstnanec univerzity. Systém umožní uživateli ověřit pracovní dobu výdejáře nebo jeho nedostupnost. Výdejář bude mít k dispozici rozhraní pro zadání a úpravy umístění své pracovny, své pracovní doby a případné plánované nepřítomnosti. Systém uživateli neumožní nastavit termín převzetí či vrácení techniky v době výdejářovy nepřítomnosti.

Podpora převzetí a vrácení techniky

Pokud je rezervace schválena, systém uživateli vygeneruje protokol o převzetí techniky, který obsahuje seznam techniky a podmínky převzetí. Výdejář připraví zdroje k převzetí na stanovený datum a poté co od uživatele získá podepsaný protokol o převzetí, předá zdroje uživateli a potvrdí v systému předání techniky. Pokud se uživatel nedostaví ve stanovený termín začátku rezervace, je pravidelně upozorňován emailem, aby rezervaci buď zrušil a uvolnil tak alokované zdroje ostatním zájemcům, nebo si rezervované položky vyzvedl.

Pokud uživatel část zdrojů přestane používat již před koncem rezervace, může je vrátit kdykoliv v přítomnosti výdejáře. Systém pak vrácené položky zpřístupní dalším zájemcům. Pokud uživatel zdroje nestíhá vrátit ve stanoveném termínu, může rezervaci prodloužit. Ovšem pouze za předpokladu, že některé zdroje v daném termínu nejsou v konfliktu s některou z nadcházejících rezervací. Na prodlužování je aplikován stejný princip schvalování jako při rezervaci. To znamená, že pokud rezervace s prodloužením přesáhne stanovenou hodnotu, je potřeba toto prodloužení schválit administrátorem.

Pokud výdejář zadá nepřítomnost až po vytvoření rezervace a datum převzetí či vrácení je v kolizi s nepřítomností, výdejář je upo-

zorněn. Pokud je nepřítomnost nevyhnutelná, autor rezervace je také upozorněn mailem, aby změnil datum převzetí nebo předání na termín, kdy je výdejář přítomen. Pokud je v kolizi začátek rezervace a uživatel nezmění datum začátku do 3 dnů, nebo do začátku rezervace, tato je stornována. Pokud je v kolizi konec rezervace, je uživatel taktéž upozorněn a muže změnit datum předání. Pokud tak neučiní, je tento termín automaticky posunut na nejbližší termín, kdy je výdejář k dispozici.

Při převzetí techniky vždy výdejář zkontroluje stav techniky a případné poruchy či chybějící položky zaznamená do systému.

Transfer rezervace

Systém umožní také transfer rezervace na jinou osobu a jiný projekt. Pokud má uživatel vypůjčenou techniku, kterou již nevyužívá, ale datum plánovaného vrácení dosud nenastal, může tuto techniku po zbývající dobu rezervace převést na jiného uživatele, který má o danou techniku zájem, bez účasti výdejáře. V takovém případě přebírá nový držitel techniky zodpovědnost za vypůjčenou techniku a je povinen podobně jako výdejář zaznamenat jakákoliv poškození, která odhalí při převzetí. Původní držitel musí potvrdit, že se záznamem poškození souhlasí. Poté může být technika předána. V takovém případě je rezervace z pohledu původního držitele označena jako vrácená a pro nového držitele je vytvořena nová rezervace s datem navrácení shodným s původní rezervací. Poté může nový držitel s rezervací zacházet jako s jakoukoliv jinou svou rezervací. Například ji může prodloužit. Systém neumožní převést zdroje, pokud příjemce nedisponuje oprávněním k jejich rezervaci.

2.2.2 Případy užití

Tato část si klade za cíl identifikovat v systému uživatelské role a popsat jednotlivé případy užití systému. Následující text poskytuje detailnější popis z pohledu aktérů systému k diagramu zachyceném na obrázku 2.1 na straně 14. Původní rezervační systém rozlišoval pětici různých rolí. Jednalo se o správce systému, výdejáře, uživatele s oprávněním výpůjčky, registrovaného uživatele a autentizovaného uživatele. Během revize procesů, kterých je rezervační systém součástí,

však bylo zjištěno, že registrovaný uživatel a autentizovaný uživatel jsou jen určitá stádia uživatele s oprávněním výpůjčky. Návrh tak byl zjednodušen na tři uživatelské role.

Běžný uživatel

Přihlášení do systému probíhá prostřednictvím jednotného přihlášení Masarykovy univerzity. Přihlásit se tak může kdokoliv, kdo má platný účet v informačním systému Masarykovy univerzity a není třeba žádné další registrace. Kdokoliv se přihlásí, je považován za běžného uživatele. Pro to, aby si však takový uživatel mohl zarezervovat zdroje, musí k tomu získat oprávnění od autority laboratoře. Tato oprávnění mají více úrovní a jsou vydávána na omezenou dobu. Podmínkou schválení žádosti je, aby měl uživatel vyplněné osobní údaje ve svém profilu. Jakmile má schválené oprávnění, jeho nejčastějším případem užití se stává vytvoření rezervace. Běžný uživatel také může vytvářet a upravovat projekty a přiřazovat k těmto projektům další uživatele s oprávněním rezervace. Dále může procházet všechny plánované a uplynulé rezervace v systému, zažádat o transfer rezervace jiného uživatele nebo přijmout či zamítnout takový požadavek od jiného uživatele. S transferem souvisí také vyplnění protokolu o převzetí techniky, ve kterém příjemce zaznamená případná nalezená poškození techniky.

Výdejář

Výdejář je uživatel, který zajišťuje vypůjčení techniky a její zpětné převzetí po fyzické stránce. Pokud uživatel vytvoří rezervaci a ta je schválena, výdejář je upozorněn e-mailem, ve stanoveném termínu rezervované zdroje předá a předání potvrdí v systému. Před koncem rezervace výdejář přebírá zdroje zpět a provádí kontrolu jejich stavu. Případná poškození zadává do systému. Zdroje mohou být vráceny všechny najednou, nebo po částech. Výdejář má dále za úkol informovat uživatele o době, ve které je k dispozici pro přebírání či vydávání techniky prostřednictvím zadání příslušných dat do systému.

Administrátor

Administrátor je zodpovědný za přidávání a úpravu zdrojů v systému. Dále může přiřadit jinému uživateli roli výdejáře případně administrátora. Dále je jeho úkolem schvalování rezervací. Pokud nákladnost rezervace překročí stanovenou mez, je administrátor upozorněn e-mailem a je požádán aby tuto rezervaci v systému schválil. Administrátor také nastavuje systémové proměnné jako například kritéria pro nutnost schválení rezervace nebo texty automatických oznámení systému. Dále může vytvářet pojmenované složky pro projekty (tzv. skupiny), do kterých uživatelé své projekty při jejich vytvoření zařazují. Dalším úkolem administrátora je definovat kategorie oprávnění rezervace. Každý zdroj v systému patří do jedné z těchto kategorií.

2.2.3 Akceptační kritéria

Tabulka 2.1 zachycuje na základě případů užití zobrazených v diagramu na obrázku 2.1 na straně 14 akceptační kritéria. Ta slouží k upřesnění jasných požadavků na systém a také jsou základem pro testování.

Tabulka 2.1: Akceptační kritéria

ID	Požadavek	Akceptační kritérium
UC1	Editace osobních údajů	Uživatel může zadat do systém a případně průběžně aktualizovat osobní údaje, které jsou nezbytné ke schválení požadavku na oprávnění rezervace.
UC2	Žádosti o oprávnění k rezervaci	Uživatel může podat žádost o oprávnění k rezervaci. Vybírá si přitom úroveň oprávnění, o kterou chce zažádat, a uvádí důvod žádosti. O výsledku je informován e-mailem.
pokračování na další stránce		

Tabulka 2.1 – pokračování z předchozí stránky

ID	Požadavek	Akceptační kritérium
UC3	Procházení zdrojů	Uživatel muže zobrazit zdroje, které je možné zarezervovat. Systém poskytne zobrazení seznamu a detailu s popisem zdroje a fotografií. Zdroje bude možné filtrovat podle kategorie oprávnění, dostupnosti, výdejáře a podle značek (např. audio, video, DSLR).
UC4	Správa projektů	Uživatel může v systému zakládat projekty. Projekt má svůj název, popis a tvůrčí tým. Členy týmu může přidávat či odebírat autor projektu. Členové týmu mohou na daný projekt rezervovat zdroje. Po ukončení práce na projektu je autor povinen ho v systému uzavřít.
UC5	Vytvoření rezervace	Registrovaný uživatel může vytvořit rezervaci. Rezervovat může zdroje, které jsou v daném termínu dostupné a které spadají do odpovídající kategorie oprávnění. Rezervace je vždy zařazena pod existující projekt. Datum převzetí či vrácení techniky lze stanovit pouze na termín, kdy je výdejář dostupný a technika není rezervována někým jiným. Pokud nákladnost rezervace překročí stanovenou mez, je vyžadováno schválení.
UC6	Procházení rezervací	Uživatel může zobrazit uplynulé a nadcházející rezervace jak ve formě tabulky, tak ve formě kalendáře. U každé rezervace je možné zobrazit seznam rezervovaných zdrojů a data o jejich vyzvednutí a vrácení.

pokračování na další stránce

Tabulka 2.1 – pokračování z předchozí stránky

ID	Požadavek	Akceptační kritérium
UC7	Prodloužení rezervace	Autor rezervace ji může prodloužit za předpokladu, že žádný ze zdrojů není v době prodloužení již rezervován jiným uživatelem. Pokud nákladnost prodloužení překročí stanovenou mez, je na prodloužení vyžadováno schválení.
UC8	Transfer rezervace	Pokud má uživatel zájem o zdroje, které má v daném termínu rezervovány jiný uživatel, může prostřednictvím systému požádat o transfer zdrojů na svou osobu. Aktuální držitel zdrojů je systémem upozorněn a požadavek může schválit či zamítnout. V případě schválení se oba účastníci domluví na předání techniky a příjemce zkontroluje stav. Jakékoliv poškození zaznamená do systému a potvrdí převzetí techniky. Ve chvíli potvrzení vzniká nová rezervace a v původní rezervaci jsou zdroje považovány za vrácené.
UC9	Správa zdrojů	Administrátor muže přidávat zdroje, jejich popis a nahrávat do systému jejich fotografie.
UC10	Správa uživatelů	Každý uživatel je při prvním přihlášení uložen do databáze systému. Jeho výchozí role je běžný uživatel. Administrátor muže roli existujícího uživatele změnit na výdejáře případně administrátora.
UC11	Schvalování nákladných rezervací	Pokud rezervace překročí stanovená kritéria, je administrátor upozorněn a vyzván ke schválení rezervace. Rezervace může být vydávána až po schválení. Uživatel je upozorněn na výsledek schvalovacího procesu.

pokračování na další stránce

Tabulka 2.1 – pokračování z předchozí stránky

ID	Požadavek	Akceptační kritérium
UC12	Udělování oprávnění	Pokud uživatel zažádá o oprávněni k rezervaci, administrátor je upozorněn a vyzván, aby se k žádosti vyjádřil. Žádost může být schválena či zamítnuta. V obou případech může administrátor uvést důvod rozhodnutí.
UC13	Správa skupin projektů	Každý projekt musí být zařazený v příslušné skupině. Administrátor může tyto skupiny vytvářet a v případě, že daná skupina již není aktuální, ji muže skrýt.
UC14	Správa kategorií oprávnění	Každý zdroj je přiřazen do jedné z kategorií oprávnění. Administrátor tyto kategorie může vytvářet.
UC15	Dokumentace stavu zdroje	Výdejář při převzetí techniky zpět může zadat do systému poškození na technice, která vznikla v průběhu rezervace. Stejně tak může záznamy o poškození zadávat příjemce techniky při transferu, nebo její držitel v průběhu rezervace.
UC16	Převzetí zdrojů	Pokud uživatel vyzvedl u výdejáře zdroje, může je vrátit kdykoliv. Nejpozději však v termínu konce rezervace. Zdroje je možné vracet po částech. Při převzetí výdejář zkontroluje stav a případná poškození zadá do systému. Poté zdroje označí jako vrácené. Od té chvíle jsou k dispozici dalším zájemcům.
UC17	Vydávání zdrojů	Výdejář je upozorněn na blížící se termín výdeje a systém poskytne rozhraní pro potvrzení vydání, zrušení rezervace případně zadávání poškození, které odhalil příjemce zdrojů při kontrole.

pokračování na další stránce

Tabulka 2.1 – pokračování z předchozí stránky

ID	Požadavek	Akceptační kritérium
UC18	Správa dostupnosti výdejáře	Systém výdejáři poskytne rozhraní pro zadání svého pracoviště, standardní pracovní doby a případné dočasné nepřítomnosti.

2.2.4 Modelování procesů

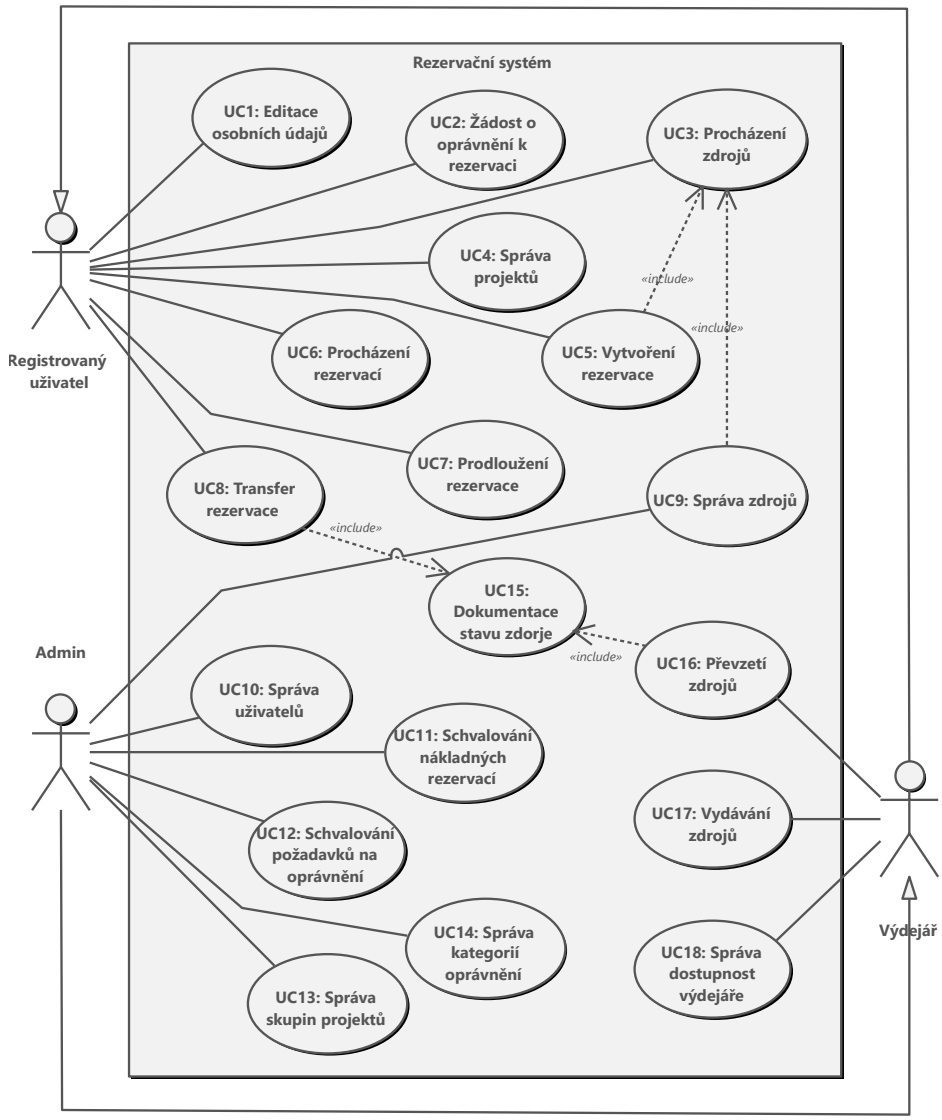
Procesy související s činností laboratoře LEMMA jsou po mnoho let neměnné a ačkoliv předchozí verze systémů byly vždy vytvářeny na míru, vykazovaly nezanedbatelné množství nedostatků, které byly způsobeny právě nedostatečnou adopcí těchto procesů. To mělo zásadní dopad především na uživatelský zážitek, kvůli čemuž v mnoha situacích docházelo i k obcházení nebo ohýbání pravidel a vlastností systému.

Z tohoto důvodu je v rámci nové implementace věnován důraz analýze a mapování procesů v laboratoři s cílem tyto procesy maximálně optimalizovat a standardizovat právě za pomocí rezervačního systému. Klíčové procesy jsou zachyceny pomocí BPMN diagramů v následující části.

Inicializace uživatele

Jak již bylo zmíněno výše v textu, do rezervačního systému má přístup každý uživatel s platnými přihlašovacími údaji do systému univerzity. K tomu, aby uživatel mohl využívat zdroje laboratoře, však předchází několik nezbytných kroků. Celý postup od prvního přihlášení do systému zachycuje diagram na obrázku 2.2 na straně 15.

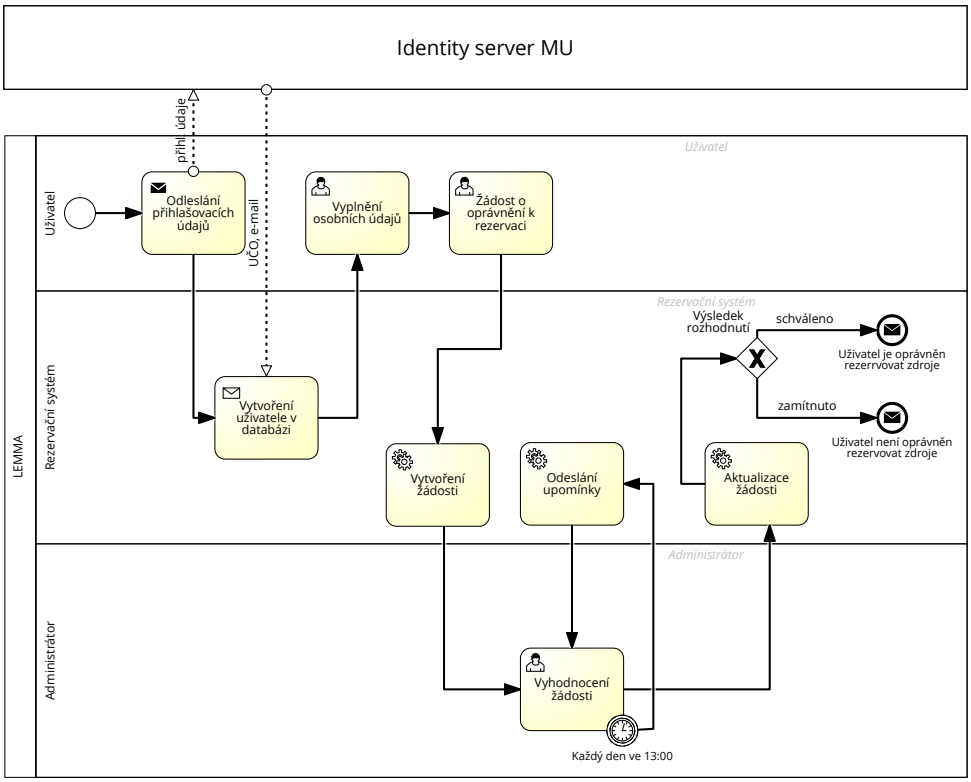
Prvním krokem při práci se systémem je vždy nezbytná autentizace prostřednictvím univerzitního identity serveru. Pokud se jedná o první přihlášení, systém vytvoří o uživateli záznam ve vlastní databázi. Informace, které jsou poskytovány z identity serveru o uživateli, však nejsou v případě rezervace dostačující. Proto pokud chce uživatel vytvářet rezervace, nejprve musí vyplnit osobní údaje a až poté může zažádat o oprávnění k rezervaci. Oprávnění schvaluje administrátor na základě kritérií stanovených pravidly laboratoře. Ten je upozorněn e-mailem při každém novém požadavku na oprávnění. Pokud poža-



Obrázek 2.1: Diagram případů užití

davek není vyřízen do následujícího dne, je administrátor periodicky upozorňován každý den v konkrétní čas, dokud se k požadavku nevyjádří. Oprávnění je vždy schvalováno na určitou dobu. Z pravidla

se jedná o jeden rok. Pokud je žádost vyřízena kladně, uživatel je informován o výsledku e-mailem a po stanovenou dobu může provádět rezervace zdrojů s příslušnou úrovní oprávnění. V opačném případě je uživatel informován o zamítnutí včetně odůvodnění rozhodnutí.



Obrázek 2.2: BPMN diagram inicializace uživatele

Vytvoření rezervace

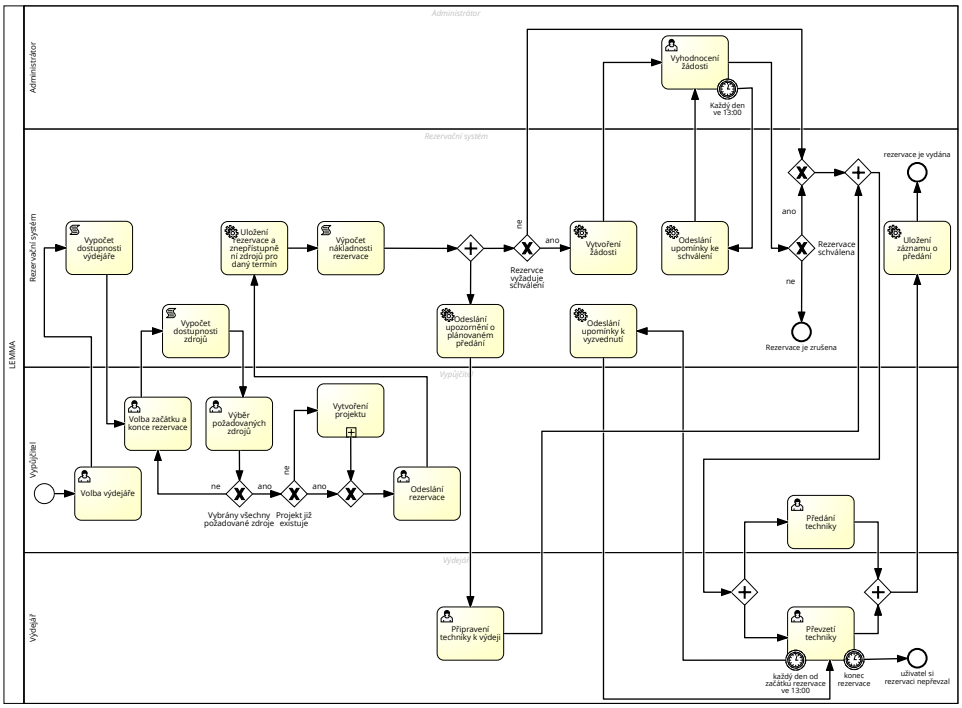
Při vytváření rezervace se jedná o kooperaci až čtyř aktérů. Jsou jimi vypůjčitel, který požaduje zdroje laboratoře, administrátor, který je vyzván ke schválení, pokud se jedná o rezervaci, která je nákladnější, než stanovený limit, výdejář, který zajišťuje předání a převzetí zdrojů a samotný rezervační systém, který mapuje jednotlivé fáze procesu a poskytuje aktérům potřebná data. V prvním kroku si uživatel vybírá výdejáře, jehož zdroje se chystá zarezervovat. Na základě jeho

pracovní doby a případné dovolené jsou pak vypočteny termíny, ve kterých je možné předání a převzetí zdrojů provést. Po výběru požadovaného termínu systém poskytne informace o tom, které zdroje jsou v tomto termínu k dispozici. Pokud v období požadované zdroje volné nejsou, může uživatel upravit data a tyto kroky opakovat, dokud nenalezne vhodný kompromis mezi termínem a požadovanými zdroji. Každá rezervace musí být přiřazena k projektu. Pokud takový projekt již existuje, vybere ho uživatel ze seznamu. V opačném případě ho může vytvořit v průběhu vytváření rezervace. Po odeslání je vypočtena nákladnost rezervace a pokud přesahuje stanovenou mez, je upozorněn administrátor, aby rezervaci schválil. Současně je informován výdejář, že je naplánována rezervace zdrojů v jeho správě. Podobně jako v případě požadavků na oprávnění je v případě nevyřízení do druhého dne administrátor periodicky upozorňován na čekající požadavky na schválení rezervací. Po schválení rezervace je možné zdroje převzít. Zpravidla k předání dochází ve stanoveném termínu při vytvoření rezervace. Po domluvě s výdejářem je však možné vyzvednout techniku s předstihem. Pokud jsou požadované zdroje k dispozici, systém toto předání umožní a označí vydané zdroje jako nedostupné pro ostatní zájemce už od okamžiku předání.

Může však dojít i k situaci že uživatel si zdroje ve stanovený termín nepřevezme. V takovém případě je periodicky upozorňován, že zdroje jsou připraveny k vydání a že si je může v pracovní době příslušného výdejáře vyzvednout, případně ať rezervaci zruší a uvolní tak zdroje dalším zájemcům. Pokud zdroje nejsou vyzvednuty do konečného data rezervace, tato je automaticky uzavřena a zdroje jsou k dispozici dalším zájemcům.

Vrácení zdrojů rezervace

Poté, co jsou zdroje úspěšně vyzvednuty, může je vypůjčitel kdykoliv v pracovní době příslušného výdejáře vrátit a to jak všechny naráz, tak i po částech. Systém vrácené zdroje průběžně zpřístupňuje dalším zájemcům, což umožňuje efektivnější využívání techniky. Během vrácení zdrojů výdejář zkontroluje jejich stav a v případě nedostatků zaznamená změny oproti předchozímu stavu do rezervačního systému. Pokud uživatel nevrátí zdroje do stanoveného termínu, je periodicky upozorňován aby techniku co nejdříve vrátil.

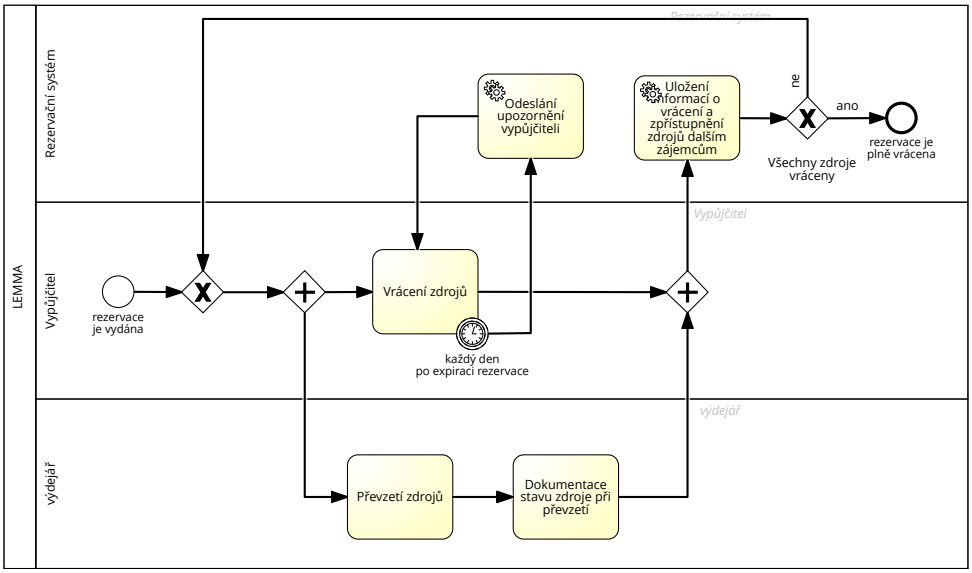


Obrázek 2.3: BPMN diagram procesu vytvoření rezervace a předání zdrojů

2.3 Design uživatelského rozhraní

Uživatelské rozhraní bylo u předchozího systému v nedávné době od základu přepracováno v rámci diplomové práce Tomáše Kovaříka. Ten využil UI frameworku poskytovaného Ústavem výpočetní techniky Masarykovy univerzity, díky čemuž systém získal vzhled korespondující s oficiálním jednotným vizuálním stylem univerzity, který byl stanoven v roce 2018 [9]. Jeho práce se však zabývala výhradně uživatelským rozhraním které bylo napojené na původní back-end, čímž aplikace zdědila řadu neduhů po svém předchůdci.

Cílem této práce je dosažení maximální uživatelské přívětivosti a s ohledem na to bylo vyhodnoceno, že využití frameworku od MU není optimální volbou, jelikož sada komponent zdaleka není tak obsáhlá a flexibilní jako u řady univerzálních open-source řešení, které jsou



Obrázek 2.4: BPMN diagram procesu vrácení zapůjčených zdrojů

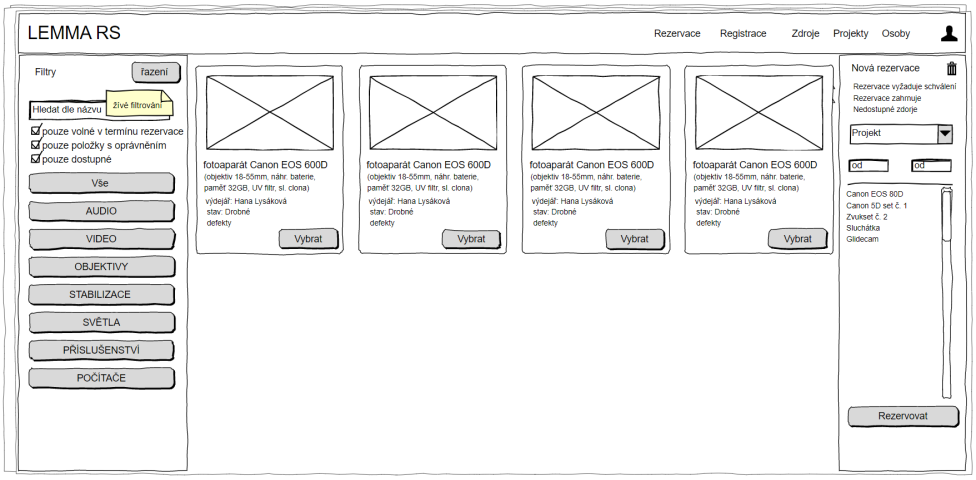
v současné době k dispozici a vývoj vlastních UI komponent v souladu s vizuálním stylem univerzity by zabral nepřiměřené množství času.

2.3.1 Material Design

Pro implementaci nového rezervačního systému byl vybrán Material Design, což je vizuální styl též označovaný jako vizuální jazyk vyvinutý společností Google v roce 2014 [10]. Ta ho od té doby používá u většiny svých aplikací a zároveň je výchozím vizuálním stylem operačního systému pro mobilní platformy Android. Díky tomu, že je současně výchozím stylem JavaScriptového frameworku Angular a knihovny UI komponent vytvořené na jeho principech jsou k dispozici i pro další frameworky dominující současnému trhu [11], je tento styl v oblasti webových a mobilních aplikací využíván velmi často. Mnoho uživatelů tak již z předchozí zkušenosti ví, jak s prvky zacházet, což opět zvyšuje uživatelský zážitek.

Material Design je postavený na metafoře reálného fyzického světa, jeho textur, odrazů světla či vrhání stínů. Kromě doporučeného rozložení, barev nebo typografie, přikládá i význam animacím a specifikuje

jejich použití pro účely dosažení intuitivnějšího uživatelského rozhraní.



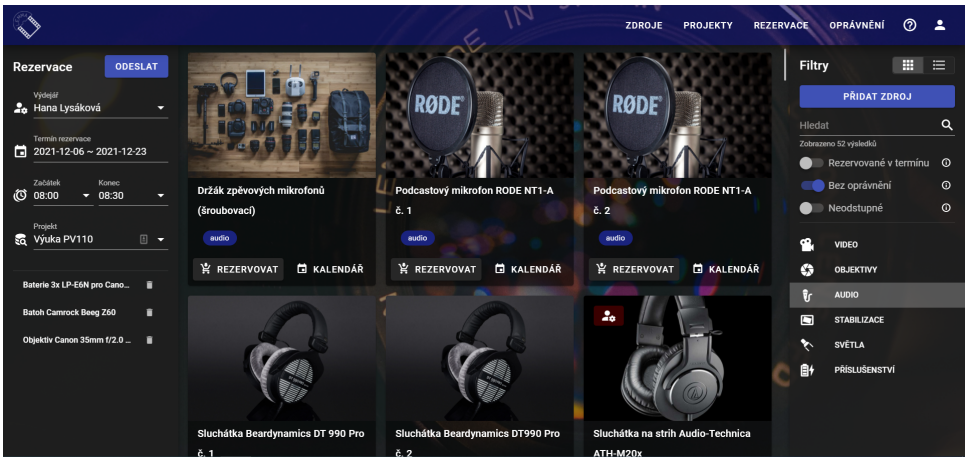
Obrázek 2.5: Wireframe UI pro vytvoření rezervace

2.3.2 Návrh pohledů

Návrh uživatelského rozhraní je výsledkem řady rozhovorů a brainstormingů s dlouholetými uživateli rezervačního systému, během kterých vznikaly různé pracovní verze. Wireframe rozhraní pro vytvoření rezervace vzniknuvší během analýzy je zachycen na obrázku 2.5. Oproti návrhu vzniklo během vývoje několik drobných změn jako je například výměna levého a pravého postranního sloupce, jelikož při uživatelském testování bylo zjištěno, že ovládací prvky, které mají být použity jako první, uživatelé očekávají většinově v levém horním rohu. Z toho důvodu bylo do horní části sloupce přesunuto i tlačítko k odeslání rezervace. Výsledný vzhled tohoto pohledu dokumentuje obrázek 2.6 na následující straně. Cílem bylo vytvořit takové rozložení ovládacích prvků, aby bylo používání intuitivní a uživatelé měli k dispozici všechny potřebné informace na první pohled.

Uživatelské rozhraní je logicky rozděleno na čtyři hlavní pohledy. Jedná se o zdroje, rezervace, projekty a oprávnění.

V závislosti na uživatelské roli se po přihlášení liší domovská stránka. Správci a výdejáři jsou přesměrování na správu rezervací.



Obrázek 2.6: Zobrazení zdrojů

Běžným uživatelům slouží jako výchozí obrazovka výpis zdrojů, ze kterého je možné vytvářet rezervace. Rezervace, oprávnění i projekty mají téměř totožný layout. Ve všech případech se jedná o tabulku, která umožňuje filtrování a řazení položek. Všechny uživatelské role mají k dispozici stejné pohledy. Administrativní role mají navíc k dispozici u jednotlivých položek v tabulkách tlačítka umožnující správu rezervací projektů a oprávnění. V případě výdejáře je pak pohled na rezervace rozšířen o flitr rezervací, které jsou u něho naplánovány. Správci mají k dispozici filtr na rezervace vyžadující schválení.

Rezervace je kromě tabulky možné zobrazit také v kalendáři. Na první pohled je tak možné zjistit, kdo má ve vybraném měsíci rezervované zdroje a ve kterých termínech je naopak všechna technika k dispozici. Aby bylo možné se v kalendáři lépe orientovat, každá rezervace má vlastní barvu. Barvy jsou automaticky generovány s ohledem na to, aby korespondovaly s celkovým vizuálním stylem systému. Generování však neprobíhá náhodně. Barva je vygenerována z názvu projektu. Díky tomu mají všechny rezervace ke stejnému projektu tutéž barvu.

Nejkomplexnějším rozhraním je stránka se zdroji, která zároveň umožňuje zadat všechny potřebné údaje pro vytvoření rezervace.

Naprostou většinu uživatelů tvoří studenti, kteří si rezervují techniku pro realizaci svých závěrečných prací. Jejich cílem je co nejrychleji

zmapovat, jaké vybavení jim laboratoř může nabídnout, případně za jakých podmínek, a sestavit z nich set potřebný pro produkci audiovizuálního díla. Klíčovou vlastností tedy bylo, aby zdroje byly zobrazeny včetně obrázků.

Jedním z problémů předchozího systému byla skutečnost, že uživatel musel nejprve vybrat zdroje, o které měl zájem, a až poté dostal k dispozici nepříliš přehlednou tabulku mapující, kdy jsou tyto zdroje k dispozici. To zpravidla vedlo k tomu, že uživatel byl nucen vybrat mnohem více zdrojů, než ve skutečnosti potřeboval, poté zjistil, co si může v daném termínu vypůjčit, a pak musel opět ručně zdroje, které v požadovaném termínu k dispozici nebyly, odebrat.

Tomuto se snaží nový systém předejít obracením procesu rezervace, který je blíže popsán v sekci 2.2.4 na straně 13. Prvním krokem je zadání časového rozpětí plánované rezervace. Jakmile systém zná termín vyzvednutí a vrácení, nabídne k rezervaci pouze zdroje, které jsou v daném rozpětí k dispozici.

Nejsou-li k dispozici, jsou označeny jako nedostupné. A tlačítko pro rezervaci je deaktivováno. Každý zdroj má navíc vlastní kalendář, ve kterém je možné zobrazit všechny jeho plánované rezervace.

Tento postup je v souladu s běžnými procesy v laboratoři, neboť termín natáčení je často v době rezervace již pevně daný, ale v případě výběru zdrojů je možné dělat více kompromisů.

Stejným způsobem jsou označeny zdroje, pokud je není možné rezervovat z jakéhokoliv jiného důvodu. Každý důvod má svou ikonu. Jednotlivé stavy jsou zachyceny na obrázku 2.7.

Obrázek 2.7: Ikony ilustrující aktuální stav zdroje

První zleva značí, že uživatel nemá na rezervaci dostatečné oprávnění. Sousední ikona znázorňuje, že je zdroj poškozen a tudíž dočasně vyřazen z provozu. Následující ikona informuje o tom, že zdroj nebyl vrácen, ačkoliv jeho rezervace již skončila. V tomto případě je zdroj možné rezervovat, ale může dojít k tomu, že v době začátku nové rezervace zdroj stále nebude vrácen. Ikona s ciferníkem značí výše zmíněnou nedostupnost z důvodu časového konfliktu s jinou rezervací

a poslední ikona značí, že tento zdroj není možné rezervovat u aktuálně zvoleného výdejáře. Z hlediska celého procesu rezervace totiž není možné do jedné rezervace zahrnout zdroje od různých výdejářů.

Ikony na první pohled nemusí být pro nové uživatele zcela intuitivní. Ostatně používání ikon na místo textů je obecně často vytýkaným prvkem Material Designu [12]. Vyjádření stavu rezervovaného předmětu ikonou však šetří mnoho prostoru a umožní tak zobrazit více zdrojů včetně všech podstatných informací na jedné obrazovce bez nutnosti rolování. Uživatelé na desktop zařízeních se mohou dozvědět význam ikon po najetí kurzorem nad ikonu, uživatelé mobilních zařízení jsou pak odkázáni na nápovědu.

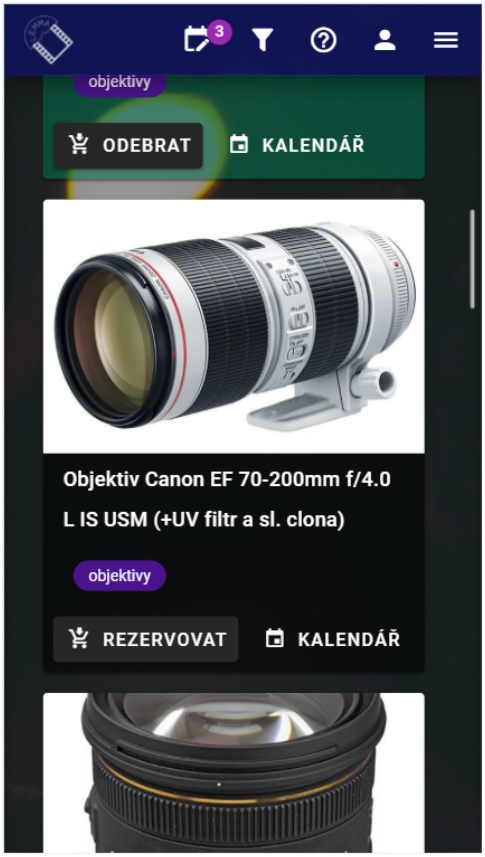
Některé zdroje je vhodné zapůjčovat dohromady, nicméně není to vždy podmínkou. Předchozí systém umožňoval vytvářet skupiny zdrojů, přičemž jednotlivé položky těchto skupin nebylo možné zapůjčovat jednotlivě. To často vedlo k situaci, že si uživatel zapůjčil celý set, i když z něj plánoval využít pouze některé položky a blokoval tak zdroje ostatním zájemcům, aniž by je využíval.

Nový systém přichází s konceptem tagů neboli štítků, přičemž ke každému zdroji je možné přiřadit neomezené množství těchto tagů, podle kterých je pak možné zdroje filtrovat. Díky tomuto řešení uživatel na první pohled ví, že pokud má zdroj přiřazený štítek, pravděpodobně souvisí s dalšími položkami a může si na základě štítku vyfiltrovat související zdroje. Tento přístup zároveň umožňuje, aby jeden zdroj byl v případě potřeby přiřazen do více skupin.

Pro dlouhodobější uživatele systému a správce, kteří mají obecný přehled o zdrojích, může být za určitých okolností zobrazení včetně obrázků nepraktické. Proto systém nabízí i zobrazení zdrojů pomocí tabulky s přizpůsobitelným řazením podle jednotlivých sloupců. Ta poskytuje větší hustotu informací a celkový pohled na zdroje v systému.

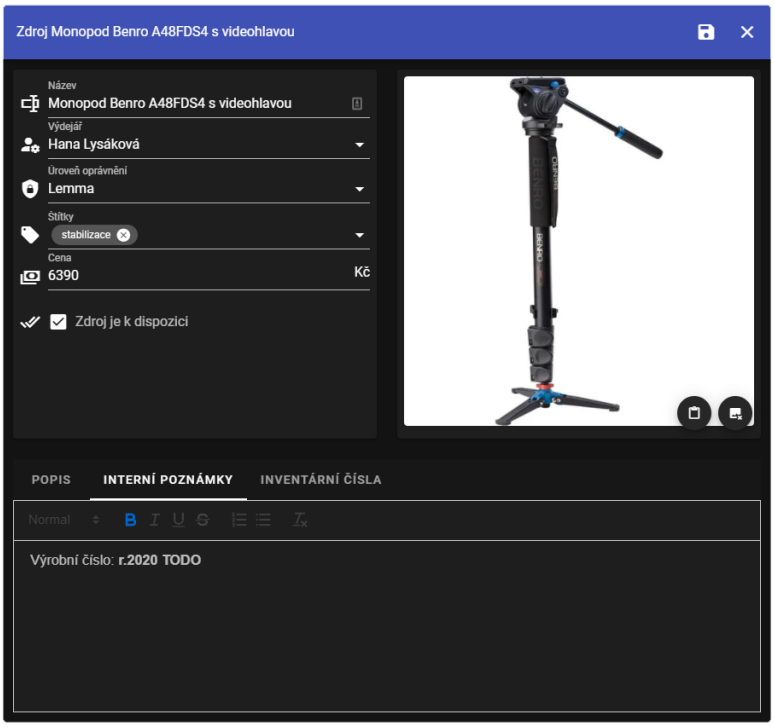
Layout stránky se zdroji byl navržen tak, aby uživatel mohl procházet a vyhledávat požadované položky a souběžně neztratil přehled o již vybraných zdrojích a dalších parametrech aktuální rezervace. Stránka je rozdělena do tří sloupců, přičemž levý postranní sloupec obsahuje veškeré informace o vznikající rezervaci, pravý postranní sloupec umožňuje nastavení filtru a vyhledávání a veškerý prostor mezi nimi je věnován samotným zdrojům ať už v podobě karet s obrázky, nebo tabulkovém zobrazení, mezi kterými může uživatel v prů-

běhu vytváření rezervace libovolně přepínat. Zejména na mobilních zařízeních však prostoru na zdroje zbylo velmi málo. Proto v rámci optimalizace pro menší obrazovky jsou od šířky 1220px níže postranní panely skryty a veškerá šířka obrazovky je věnována zdrojům, jak je vidět na obrázku 2.8. Panely je možné vysunout pomoci ikon z horní navigace. Na mobilních zařízeních jsou navíc pro vysunutí panelů podporována gesta. Podobně je také převedeno do výsuvného panelu horizontální menu.



Obrázek 2.8: Zobrazení zdrojů na mobilním zařízení

Pro veškeré úpravy nebo vytváření nových objektů v systému jsou použity modální dialogy. Dialog na vytváření a editaci zdrojů zachycený na obrázku 2.9 na následující straně zahrnuje dva WYSIWYG editory. Jeden slouží pro popis zdroje, druhý pak pro interní poznámky,



Obrázek 2.9: Modální okno pro editaci zdroje

ke kterým má přístup pouze výdejář a administrátoři. WYSIWYG editorem pro popis zdroje disponoval také předchozí systém. Bohužel ale neumožňoval upload obrázků na vlastní server. Jelikož náhledy zdrojů jsou ale pro orientaci v systémů klíčkové, snažili se tento problém správci systému řešit vkládáním externích odkazů. Často se ale stávalo, že soubory byly z externích serverů po čase smazány a systém tak byl plný nefunkčních obrázků.

Tento problém řeší nový systém vlastním úložištěm obrázků. Kromě toho umožňuje vkládat také obrázky přímo ze schránky, což je velmi efektivní způsob při získávání obsahu z internetu.

Obrázek 2.10: Kalendář rezervací

2.4 Objektově orientovaný návrh

Souběžně s návrhem uživatelského rozhraní vznikal objektový návrh systému. Oproti modelu předchozího systému se jedná o značně odlehčenou verzi. Celý proces návrhu aplikace byl silně prostoupen myšlenkou minimalismu a KISS. S vědomím, že aplikace by měla sloužit několik dalších let a že ji pravděpodobně bude udržovat, případně rozšiřovat i někdo jiný než její autor, byl kladen velký důraz na ponechání pouze objektů a závislostí, které mají skutečný přínos pro koncového uživatele. Například datový model se tak oproti předchozí verzi zmenšil přibližně na třetinu. V následující části budou popsány jednotlivé třídy, jejich význam a vztahy v rámci systému.

2.4.1 Klíčové třídy systému

Uživatel

Uživatel je v systému vytvořen automaticky, jakmile se přihlásí pomocí jednotného přihlášení MU. Rozhraní jednotného přihlášení o uživateli poskytne systému jeho jméno, e-mail a UČO. Uživatel má vždy jednu ze tří rolí definovanou atributem role. Je buď administrátor, výdejář, nebo běžný uživatel. Mezi těmito rolemi může během svojí existence různě přecházet. Proto má každý uživatel k dispozici i atributy, které jsou specifické pouze pro některou z rolí. V závislosti na zvolené roli pak uživatelské rozhraní zpřístupní pouze relevantní atributy k příslušné roli. Dále také uživatel disponuje atributem role_display, který umožní administrátorům zvolit si preferovaný způsob zobrazení uživatelského rozhraní. Důvodem tohoto řešení je, že uživatelské role se z praktického hlediska v určitých situací prolínají a pro administrátora tak může být někdy efektivnější pracovat v rozhraní běžného uživatele. Uživatelé, kteří chtějí vytvářet rezervace musí doplnit do svého profilu adresu a telefonní kontakt. V případě výdejáře je nutné doplnit místnost, pracovní dobu a případně také záznamy o plánované dovolené, které slouží pro automatické generování povolených termínů pro vyzvednutí, respektive vrácení zdrojů. Administrátoři si navíc mohou ve svém uživatelském profilu zvolit, zda chtějí dostávat upozornění o požadavcích na schválení rezervace či o požadavcích na zvýšení úrovně oprávnění.

Zdroj

Veškeré položky, které lze v systému zarezervovat, jsou zdroje. Každý zdroj má své jméno, popis, cenu a náhledový obrázek. Zdroj je možné deaktivovat, pokud je z nějakého důvodu vyřazen. V takovém případě ho nelze rezervovat. Každý zdroj je přiřazen k jednomu výdejáři, který za něj zodpovídá. Zdroj má konkrétní úroveň oprávnění, kterého musí uživatel dosáhnout, aby mohl daný zdroj zarezervovat. Pro účely správy jsou k dispozici také interní poznámky. Zdroji je též možné přiřadit libovolný počet inventárních čísel. Některé zdroje totiž sestávají z většího počtu položek s různými inventárními čísly. Jejich rezervace samostatně ale nedává smysl. Zdroj může mít také libovolný počet štítků, pomocí kterých lze související zdroje vyhledávat.

Požadavek na oprávnění

Pokud si chce uživatel zapůjčit zdroje, musí nejprve zažádat o oprávnění k rezervaci. Žádost vznikne vytvořením požadavku, který definuje kdo o oprávnění žádá, jakou úroveň požaduje a z jakého důvodu. O vytvoření požadavku je upozorněn administrátor, který požadavek označí jako schválený případně zamítnutý a doplní důvod rozhodnuti a datum expirace oprávnění.

Úroveň oprávnění

Úroveň oprávnění je jednoduchá třída mající pouze své jméno a číselně vyjádřenou úroveň. Logika úrovní oprávnění je taková, že pokud uživatel získá například úroveň 3, automaticky může rezervovat i zdroje spadající do úrovně 1 a 2.

Rezervace

Rezervace je vytvořena uživatelem, který si plánuje vypůjčit zdroje. Má stanovený datum vyzvednutí a vrácení. Drží o sobě informaci, zda byla již vyzvednuta, kdy byla vytvořena a zda je schválena. Při jejím vytvoření je vyhodnoceno, zda rezervace splňuje kritéria pro automatické schválení. Pokud rezervace kritéria splňuje, je atribut schválení automaticky nastaven na hodnotu `true` a rezervace může být kdykoliv vydána. V opačném případě má atribut hodnotu `null`,

dokud se k dané rezervaci nevyjádří administrátor, který ji buď schválí, nebo zamítne. Každá rezervace má také projekt, ke kterému je přiřazena, a výdejáře, který za ni zodpovídá. Všechny zdroje v rámci jedné rezervace tak musí patřit témuž výdejáři.

Rezervovaný zdroj

Během vytvoření rezervace je pro každý zdroj vytvořen záznam o jeho rezervaci zvlášť. Systém totiž umožňuje i částečné vrácení techniky, která byla zapůjčena v rámci jedné rezervace. Pokud je část techniky vrácena již před koncem rezervace, systém tyto položky zpřístupní dalším zájemcům. Další přípustnou variantou je, že zájemce si přijde rezervaci vyzvednout po domluvě s výdejářem ještě před datem deklarovaného začátku rezervace, a v takovém případě je nutné, aby zdroje byly považovány pro ostatní zájemce za nedostupné od chvíle vydání, nikoliv až od začátku rezervace. Z toho důvodu je pro každý rezervovaný zdroj evidován datum skutečného vyzvednutí a skutečného vrácení techniky. Tyto údaje jsou pak použity k výpočtu skutečné dostupnosti zdroje. V okamžiku vrácení je možné k rezervovanému zdroji doplnit komentář, pokud například došlo k nějakému poškození nebo problému. To umožňuje administrátorům lépe monitorovat stav zdrojů.

Projekt

Projekt je zpravidla audiovizuální dílo, k jehož realizaci je zapotřebí využití zdrojů laboratoře. Rezervace vždy vzniká k určitému projektu. Před vytvořením rezervace tedy uživatel musí nejprve vytvořit projekt. Projekt má název, popis, vlastníka, který ho vytvořil, a také libovolný počet členů, tedy dalších tvůrců, kteří mohou rezervace k tomuto projektu vytvářet. Po dokončení díla je vlastník projektu povinen označit projekt jako ukončený. Na ukončený projekt nelze vytvářet další rezervace. Každý projekt je také přiřazen do konkrétní skupiny.

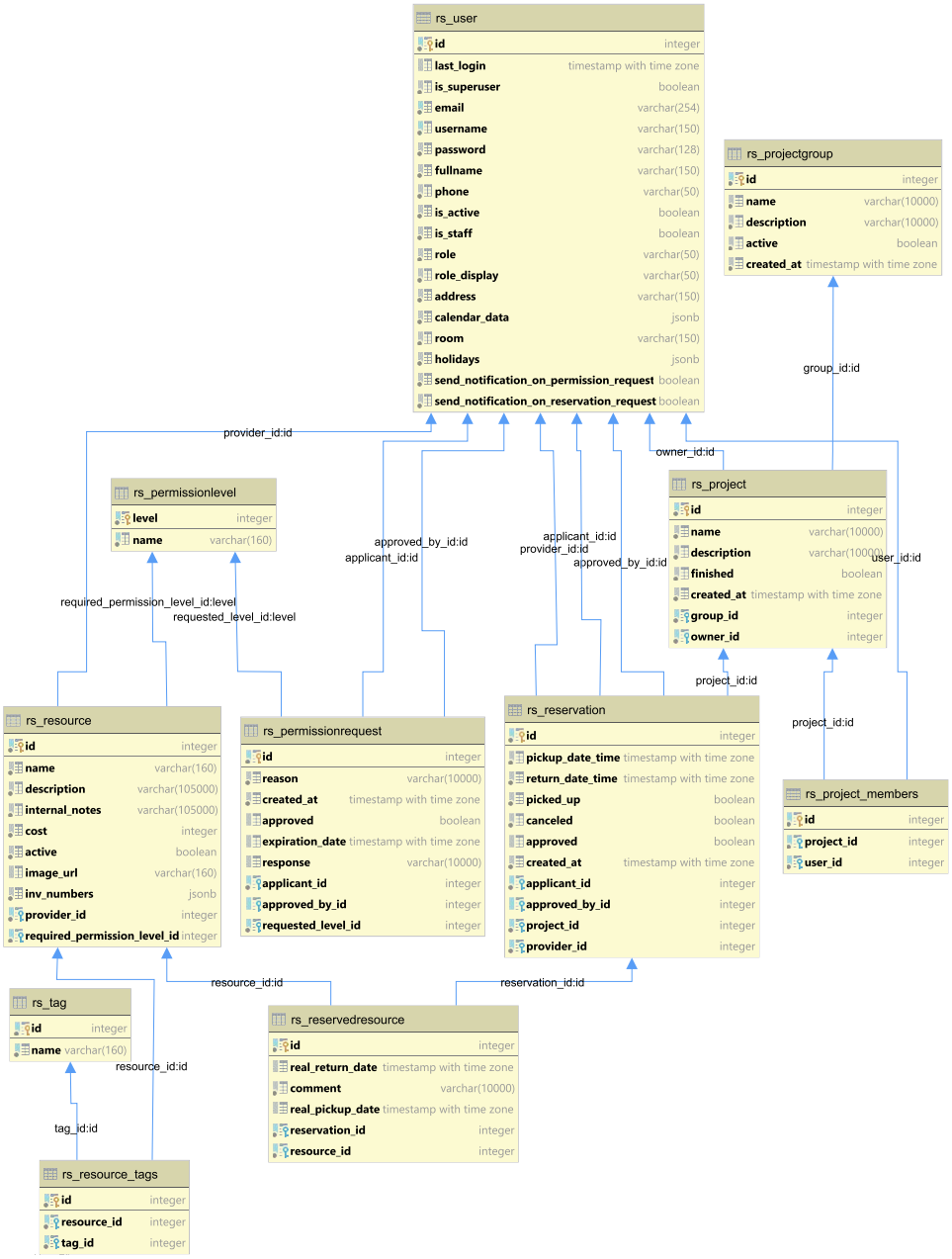
Skupina projektů

Pro lepší přehlednost jsou projekty řazeny do skupin. Skupina sdružuje projekty, které spolu nějakým způsobem souvisí (např. projekty

vytvořené pro konkrétní ročník FFIMU). Pokud již není skupina aktuální, může ji administrátor deaktivovat, čímž znemožní vytváření nových projektů pro tuto skupinu.

2.4.2 ER diagram

Objekty popsané výše jsou zachyceny na ER diagramu na obrázku 2.11 na následující straně v podobě databázových entit. Návrh databáze byl jedním z prvních kroků při vstupní analýze, což přispělo k pochopení celého systému po logické stránce a efektivnímu vývoji. Během implementace na modelu proběhla řada úprav a optimalizací. Klíčové koncepty však zůstaly zachovány. Pro zjednodušení návrhu byly použity v několika případech atributy typu JSONB. Konkrétně se jedná o inventární čísla, záznamy o pracovní době a dovolené výdejářů. Toto řešení sice porušuje kritéria hned první normální formy relační databáze, nicméně v těchto konkrétních případech nevzniká žádná redundance a zvolené řešení výrazně zjednodušuje implementaci jak aplikačního tak uživatelského rozhraní.



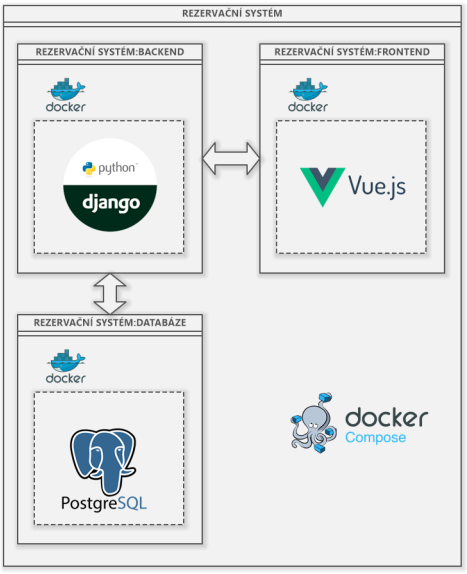
Obrázek 2.11: ER diagram

3 Implementace

Technologie pro tvorbu webových aplikací se vyvíjejí extrémním tempem. Výběr vhodných nástrojů je tak nelehkým úkolem. Cílem bylo zvolit nástroje, které adoptují nejnovější trendy v dané oblasti, ale zároveň mají dostatečné zázemí a pevné místo na trhu. Rezervační systém je svým rozsahem spíše menší aplikací, která není příliš náročná na výkon. Důraz je kladen spíše na udržitelnost respektive udržovatelnost, rozšířitelnost a kvalitní UX. Z toho důvodu bylo odděleno uživatelské rozhraní a logická vrstva aplikace neboli frontend a backend do dvou samostatných modulů. Toto rozdělení s sebou nese oproti monolitickým systémům nevýhodu v podobě nutnosti implementace API mezi oběma vrstvami. Výhodou je ovšem možnost využití velmi pokročilých nástrojů pro efektivní vývoj UI. Zároveň je aplikace připravena pro integraci dalších systémů pomocí již hotového API. Aktuálně je naplánovaná například integrace na nově vznikající video archiv laboratoře v rámci bakalářské práce Viktora Kozáka [13].

3.1 Architektura

Serverová část je postavená na jazyce Python s využitím frameworku Djanogo a jeho nástavby Django rest framework. Klientská část je napsána převážně v jazyce JavaScript za použití populárního[11] frameworku Vue.Js. Obě části spolu komunikují skrze REST API prostřednictvím https protokolu. Pro ukládaní dat je využit databázový systém PostgreSQL. Systém je navržený tak, aby byl spustitelný na libovolné platformě. Z toho důvodu je využita technologie Docker, která izoluje jednotlivé moduly do kontejnerů s vlastním virtualizovaným prostředím. To napomáhá nejen při vývoji, ale díky nástroji Docker Compose je možné celý systém nasadit pomocí jediného příkazu na libovolný server podporující tuto technologii. Architekturu systému znázorňuje obrázek 3.1 na následující straně.



Obrázek 3.1: Architektura systému

3.2 Frontend

Frontend je část aplikace, se kterou přichází do styku uživatel. Jelikož se v tomto případě jedná o webovou aplikaci, jde o uživatelské rozhraní postavené na HTML, CSS a JavaScriptu. Vývoj efektivního a uživatelsky přívětivého rozhraní pouze v rámci těchto technologií by byl extrémně náročný a neflexibilní. K dispozici je ale obrovské množství frameworků a knihoven, které vývoj usnadňují. Jedním z nepoužívanějších a nejoblíbenějších frameworků současnosti pro vytváření webových aplikací pomocí JavaScriptu je Vue.js [11].

3.2.1 Vue.js

Vue.js je open-source nástroj pro vývoj interaktivních webových aplikací balancující na pomezí knihovny a komplexního frameworku [14]. K jeho použití stačí importovat JavaScriptový kód z CDN, ale nabízí i robustní řešení postavené na Node.js. Aktuálně tento framework využivají i technolgičtí giganti jako Google, Apple nebo Gitlab [15]. Vue umožňuje vytváření vlastních HTML elementů, tedy komponent, a řídí komunikaci mezi nimi. Tyto komponenty jsou reaktivní, což

znamená, že pokud se změní model komponenty, UI se automaticky překreslí. Podobně jako například React používá Vue pro vykreslování rozhraní virtuální DOM (Document Object Model), čímž minimalizuje počet pomalých operací s reálným DOM a zvyšuje tak výkon UI. Za dobu jeho existence od roku 2014 se vytvořila rozsáhlá sada podpůrných nástrojů[1]. Kromě obrovského množství knihoven se jedná i například o doplněk do prohlížeče Google Chrome, který umožňuje procházet komponenty a sledovat jejich aktuální datový model, nebo komunikaci, a výrazně tak zjednodušuje vývoj.

3.2.2 Vue CLI – konfigurace projektu

Jedním z velmi užitečných podpůrných nástrojů je Vue CLI, což je konzolová aplikace, která na základě vstupních parametrů připraví celý projekt pro vývoj i kompilaci pomocí nástroje Webpack a nabídne automatickou integraci řady užitečných balíčků. V prvním kroku dává aplikace na výběr mezi instalací Vue 2 a Vue 3. Verze Vue 3 byla vydána v roce 2020 a původním plánem bylo tuto novinku využít i pro vývoj rezervačního systému. Bohužel nová verze přináší řadu změn ve svém API a většina knihoven se na úpravy zatím nestihla adaptovat. Rozhodujícím prvkem byla knihovna Vuefity poskytující UI komponenty, která prozatím podporuje pouze Vue 2. Z toho důvodu byla nakonec pro vývoj rezervačního systému zvolena starší verze.

Dále instalátor nabízí podporu Babelu, což je nástroj na transpilaci JavaScriptového kódu. Aktuální verzí JavaScriptu je ECMAScript 2021. Ta ale obsahuje řadu funkcí, které zatím nejsou podporovány všemi webovými prohlížeči. Díky implementaci Babelu je kód při sestavení automaticky transpilován do starší verze JavaScriptu a vývojář tak může používat moderní funkce bez ohledu na podporu prohlížečů[16].

Vue CLI nabízí též zvolit jako primární jazyk TypeScript, což je nástavba JavaScriptu umožňující využívat výhod datových typů a typové kontroly při kompilaci. S plnou podporou TypeScriptu však přichází až Vue 3. V případě předchozí verze se jedná spíše o experimentální použití. Z toho důvodu byl vybrán JavaScript.

1. `https://github.com/vuejs/awesome-vue`

Paleta užitečných nástrojů je dále doplněna o Vue Router. Vzhledem k tomu, že rezervační systém je koncipovaný jako SPA, neboli webová aplikace, která dynamicky mění obsah jediné načtené stránky pomocí JavaScriptu, místo aby při každé změně načetla celou stránku znovu, je potřeba řídit navigaci mezi jednotlivými pohledy aplikace a tuto službu poskytuje právě Vue Router.

Další z problémů, který je třeba vyřešit, je udržování interního stavu webové aplikace. Aplikace se skládá z velkého množství komponent, které mezi sebou potřebují sdílet data. Vue poskytuje pro tuto komunikaci jednoduché nástroje již v základní konfiguraci. U rozsáhlejších aplikací je však doporučované řešení spravovat stav pomocí nástroje Vuex. Vuex je mutací state-management nástroje Flux, který je založený na konceptech unidirectional data flow, tedy jednosměrného toku dat, a single source of truth – jediného zdroje dat pro všechny komponenty.

Vue CLI dále nabízí integraci CSS preprocesoru. Díky tomuto nástroji je možné kromě klasických kaskádových stylů používat v kódu komponent pokročilejší funkce SCSS. Preprocesor umožňuje využívat ve stylopisu například proměnné, cykly, funkce a další užitečné prvky[17], které umožňují psát čistší a kratší kód. Ten je při sestavení kompilován do klasického CSS. Díky tomu není problém s kompatibilitou prohlížečů[17].

Dalším nástrojem, který rezervační systém z nabízeného setu využívá, je Linter. Jedná se o program, který provádí nad kódem statickou analýzu. Napomáhá tak k odhalení chyb již při psaní kódu a umožňuje nastavit pravidla pro formátování, což usnadňuje psaní konzistentního a čistého kódu.

Posledním balíčkem poskytnutým přes Veu CLI je Vuetify[2], což je open-source knihovna poskytující širokou škálu hotových přizpůsobitelných komponent vytvořených na základě pravidel Material Designu s konzistentním API a rozsáhlou dokumentací.

Díky Vue CLI jsou všechny tyto nástroje nainstalovány a připraveny k použití během několika jednoduchých kroků. Vue CLI navíc poskytuje i automatickou konfiguraci serveru pro efektivní vývoj. Pomocí příkazu `npm serve` je možné spustit vývojový server s funkcí hot-reload, která sleduje změny v kódu, a při každé změně aktualizuje

2. `https://vuetifyjs.com/`

náhled v prohlížeči. Není tak nutné ani ručně spouštět kompilaci kódu, ani obnovovat prohlížeč, aby se změny projevily.

Všechny nástroje poskytnuté skrze inicializaci projektu přes Vue CLI shrnuje tabulka 3.1.

Projekt vytvořený pomocí Vue CLI je zcela nezávislý na vývojovém prostředí a platformě.

Tabulka 3.1: Nástroje integrované prostřednictvím Vue CLI

Webpack	build a devserver
Babel	transpiler do starších verzí JS
Vue Router	navigace v SPA
Vuex	state management
SCSS	CSS preprocesor
ESLint	statická analýza kódu
Vuetify	knihovna UI komponent

3.2.3 Struktura projektu

Vue.js poskytuje jako jednu z variant vytváření komponent takzvané Single-File komponenty. Jedná se v podstatě o soubory s příponou .vue, které se skládají ze tří částí. První částí je template. Ta obsahuje HTML kód rozšířený o speciální atributy, které umožňují dynamicky renderovat obsah na základe funkcí a dat definovaných v následující části script. Tato část obsahuje data a metody definující chování komponenty a poslední částí je style, tedy stylopis.

Z těchto komponent se skládá celé uživatelské rozhraní. Výchozí komponentou je je `App.vue`, která obsahuje horní navigační panel a RouterView komponentu, která za použití Vue Routeru zajišťuje vykreslování všech ostatních komponent. Komponenty jsou rozděleny na `views`, které představují jednotlivé pohledy zobrazované pomocí routeru a `components`, což jsou opakovaně použitelné univerzální komponenty. Většinu těchto komponent poskytuje knihovna Vuetify, nicméně v některých případech bylo efektivnější udělat nad kompo-

nentami vlastní wrapper, který upravuje jejich rozhraní, což přispělo k přehlednějšímu kódu v rámci principu DRY (Don't Repeat Yourself).

3.2.4 Optimalizace kódu komponent

Ne vždy bylo ale zvýšeni granularity komponent přínosem. Se zvyšujícím se počtem komponent narůstala i režie jejich vzájemné výměny dat, a tak jejich dekompozice na komplexnosti kódu často spíše přidala. Bylo tedy nutné nalézt vhodný kompromis mezi granularitou a rozsahem jednotlivých komponent. Tím, že Vue komponenty kombinují v jednom souboru HTML CSS a JavaScript, počet rádků kódu rychle narůstal. Vhodným řešením se nakonec ukázal byt HTML preprocesor Pug.

Pug nahrazuje uzavírání tagů pomocí odsazení a poskytuje syntactic sugar pro často používané opakující se části HTML kódu[18]. Šablony komponent se díky využití tohoto nástroje zkrátily v určitých případech až o stovky řádků. A kód se stal mnohem lépe čitelným.

3.2.5 Komunikace s backendem

Veškerá komunikace s backendem probíhá pomocí REST API, které je dotazované pomocí knihovny Axios. Axios umožňuje pomocí takzvaných interceptorů před odesláním každého dotazu provést definované akce a případně dotaz upravit. Této funkce je využito při autorizaci requestů.

API je zabezpečeno pomocí JWT (JSON Web Token), což je standard umožňující autorizaci bez použití sessions. Každý dotaz tak musí v hlavičce obsahovat autorizační token. Pomocí interceptorů se před každým dotazem Axios nejprve podívá, zda má k dispozici platný token, případně zda se token nachází v local storage prohlížeče. Pokud token nalezne, přidá jej do hlavičky. V opačném případě končí dotaz chybou 401: Unauthorized.

3.2.6 Generování PDF

Jedním z úkolů rezervačního systému je generování předávacího protokolu v PDF, který je nutné podepsat při převzetí techniky. Jedná se

v podstatě o smlouvu mezi fakultou a vypůjčitelem. Tato smlouva obsahuje veškeré detaily o rezervaci včetně všech vypůjčených zdrojů.

Na generování PDF dokumentů existuje řada knihoven. Často ale mají nedostatečnou podporu výše zmíněných požadavků na formátování, a nebo vyžadují externí služby operačního systému, což by kvůli jednomu jednoduchému PDF souboru zbytečně zvýšilo komplexnost celého systému.

Po zvážení několika variant byla vybrána knihovna jsPdf, která umožňuje převádět do pdf jednoduchý HTML kód a navíc má k dispozici velmi dobře fungující zásuvný modul na dlouhé tabulky. Ani toto řešení však nebylo bez komplikací. Nástroj ve výchozí konfiguraci podporuje pouze základní ASCII znaky. Pří použití české diakritiky se tak stal dokument nečitelným. Bylo tedy potřeba importovat vlastní písma, která unicode znaky podporují. Balíček jsPdf nabízí API pro import písem pouze ve formátu Base64. Bylo tedy nutné nejprve získat soubory jednotlivých řezů písma, tyto převést do Base64 a naimportovat je do příslušné instance tohoto nástroje.

Další překážkou byla chyba v kódu knihovny, která umožňovala vložit do PDF dokumentu HTML kód pouze jedenkrát. Předávací protokol ale obsahoval formátovaný text, poté tabulku a následoval opět formátovaný text. Bylo tedy potřeba vložit HTML před i za tabulku. Chyba byla nahlášena v repozitáři a v době psaní této práce byl k dispozici i merge request, který tuto chybu opravoval. Bohužel ale nebyl součástí poslední vydané verze. Proto bylo nutné najít alternativní řešení.

Zásuvný modul pro generování tabulky poskytuje informaci o absolutních souřadnicích posledního řádku tabulky. Díky tomu bylo možné vypočítat jak je tabulka vysoká a na základě teto informace byl vložený mezi první a druhou část HTML margin, který vytvořil volný prostor na tabulku.

3.3 Backend

Backend zajišťuje logickou vrstvu mezi uživatelským rozhraním a databází a implementuje potřebné algoritmy a validace dat. V případě rezervačního systému se jedná o vytvoření API, které poskytne přes https protokol klientské aplikaci (frontendu) přistup ke čtení a úpra-

vám dat. Pro tento úkol bylo třeba najít nástroj, který bude dostatečně robustní, aby pro vytvoření API nebylo potřeba mnoho vlastního kódu, jehož údržba by padla na bedra budoucích vývojářů rezervačního systému. Zároveň bylo ale žádoucí, aby systém měl minimum závislostí a zůstal tak modulární multiplatformní a flexibilní. Vhodným adeptem splňujícím tyto požadavky je framework Django.

3.3.1 Django Framework

Django je open-source high-level framework pro vývoj webových aplikací podporující rapidní vývoj. Je vytvořený na základě požadavků zkušených webových vývojářů a adresuje většinu běžných úkolů a problémů, které vývojáři webových aplikací řeší[19]. Django rozděluje aplikaci na modely views a serializéry. Pomocí modelů je možné definovat veškerá data v systému vytvořením tříd. Obecně se dá říci, že každý model reprezentuje jednu tabulku v databázi[20]. Views zodpovídají za zpracovávání webových požadavků a generování odpovědí. V rámci souboru `urls.py` jsou pak tyto views namapované na na konkrétní url adresy. Serializéry pak zajišťují efektivní převod dat z datových struktur Pythonu do formátu vhodného pro odpověď na webový požadavek jako je například JSON. Dále mají na starosti validaci příchozích dat a jejich převod pro zpracování na straně Pythonu.

Velkou předností Django frameworku je implementace vlastního ORM (object-relational mappig), což je funkce umožňující přistupovat k databázi podobně, jako by se jednalo o objekty v Pythonu. Pomocí ORM lze vytvářet komplexní dotazy na databázi bez nutnosti vytváření SQL dotazů.

3.3.2 Django admin

Django také poskytuje širokou paletu nástrojů pro podporu vývoje prostřednictvím konzolové aplikace. Ta umožňuje například lokálně spustit vývojový server, který sleduje změny kódu a po každé změně aplikaci aktualizuje. Není tak nutné restartovat server, aby se úpravy projevily. Dále umožňuje vytvářet automatické migrace databáze nebo tyto migrace provádět, vytvářet uživatele nebo přistupovat z konzole přímo k objektům aplikace.

3.3.3 Django admin site

Pro usnadnění správy aplikace přichází Django s konfigurovatelným administrátorským rozhraním, skrze které lze přes webový prohlížeč přistupovat do databáze, nebo konfigurovat zásuvné moduly. V případě rezervačního systému jsou některé výjimečné administrátorské operace ponechány právě pouze v tomto rozhraní, jelikož například seznam výdejářů se za posledních 10 let nezměnil, stejně jako výčet kategorií oprávnění. Bylo by tak neoptimální pro tyto operace vytvářet dedikované UI ve frontendu aplikace.

3.3.4 Django Rest Framework

Django je primárně určený pro vývoj full-stack aplikací, tedy včetně uživatelského rozhraní. Pro účely rezervačního systému je však využita pouze logická vrstva. Z toho důvodu byla použita nástavba Django Rest Framework (DRF), což je sada nástrojů, která umožňuje pomocí Django frameworku velmi efektivně vytvářet REST API. Například pomocí třídy `ModelViewSet` z DRF je možné vytvořit API na CRUD (create, read, update, delete) pro konkrétní model za použití čtyř řádků kódu.

Model View Set má definovaný queryset, který určuje jaká data budou prostřednictvím API poskytována, `serializer_class` určující který serializér bude použitý k serializaci dat a `permission_classes` definující kdo a k jakým operacím má oprávnění u daného API přistupovat. DRF poskytuje několik základních tříd oprávnění jako například `IsAuthenticatedOrReadOnly` nebo `IsAdminUser` a podobně. Pro komplexnější řízení oprávnění je však možné vytvořit vlastní třídy oprávnění, které umožňují implementovat logiku specifickou pro konkrétní aplikaci.

Například třída `CommonReadAdminAndProviderAll` na ukázce 3.1 na následující straně umožňuje Administrátorům a výdejářům provádět všechny CRUD operace, ale běžným uživatelům umožňuje pouze čtení.

```
class CommonReadAdminAndProviderAll(permissions.
    BasePermission):
    def has_permission(self, request, view):
        return request.user.is_authenticated and
        (request.user.role == Role.ADMIN or
        request.user.role == Role.PROVIDER or
        request.method in SAFE_METHODS)
```

Listing 3.1: Třída oprávnění umožňující čtení uživatelům a zápis správcům

Vzhledem k tomu, že tyto restrikce se týkají mnoha objektů, stačí pouze použít tuto třídu a není nutné definovat přístupová oprávnění pro každý model zvlášť.

3.3.5 Výpočet blokujících rezervací

Backend rezervačního systému je poměrně minimalistický a v podstatě se jedná pouze o API mezi databází a frontendem, které přidává prvky autentizace, validace a řízení oprávnění. Jeden z mála algoritmů, které bylo potřeba navrhnout a implementovat, je výpočet potenciálních konfliktů rezervací. Na rozdíl od předchozího rezervačního systému disponuje nový RS dynamickým filtrováním zdrojů na základě dostupnosti v daný termín. Z toho důvodu je potřeba při každém načtení všech zdrojů na frontendu vypočítat pro každý zdroj všechny intervaly, ve kterých je položka již rezervována. Ač se tento problém může jevit na první pohled triviální, není řešení zcela přímočaré. Důkazem může být i fakt, že předchozí systém měl právě v této detekci chybu, která způsobovala, že systém za velmi specifických podmínek umožnil zapůjčit tentýž zdroj více uživatelům ve stejný čas.

Vzhledem k tomu, že výpočet je nutné provést pro všechny zdroje při každém jejich načtení, bylo navíc potřeba jej optimalizovat tak, aby netrval příliš dlouho.

Naivní strategií by mohlo být pro každý zdroj vyfiltrovat všechny rezervace, které zahrnují daný zdroj, a z jejich začátků a konců sestavit pole blokujících intervalů. Tento výpočet by však poskytoval přesné výsledky jen za předpokladu, že uživatelé vyzvedávají zdroje přesně v termínu začátku rezervace a vrací je v deklarovaném termínu vrácení. To však zdaleka neodpovídá realitě. Situaci navíc komplikuje fakt, že zdroje z jedné rezervace není nutné vracet všechny naráz. Pro každý

zdroj tak může být skutečný blokující interval jiný, byť se jedná o stejnou rezervaci.

Výpočet blokujícího intervalu

Skutečnou informaci o dostupnosti zdroje nám tak neposkytne třída `Reservation` ale `ReservedResource` (viz Rezervovaný zdroj v sekci 2.4.1 na straně 26). Její instance mapují zdroje ke konkrétním rezervacím a drží mimo jiné skutečný datum vyzvednutí a vrácení daného zdroje v rámci příslušné rezervace.

Tato třída má metody `blocking_start` a `blocking_end`, které vrací skutečný začátek a konec intervalu, po který je zdroj nedostupný.

```
def blocking_start(self):
  if self.real_pickup_date is not None:
    return self.real_pickup_date
  else:
    return self.reservation.pickup_date_time

def blocking_end(self):
  if self.reservation.pickup_date_time > timezone.now():
      # reservation not started yet
    return self.reservation.return_date_time
  if self.real_return_date is not None:
      # res. started but resource was already returned
      return self.real_return_date
  if self.reservation.return_date_time > timezone.now():
      # reservation started, resources was not returned,
   but its ok because reservation is still in progress
      return self.reservation.return_date_time
  if self.real_pickup_date is None:
      # reservation started, reservation ended,
      # resources was not returned, but its ok because
      # reservation was not picked up
      return self.reservation.return_date_time
  # res. started, res. ended, resources was picked up and
    not returned yet so resource is blocked till now.
  return timezone.now()
```

Listing 3.2: Výpočet blokujícího intervalu zdroje

Vyfiltrováním všech objektů `ReservedResource` pro daný zdroj a voláním metod `blocking_start` a `blocking_end` získáme kompletní

výpis všech skutečných intervalů, ve kterých je nebo byl zdroj vypůjčen se zohledněním všech možných scénářů.

Optimalizace výpočtu

Jak bylo uvedeno výše, pro zajištění dynamického filtrování zdrojů na základě potenciálních konfliktů je nutné provést výpočet blokujících intervalů pro všechny zdroje při každém jejich načtení. Za posledních 10 let však proběhlo bezmála 3000 výpůjček a výpočet všech skutečných blokujících intervalů každé výpůjčky by při takovém počtu trval velmi dlouho.

Časovou náročnost výpočtu navíc dramaticky zvyšuje skutečnost, že při použití referencí na cizí klíče u metod za použití Django ORM dochází k takzvanému N+1 problému. Obě metody pro výpočet blokujícího intervalu využívají referenci na objekt Rezervace. Django ORM tak při požadavku na `ReservedResource` nejprve načte z databáze tabulku `ReservedResource` a pak na úrovni Pythonu pro každý řádek tabulky posílá další dotaz na databázi pro získání objektu konkrétní rezervace na základě cizího klíče. Výsledkem je N+1 dotazů na databázi, kde N je počet řádků v tabulce `ReservedResource`.

Jedná se o známý problém ORM a Django na tento problém nabízí také řešení[21]. Při definování querysetu u ModelViewSetu je možné použít metodu `prefetch_related`, která dopředu poskytuje ORM informaci, že budou pro vyhodnocení požadavku potřeba data z jiné tabulky a ORM díky tomu už při dotazu na tabulku `ReservedResource` udělá join na úrovni databáze, což je znatelně efektivnější, než postupné dotazování řádek po řádku.

Na související tabulku je navíc možné aplikovat samostatný queryset, což umožňuje ještě více optimalizovat dotaz. Velké množství rezervací totiž nemůže z hlediska svých atributů vůbec konflikt způsobit a proto není nutné je do výpočtu vůbec zahrnovat.

Riziko konfliktu způsobují pouze rezervace, které nejsou zamítnuté (neschválené rezervace jsou implicitně blokující, dokud nejsou explicitně zamítnuté. Zdroje jsou tedy blokovány od chvíle vytvoření rezervace přesto, že rezervace prozatím čeká na schválení), dále rezervace, které mají datum vrácení nastavený na pozdější termín, než je aktuální čas a rezervace, které jsou vyzvednuté a zároveň nejsou vrácené.

```
queryset=ReservedResource.objects.filter(
  ~Q(reservation__approved=False) &
  # exclude declined reservations
  (Q(reservation__return_date_time__gt=timezone.now()) |
  # include upcoming and ongoing reservations
  (Q(real_pickup_date__isnull=False) &
   Q(real_return_date__isnull=True))))
  # include picked up reservations
```

Listing 3.3: Filtr rezervací, které mohou způsobit konflikt

3.3.6 Mailing

Kromě API pro uživatelské rozhraní zajišťuje backend ještě jednu službu a tou je zasílání notifikací prostřednictvím e-mailu. Django má implementovanou podporu e-mailingu ve svém jádře. Stačí tedy doplnit údaje k SMTP serveru a pomocí metody `send_mail` je možné jednoduše zasílat zprávy.

Upozornění jsou zpravidla zasílána v reakci na událost, jako je například vytvoření rezervace. V některých případech je ale třeba zaslat upozornění, pokud nastane konkrétní moment v čase. Příkladem může být upozornění na blížící se expirace rezervace nebo na periodické notifikace o nevyřízených požadavcích.

K tomuto účelu je často využívána systémová služba Cron, která umožňuje naplánovat spouštění příkazů v pravidelných intervalech. Django má pro tento účel k dispozici knihovnu Django_cron, která poskytuje API pro práci s Cronem z prostředí aplikace. Tato knihovna je však podporována pouze na operačním systému Linux a rezervační systém byl od začátku koncipován tak, aby byl spustitelný na libovolné platformě.

Proto byl zvolen nástroj Django APScheduler, který poskytuje stejné API jako Cron, je nezávislý na platformě a přidává navíc užitečné funkce jako automatické logování provedených akcí do databáze.

Scheduler je nutné spustit jako samostatný proces voláním příkazu `python manage.py runapscheduler`. Všechna upozornění jsou uvedena v tabulce 3.2 na následující straně.

Tabulka 3.2: E-mailová upozornění

Událost	**Příjemce**	**Čas**
nová rezervace	výdejář	při odeslání
požadavek na schválení rezervace	admin	při odeslání
potvrzení/zamítnutí rezervace	výdejář, vlastník	při odeslání
požadavek na oprávnění	admin, výdejář	při odeslání
vyhodnocení požadavku na oprávnění	žadatel	při odeslání
vyhodnocení požadavku na rezervaci	vlastník	při odeslání
report o nevyřízených požadavcích na rezervace	admin, výdejář	denně ve 13:00
report o nevyřízených požadavcích na oprávnění	admin, výdejář	denně ve 13:00
nevyzvednutá započatá rezervace	vlastník	denně ve 13:00
nevrácená ukončená rezervace	vlastník	denně ve 13:00
blížící se termín vyzvednutí rezervace	vlastník	24 hodin před
blížící se termín vrácení rezervace	vlastník	24 hodin před

3.4 Autentizace uživatelů

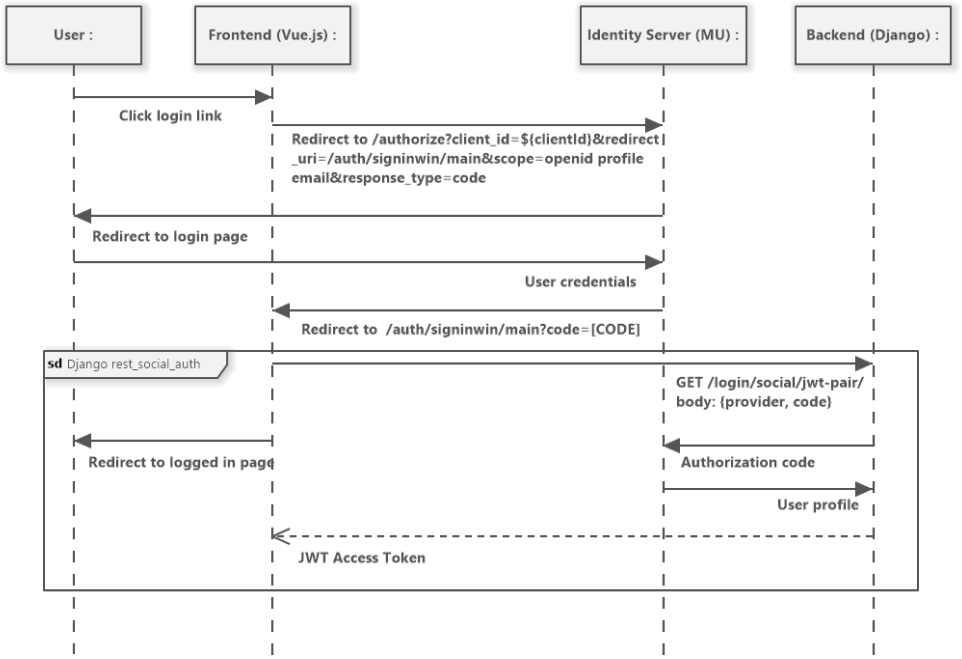
Jedním z klíčových úkolů bylo zajistit Autentizaci uživatelů prostřednictvím jednotného univerzitního přihlášení. MU poskytuje službu, umožnující využívat univerzitní login pro aplikace třetích stran. Aplikaci je nutné zaregistrovat na stránce v systému pro správu identit Perun[22], a poté, co projde registrace schvalovacím procesem, je možné začít službu využívat.

Systém nabízí autentizaci prostřednictvím dvou protokolů. Jedná se o OIDC (OpenID Connect) a SAML (Security Assertion Markup Language). SAML je rozsáhlá technologie vytvořená v roce 2005 založená na XML. Oproti tomu OIDC je poměrně čerstvým rozšířením protokolu OAuth 2.0[23] a bylo vyvinuto právě pro účely webových a mobilních aplikací s důrazem na jednoduchost jeho použití[24].

Z výše uvedeného popisu vyplývá, že pro účely rezervačního systému se jeví OIDC jako jasná volba. Přes deklarovanou jednoduchost však byla implementace tohoto řešení poměrně komplexní a vyžadovala podrobné nastudování fungování protokolu. Náročné bylo především zajištění předávání autentizačních dat mezi uživatelem, frontendem, backendem a identity providerem, tedy univerzitou. OIDC poskytuje pro účely autentizace několik možných postupů tzv. flows. Vzhledem k tomu, že frontend je samostatná klientská aplikace interpretovaná webovým prohlížečem, nebylo možné použít poměrně přímočarý implicit flow, jelikož by to znamenalo odhalení `client_secret` klíče. Z toho důvodu byl implementován Authorization code flow, který přidává do ověřovacího procesu další kroky, které umožňují použít `client_secret` pouze na straně backendu a tím ho skrýt. Celý proces je zachycený na sekvenčním diagramu 3.2 na následující straně.

3.4.1 Proces autentizace

Pokud nový uživatel uživatel otevře rezervační systém, jako první se zobrazí obrazovka s tlačítkem pro přihlášení. Po kliknutí se otevře nové okno prohlížeče, ve kterém je uživatel přesměrován na poskytovatele identity, tedy na `oidc.muni.cz/oidc/authorize`, s příslušnými parametry, a vyzván k zadání přihlašovacích údajů. Následuje výzva k souhlasu s poskytnutím osobních údajů aplikaci Rezervační systém LEMMA. Pokud uživatel potvrdí souhlas, je přesměrován nazpět do rezervačního systému spolu s autorizačním kódem v parametru viz diagram 3.2 na následující straně. Tento kód je předán backendu aplikace prostřednictvím endpointu `/login/social/jwt-pair`. Backend pak odešle tento kód a v kombinaci s `client_id` a `client_secret` zpět poskytovateli identity a jako odpověď získá informace o uživateli, který se právě přihlásil, včetně UČO a e-mailové adresy. Backend se pokusí najít uživatele s daným UČO v databázi, a pokud ho nenalezne, vytvoří nový záznam. V odpovědi backend vrací JWT Access a



Obrázek 3.2: Sekvenční diagram procesu autentizace

Refresh token. Access token pak slouží k autentizaci všech dotazů na API rezervačního systému, ale jeho platnost je omezena na 15 minut. K prodloužení jeho platnosti je možné použít Refresh token, jehož platnost je nastavena na 3 dny.

3.4.2 Testování autentizace

Vývoj a testování do značné míry komplikovala skutečnost, že každá úprava konfigurace služby na straně univerzity musela vždy projít schvalovacím procesem což trvalo i 24 hodin. Závislost na externí službě byla nežádoucí jak při vývoji, tak pro účely samotného testování.

Řešení tohoto problému poskytl nástroj OpenId Connect Server Mock[3]. Ten nabízí prostřednictvím nástroje Docker Compose velmi jednoduchý způsob, jak spustit lokálně vlastní identity server. Veš-

3. `https://github.com/Soluto/oidc-server-mock`

který kód pro spuštění lokálního serveru včetně předpřipravených testovacích uživatelským účtů je v jediném yaml souboru.

Pokud je rezervační systém spuštěn ve vývojovém módu, je na přihlašovací stránce k dispozici kromě tlačítka pro univerzitní login také tlačítko mock login, které je nastavené právě na přihlášení pomocí lokálního identity serveru. Testovací uživatelské účty je možné upravit či rozšířit v souboru `mock-oidc/docker-compose.yml`.

4 Nasazení

Nedílnou součástí této diplomové práce je uvedení rezervačního systému do provozu. Cílem práce bylo systém navrhnout tak, aby jeho údržba a aktualizace byla co nejsnazší podle principů continous delivery [25]. K tomuto účelu byla využita řada DevOps nástrojů jako Docker Compose, Gitlab CI/CD, nebo Portainer. Následující kapitola popisuje postupy, které jsou využity k dosažení tohoto cíle, a kroky, které bylo nutné provést při prvním nasazení.

4.1 Produkční prostředí

U backendu i frontendu byly využity během vývoje nástroje poskytované frameworky ke spuštění vývojových serverů na lokálních portech. Tyto servery jsou velmi praktické pro účely vývoje, ale nejsou optimalizované ani určené k využití v produkčním prostředí. Pro účely produkce je u obou modulů nutné vytvořit build. K sestavení a spuštění jednotlivých modulů byla využita technologie Docker.

4.2 Kontejnerizace

4.2.1 Docker

Docker je open-source nástroj, jehož cílem je poskytnout jednotné rozhraní pro kontejnerizaci aplikací v prostředí Linux, Windows a Mac OS. Jedná se v podstatě o odlehčenou virtualizaci [26]. Poskytuje také rozhraní pro efektivní komunikaci mezi jednotlivými virtualizovanými kontejnery pomocí Docker Compose.

Kontejner obsahuje pouze požadované aplikace a pro ně specifické soubory, ale neobsahuje jádro operačního systému. Tím je výrazně snížena režie na rozdíl od klasických virtuálních strojů. Každý kontejner ale může mít nainstalovány jiné aplikace nebo jiné verze aplikací, aniž by se vzájemně ovlivňovaly.

K vytvoření kontejnerů je potřeba takzvaný image, což je v podstatě šablona, podle které Docker dokáže spustit kontejner s požadovanými parametry. Tyto předpřipravené images je možné ukládat do registrů a je možné je buď rovnou použít k vytvoření kontejneru, nebo na

jejich základě vytvořit další vrstvu a vytvořit vlastní image. Docker poskytuje Docker Hub[1], což je rozsáhlý registr hotových images. Je tak možné využít například připravený optimalizovaný image pro běh Python aplikace již s předinstalovaným Pythonem, nebo image s připraveným databázovým systémem Postgres, kterému stačí předat přihlašovací údaje k databázi v parametrech a spustit jej.

K přidávání vlastních vrstev na již existující image slouží Dockerfile. Pomocí něj je možné k existujícímu image nakopírovat například zdrojový kód, nainstalovat potřebné knihovny, spustit kompilaci nebo případně přidat i instrukce ke spuštění konkrétního procesu při spuštění kontejneru. Proces vytváření nového image lze navíc rozdělit do jednotlivých fází.

4.2.2 Dockerfile

Výše popsaný postup lze dobře ilustrovat na Dokefile pro vytvoření produkčního image pro frontend aplikace, jehož kód je zachycen v ukázce 4.1.

```
# build stage
FROM node:16-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY ./ .
RUN npm run build

FROM nginx:latest as production-stage
RUN mkdir /app
COPY --from=build-stage /app/dist /app
COPY nginx.conf /etc/nginx/nginx.conf
```

Listing 4.1: Frontend dockerfile

V první části je jako základ využit image node:16-alpine, do kterého jsou přidány zdrojové kódy a nainstalovány všechny potřebné knihovny a balíčky pro kompilaci frontendu. Poté je kompilace spuštěna a zkompilovaný kód je vložen do složky `dist`. Node.js, stejně jako všechny knihovny použité při kompilaci, však nejsou pro běh aplikace potřeba. Proto Dockerfile obsahuje ještě druhou část neboli

1. `https://hub.docker.com/`

stage, která je postavená na image nginx:latest, což je webový server. K tomuto serveru je nakopírována pouze již zkompilovaná aplikace z předchozí fáze. Díky tomu je build znatelně rychlejší a výsledný image má menší velikost.

Podobným způsobem je řešený i Dockerfile pro backend.

4.2.3 Docker Compose

Předchozí sekce popisuje vytvoření images jednotlivých modulů. K tomu, aby na základě těchto images byly vytvořeny kontejnery, které spolu budou spolupracovat, slouží nástroj Docker Compose.

Ten umožňuje pomocí yaml soubrů definovat images, ze kterých budou konteinery vytvořeny, poskytnout jim virtuální prostor pro sdílená data, vytvořit síť, po které mohou mezi sebou komunikovat, definovat závislosti a procesy, které budou v jednotlivých kontejnerech spuštěny, a také nastavit potřebné proměnné prostředí. Díky této konfiguraci je pak možné pomocí příkazu `docker-compose up` spustit naráz všechny kontejnery, ze kterých se systém skládá a bez nutnosti žádných dalších úprav.

4.3 Produkční server

Pro účely nasazení rezervačního systému má LEMMA k dispozici nový virtuální server s operačním systémem Debian. Rezervační systém je však navržený pro běh v izolovaném prostředí Dockeru, a proto na produkčním serveru nebylo třeba mnoho úprav.

Na produkční server byl nainstalován pouze Docker a GitLab runner, který se stará o automatické nasazování nových verzí. Všechny ostatní služby jsou izolovány ve vlastních kontejnerech.

Kromě samotného rezervačního systému jsou na serveru spuštěny ještě dvě další služby, přičemž každá má svůj vlastní docker-compose soubor, který se stará o její konfiguraci a spuštění. Kompletní architektura služeb na serveru je zachycená na obrázku 4.1 na straně 53.

4.3.1 Nginx proxy manager

Aby bylo možné přistupovat do rezervačního systému z internetu pomocí https protokolu, byla zapotřebí implementace reverzního proxy

serveru, který zpracovává požadavky z internetu a předává je požadovaným službám na základě url.

Kromě routování požadavků je také potřeba zajistit generování a obnovu SSL certifikátů pro zajištění zabezpečené komunikace. Vhodným řešením pro tyto úkoly je Nginex proxy manager, což je nástroj, který poskytuje předpřipravený docker image, který stačí spustit a pomocí webového GUI lze proxy server snadno nakonfigurovat. Navíc je integrován na službu Let's Encrypt, což je certifikační autorita, která vydává SSL certifikáty zdarma. Díky tomu je možné SSL certifikát vygenerovat jedním kliknutím a Nginex proxy server se od té doby postará o automatickou obnovu certifikátů před jejich expirací [27].

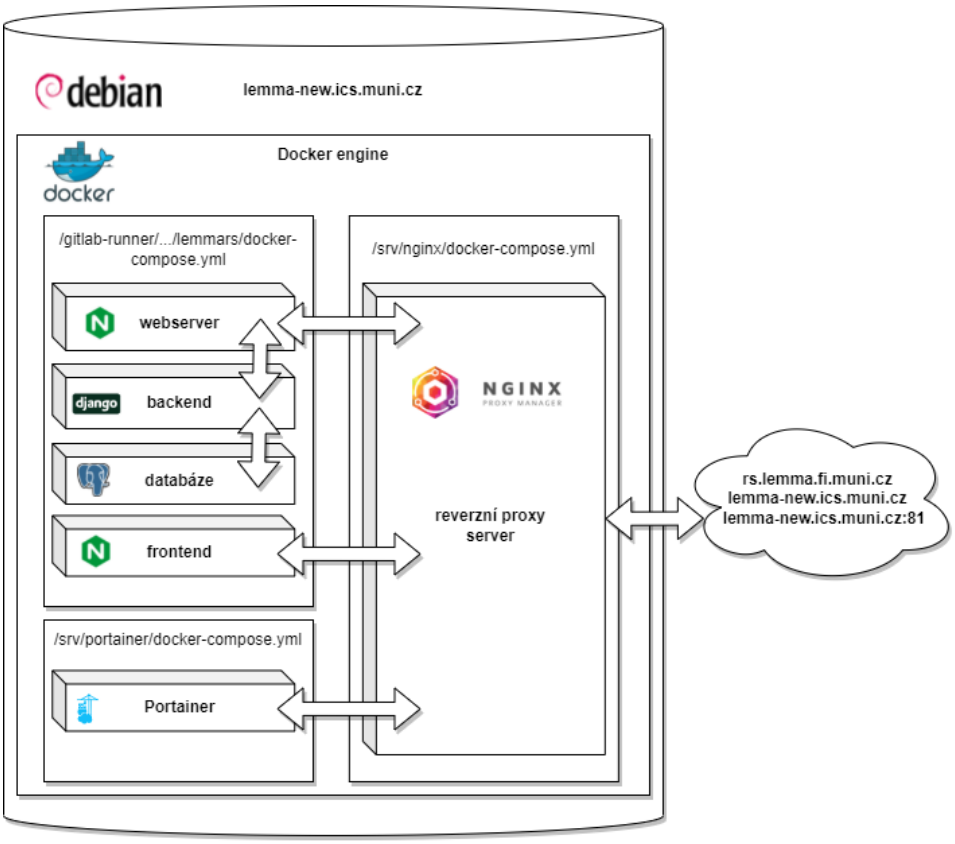
4.3.2 Portainer

Portainer je jednoduchá webová aplikace, která umožňuje spravovat Docker služby na vzdáleném serveru skrz internetový prohlížeč. Prostřednictvím přehledného GUI umožňuje například spouštět a vypínat jednotlivé kontejnery, spravovat Docker images, procházet logy jednotlivých kontejnerů, mapovat jejich stav nebo i vytížení procesoru a operační paměti v reálném čase. V případě výpadku nebo problémů tak napomáhá k rychlé lokalizaci případně i odstranění chyby [28].

4.4 GitLab CI/CD

Kód rezervačního systému je verzován na fakultním GitLab serveru. Kromě správy zdrojového kódu je však využíváno i dalších služeb této platformy jako je evidování tiketů a také CI/CD pipelines. Rezervační systém je navržen tak, aby jeho údržba a aktualizace byla co nejsnazší. Kromě důrazu na multiplatformní podporu pro účely vývoje je také zajištěno i automatické nasazení na produkční server.

Gitlab umožňuje pomocí souboru `.gitlab-ci.yml` definovat kroky, které budou automaticky provedeny při konkrétní akci [29]. Rezervační systém této funkce využívá tak, že při každém commitu do master větve je vytvořena pipeline, která nahraje aktuální zdrojový kód na produkční server a provede nasazení nové verze.



Obrázek 4.1: Architektura produkčního serveru

4.4.1 GitLab runner

K tomu, aby GitLab mohl provádět operace na produkčním serveru, slouží nástroj GitLab runner. Jedná se o program, který poslouchá požadavky z GitLab serveru a při přijetí požadavku na základě instrukci ze souboru `.gitlab-ci.yml` provede požadovaný úkol.

Runner je nejprve nutné nainstalovat na produkční server a spustit jako službu operačního systému. V dalším kroku je pak třeba zaregistrovat v tomto runneru repozitář pomocí registračního tokenu, který je možné získat v nastavení konkrétního repozitáře, a URL GitLab serveru.

Poté je runner připraven zpracovávat požadavky. Aby nedošlo k nechtěnému automatickému nasazení, je pipeline nastavená na ruční spuštění. S commitem do master větvě je tedy pouze vytvořena a čeká na ruční spuštění na záložce pipelines.

Z dosud neznámého důvodu se občas stane, že runner požadavek nepřevezme a pipeline se zasekne ve stavu pending. V takovém případě je potřeba se přihlásit na produkční server přes SSH a spustit přikaz `gitlab-runner restart`. Poté je zpravidla úkol automaticky zpracován.

4.5 Import dat

Původní rezervační systém za svou mnohaletou existenci obsloužil stovky rezervací a projektů, v rámci kterých bylo zapůjčováno přibližně 300 různých zdrojů. Vzhledem k tomu, že nový rezervační systém pozměnil datovou strukturu zásadním způsobem, bylo rozhodnuto, že import historie rezervací a projektů realizován nebude. Nevyhnutelný byl však transfer zdrojů, u kterých došlo k restrukturalizaci také.

Například skupiny zdrojů byly nahrazeny štítky a výrobní číslo nahradily interní poznámky. Import také komplikovala skutečnost, že zdroj má jak ve starém, tak v novém RS několik závislostí na jiných objektech. Konkrétně se jedná o výdejáře, úroveň oprávnění a štítky. Bylo tak nejprve nutné vytvořit tyto objekty a až poté naimportovat samotné zdroje.

Django pro tento účel disponuje velmi praktickým nástrojem Djnago fixtures. Ten umožňuje pomocí příkazu `python manage.py load <fixturename>` načíst data z JSON souboru, validovat je podle modelu a importovat přímo do databáze.

Pro účely importu tak byl vytvořen soubor `sources.json`, který obsahuje předpřipravené objekty výdejářů, úrovní oprávnění a také štítků, které jsou odkazovány ze zdrojů.

Pro správné namapování objektů a konverzi struktur byl vytvořen skript `migrateResources.js`, který na svém vstupu převezme výstup SQL dotazu na zdroje z původní databáze a jehož výstupem je JSON fixture obsahující všechny potřebné objekty pro import do nové databáze.

Tento vygenerovaný soubor se stal nejen zdrojem dat pro nový systém, ale také velmi užitečnou pomůckou pro naplnění testovací databáze daty při vývoji a testování.

4.6 Spuštění a testování systému

Beta verze rezervačního systému již byla nasazena na produkční server a v době psaní tohoto textu probíhá již přibližně měsíc uživatelské testovaní. Správce techniky již plní nový rezervační systém novými daty a uživatelé jsou vyzváni, aby pro účely důkladného otestování vytvářeli rezervace paralelně jak ve starém tak v novém rezervačním systému.

Během testování bylo odhaleno několik drobných nedostatků, které jsou evidovány prostřednictvím tiketů v GitLabu a před spuštěním ostrého provozu je naplánována jejich oprava. Jedná se především o drobnosti v rámci uživatelského rozhraní, ale také byla zjištěna například nedostatečná kompatibilita s preferovaným prohlížečem na zařízeních Apple Safari. Nově hlášené nedostatky jsou průběžně odstraňovány a díky velmi efektivnímu automatickému nasazování nových verzí jsou opravy k dispozici na produkčním serveru velmi rychle.

Nasazení do ostrého provozu je naplánováno na začátek roku 2022.

5 Závěr

Laboratoř LEMMA disponuje rezervačním systémem, který byl vytvořen v roce 2008 a od té doby je udržován prostřednictvím závěrečných prací studentů FI. Za 13 let svojí existence nasbíral v rámci několika iterací mnoho různorodých funkcí a stal se tak robustní monolitickou aplikací s nedostatečnou dokumentací, která vykazovala řadu nedostatků a nebylo snadné ji nadále udržovat.

Cílem diplomové práce bylo nahradit tento systém odlehčenou modulární aplikací postavenou na moderních technologiích, která bude mít přívětivé uživatelské rozhraní a bude snadno udržovatelná. Za tímto účelem byla provedena důkladná analýza požadavků a zmapování procesů v laboratoři, jejichž je rezervační systém součástí.

Výstupem práce je responzivní webová SPA aplikace založená na technologiích Vue.js (JavaScript) a Django (Python). Nasazování nových verzí na produkční server je plně automatizované za pomocí nástrojů Gitlab CI/CD a Docker a k dispozici je také dashboard pro monitorování stavu aplikace skrze Portainer.

Při vývoji byl kladen maximální důraz na to, aby údržba a další vývoj rezervačního systému byly co nejsnazší. Systém byl tedy vyvinut jako multiplatformní. Díky tomu je možné ve vývoji pokračovat na OS Windows, MacOS nebo Linux. Kromě samotné aplikace zahrnuje práce i sadu předpřipravených nástrojů, které usnadňují přípravu vývojového prostředí i vývoj samotný. Příkladem může být simulátor jednotného univerzitního přihlašování viz sekce 3.4 na straně 44.

5.1 Splnění cílů a další vývoj

V době psaní tohoto textu má laboratoř k dispozici nový rezervační systém nasazený na produkčním serveru a připravený k použití. Díky důkladné analýze byla řada úkonů prováděných v systému významně zjednodušena a byly odstraněny funkční nedostatky předchozího řešení.

V rámci analytické části bylo identifikováno 18 případů užití, ke kterým by mohl být systém využíván a na jejíchž základě byla formulována akceptační kritéria. Bohužel, z důvodu omezeného časového rámce diplomové práce, musely být tyto požadavky prioritizovány

a dvě kritéria bylo nutné přenechat k dalšímu vývoji. Konkrétně se jedná o podporu prodloužení rezervace a transferu rezervace na jiného uživatele.

Kromě těchto požadavků tiketovací systém v repozitáři aplikace v současné době eviduje dalších 16 nápadů na vylepšení systému, které vznikly v průběhu vývoje nebo testování. Rezervační systém je navzdory očekávání poměrně komplexním nástrojem a prostor pro zlepšení zde bude vždy. V rámci vývoje však byla věnována maximální energie tomu, aby budoucí zlepšení byla co nejsnazší.

5.2 Shrnutí

Osobně považuji výstup práce za úspěch ve dvou rovinách. Především jsem poskytl laboratoři, ke které mám osobní vztah, nástroj, který, jak doufám, přispěje k jejímu dalšímu fungování a rozvoji, a také jsem si díky řešení netriviálních problému širokého spektra rozšířil skill set v oblasti analýzy, full-stack vývoje i DevOps.

Bibliografie

1. JELÍNEK, Jan. *Webový rezervační systém*. 2019. Dostupné také z: https://is.muni.cz/th/dpdve/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Jan SEDMIDUBSKÝ.
2. FILIPOVIČ, Jiří. *Systém alokace zdrojů laboratoře*. 2006. Dostupné také z: https://is.muni.cz/th/vz98h/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Eva HLADKÁ.
3. ROZUM, Peter. *Vývoj a provoz rezervačního systému*. 2015. Dostupné také z: https://is.muni.cz/th/f5a4y/. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Petr SOJKA.
4. VEJMOLA, Jakub. *Akvizice obrazu a zvuku pomocí vybavení laboratoře LEMMA* [online]. 2008 [cit. 2021-02-25]. Dostupné z: https://is.muni.cz/th/dnbr0/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Petr SOJKA.
5. JELÍNEK, Pavel. *Rozšíření a údržba rezervačního systému laboratoře LEMMA* [online]. 2008 [cit. 2021-02-26]. Dostupné z: https://is.muni.cz/th/m5ukw/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Petr SOJKA.
6. URBÁŠEK, Radim. *Webový video archiv laboratoře LEMMA*. 2011. Dostupné také z: https://is.muni.cz/th/bjdc7/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Petr SOJKA.
7. *Adobe Flash Player EOL General Information Page* [online] [cit. 2021-01-31]. Dostupné z: https://www.adobe.com/cz/products/flashplayer/end-of-life.html.
8. KOVAŘÍK, Tomáš. *Uživatelské rozhraní webových aplikací laboratoře LEMMA*. 2019. Dostupné také z: https://is.muni.cz/th/z4vxz/. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Petr SOJKA.

9. KUDELA, Tomáš. Jednotný vizuální styl [online]. [N.d.] [cit. 2021-12-07]. Dostupné z: https://www.muni.cz/o-univerzite/fakulty-a-pracoviste/rektorat/994200-oddeleni-vnejsich-vztahu-a-marketin/jednotny-vizualni-styl.

10. *Material Desgn* [online] [cit. 2021-12-07]. Dostupné z: https://en.wikipedia.org/wiki/Material_Design.

11. *State of JS 2020* [online] [cit. 2021-12-07]. Dostupné z: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks.

12. CHANG, Darren. *Thoughts on Material Design* [online] [cit. 2021-12-07]. Dostupné z: https://uxplanet.org/thoughts-on-material-design-4d61fa2176b5.

13. KOZÁK, Viktor. *Webový mediální archiv laboratoře LEMMA*. 2021. Dostupné také z: https://is.muni.cz/th/upxmh/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Petr SOJKA.

14. *Vue introduction* [online]. Vue.js [cit. 2021-05-21]. Dostupné z: https://vuejs.org/.

15. LUCA, Spezzano. *Google, Apple and Other Users of Vue.js* [online]. Medium, 2020-11-23 [cit. 2021-12-12]. Dostupné z: https://medium.com/notonlycss/google-apple-and-other-users-of-vue-js-e4505359e5d5.

16. PASSAGLIA, Andrea. *Vue.Js 2 Cookbook*. Birmingham: Packt Publishing, 2017. ISBN 9781786468093.

17. WATTS, Luke. *Mastering Sass*. Birmingham: Packt Publishing, 2016. ISBN 9781785883361.

18. *Getting Started with Pug* [online]. Pug, 2021 [cit. 2021-05-23]. Dostupné z: https://pugjs.org/api/getting-started.html.

19. JULIE, Korsun. *Why We Use Django Framework & What Is Django Used For* [online]. Django Stars, 2021-03-03 [cit. 2021-05-23]. Dostupné z: https://djangostars.com/blog/why-we-use-django-framework/.

20. *Django makes it easier to build better Web apps more quickly and with less code.* [Online]. Django Software Foundation [cit. 2021-05-23]. Dostupné z: https://www.djangoproject.com/.

21. ALEXANDRE, Ignjatovic. *Understanding and fixing N+1 query* [online]. Medium, 2020-06-10 [cit. 2021-12-08]. Dostupné z: https://medium.com/doctolib/understanding-and-fixing-n-1-query-30623109fe89.

22. *Zavádíme na univerzitě Perun, systém pro správu identit* [online]. Masarykova univerzita, 2020 [cit. 2021-06-11]. Dostupné z: https://it.muni.cz/aktuality/zavadime-na-univerzite-perun-system-pro-spravu-identit.

23. LI, Wanpeng; MITCHELL, Chris J. User Access Privacy in OAuth 2.0 and OpenID Connect. In: *2020 IEEE European Symposium on Security and Privacy Workshops* (*EuroS PW*). 2020, s. 664–672. Dostupné z DOI: 10.1109/EuroSPW51379.2020.00095.

24. *SAML vs. OpenID* (*OIDC*) [online]. Auth0® Inc., 2020 [cit. 2021-09-11]. Dostupné z: https://auth0.com/intro-to-iam/saml-vs-openid-connect-oidc/.

25. SWARTOUT, Paul. *Continuous Delivery and DevOps : A Quickstart Guide*. Birmingham: Packt Publishing, 2017. ISBN 9781849693684.

26. *Docker overview* [online]. Docker Inc. [cit. 2021-05-21]. Dostupné z: https://docs.docker.com/get-started/overview/.

27. *Nginx Proxy Manager Guide* [online]. Jamie Curnow [cit. 2021-07-21]. Dostupné z: https://nginxproxymanager.com/guide/.

28. *Introducing Portainer* [online]. Portainer, 2021 [cit. 2021-05-23]. Dostupné z: https://www.portainer.io/.

29. NAIR, Sanjay. *What is CICD — Concepts in Continuous Integration and Deployment* [online]. Medium, 2018-05-06 [cit. 2021-05-23]. Dostupné z: https://medium.com/@nirespire/what-is-cicd-concepts-in-continuous-integration-and-deployment-4fe3f6625007.

A Záloha kódu rezervačního systému

Přiložený archiv s názvem lemmars.zip obsahuje zálohu GitLab repozitáře[1] rezervačního systému ke dni 13.12. 2021.

1. `https://gitlab.fi.muni.cz/lemma/lemmars`