FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Systém pro podporu vedení závěrečných prací
Student:	Fidan Guliyeva
Vedoucí:	Ing. Pavel Náplava, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Navrhněte a vytvořte systém pro podporu vedení závěrečných prací. Postupujte následujícím způsobem:
1) analyzujte požadavky na systém ze strany vyučujících a studentů,
2) na základě požadavků prozkoumejte, zda podobný systém již neexistuje,
3) pokud existuje, inspirujte se jím pro vytvoření systému nového,
4) tento systém musím být jednoduše použitelný a provozovatelný,
5) vyberte vhodné implementační nástroje a prostředí systému,
6) proveďte návrh systému,
7) navržený systém implementujte a proveďte jeho uživatelské testování,
8) na závěr vyhodnoťte benefity systému, porovnejte je s náklady na vývoj a provoz a rozhodněte, zda je vhodné tento systém provozovat a za jakých podmínek.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 29. října 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Systém pro podporu vedení závěrečných prací

Fidan Guliyeva

Katedra softwarového inženýrství
Vedoucí práce: Ing. Pavel Náplava, Ph.D.

23. června 2022

Poděkování

Ráda bych poděkovala vedoucímu Ing. Pavlu Náplavovi, Ph.D., za odborné vedení a morální podporu poskytnutou při zpracování této práce. Dále bych chtěla poděkovat své rodině a přátelům za podporu po celou dobu mého studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 23. června 2022 . . . . . .

Odkaz na tuto práci

Guliyeva, Fidan. *Systém pro podporu vedení závěrečných prací*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022. Dostupný také z WWW: ⟨`https://supervision-support-system.herokuapp.com`⟩.

Abstrakt

Tato bakalářská práce ukazuje celý proces vývoje webové aplikace určené pro podporu vedení závěrečných prací. Pro vývoj webové aplikace byl zvolen programovací jazyk Python a také byly použity webframy Flask a Bootstrap 5. Výstupem práce je bezplatný systém určený pro zjednodušení plánování konzultací a evidenci pokroku studentů během procesu vedení závěrečných prací.

Klíčová slova web aplikace, rezervační systém, konzultace závěrečné práce

Abstract

This bachelor thesis shows the whole process of web application development designed to support the management of final theses. The Python programming language was chosen for the development of the web application and the Flask and Bootstrap 5 web frames were also used. The output of the work is a free system designed to simplify the planning of consultations and record the progress of students during the process of conducting final theses.

Keywords web application, reservation system, final thesis consultation

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Moderní svět je těžké si představit bez použití informačních systémů. Jedním z nejjednodušších příkladů každodenního informačního systému lze nazvat telefonní seznam nebo deník. Hlavní funkcí, kterou sleduje naprosto každý informační systém, je sběr, uchovávání a vyhledávání informací. Ale stejně tak, jako se tok informací zvyšuje, roste i čas potřebný k jejich zpracování. V dnešní realitě je ruční provádění většiny procesů poměrně obtížné, takže v posledních letech se stále více každodenních úkolů provádí online pomocí počítačů a mobilních zařízení. Mezi tyto úkoly patří také objednávání různých služeb. Téměř každá společnost, která poskytuje služby, má svůj systém, přes který si můžete tyto služby objednat a využívat, včetně vzdělávacích institucí. Služby vzdělávacích institucí zahrnují přednášky, semináře, cvičení a konzultace.

Například na univerzitě ČVUT funguje systém, který studentům umožňuje rezervovat si místa na předměty, které studují, a čas na výuku. Tento systém však nepodporuje možnost rezervace soukromé konzultace s vyučujícím v rámci psaní závěrečných prací. Tento typ konzultací je dlouhodobě žádaný a vyučující má zpravidla více studentů, což komplikuje proces plánování konzultačních termínů. Vyučující se mohou při řešení tohoto problému obrátit na existující systémy, ale tento směr je bohužel příliš úzký. Většina systémů, které mají funkci, kterou potřebujeme (rezervace služeb), je určena pro podnikatele. Kvůli tomu mají tyto systémy přetížené rozhraní, což zvyšuje náklady. V rámci své bakalářské práce vytvořím systém pro podporu vedení závěrečných prací. Systém bude evidovat veškeré konzultace konané mezi vyučujícím a studenty.

Mojí hlavní motivací bylo přání projít všemi fázemi implementace informačního systému, od analýzy až po nasazení.

Kapitola 1

Cíl práce

Hlavním cílem této bakalářské práce je vytvořit systém určený pro podporu vedení závěrečných prací. Práce je rozdělena do několika dalších dílčích cílů. Prvním dílčím cílem je analýza systémových požadavků ze strany vyučujících a studentů. Druhým dílčím cílem je studium podobných systémů, které částečně nebo zcela pokrývají požadavky stanovené pro náš systém, a inspirace jimi při implementaci vlastního systému. Třetím dílčím cílem je výběr vhodných implementačních nástrojů a systémového prostředí. Čtvrtým dílčím cílem je vytvoření návrhu systému. Pátým dílčím cílem je implementace aplikace dle vytvořeného návrhu a uživatelské testování naší aplikace. Posledním cílem je vyhodnotit přínosy systému, porovnat je s náklady na vývoj a provoz a rozhodnout se, zda tento systém provozovat a za jakých podmínek.

Kapitola 2

Analýza

Tato kapitola se zaměřuje na prozkoumání problémové domény současně používaného systému či spíše na analýzu procesů spojených s plánováním konzultačních termínů a jejich nedostatků, návrh na zlepšení těchto procesů a identifikaci uživatelských požadavků.

2.1 Současná řešení

Vedoucí mé bakalářské práce v současné době používá komerční software Microsoft OneNote[2]. Microsoft OneNote je digitální poznámkový blok.
V současné době je využívána vzdělávací verze tohoto softwaru, která poskytuje řadu doplňkových funkcí pro interakci učitele a studenta.
Hlavní výhody tohoto softwaru jsou:

- pracovní prostor pro spolupráci
- spousta nástrojů pro úpravu textu
- historie verzí
- přístupnost z jakéhokoli zařízení
- bezpečnost

2.1.1 Problémová doména

Pro zadání konzultačních termínů byla vyučujícím ručně vytvořena tabulka (viz 2.1). Student musel zadat své unikátní uživatelské jméno do prázdné buňky tabulky, aby se mohl přihlásit ke konzultačnímu termínu. Pokud učitel z nějakého důvodu nemohl v daný den provést konzultaci (den pracovního klidu nebo osobní důvod), byl sloupec označen červeně a níže byla přidána

vysvětlivka. Učitel pořádal konzultace každé úterý a středu v pevném hodinovém intervalu.
Nevýhody:

- Nedostatek potřebné funkčnosti pro plánování konzultačních termínů.
  - V důsledku toho musí vyučující při vytváření konzultačního termínu neustále kontrolovat kalendář, aby věděl, na který den v týdnu připadá nebo zda termín nepřipadá na den pracovního klidu.

Proces plánování konzultačních termínů je poměrně pracný a na základě výše uvedeného můžeme konstatovat, že toto řešení není efektivní. Možným řešením je využití jiných systémů určených speciálně pro rezervaci termínů, tzv. rezervační systémy.

Tabulka 2.1: Příklad tabulky určený pro přihlášení na konzultační termín.

	22.03.2022	29.03.2022	5.04.2022	12.04.2022	19.04.2022
15:00	guliyfi1				
15:30			someone		
16:00					
16:30		someone1			

2.2 Uživatelské požadavky

Než se pustíme do hledání stávajících řešení, musíme také zjistit, jakou funkcionalitu by v tomto systému rádi viděli samotní koncoví uživatelé. S vedoucím práce a jeho studenty proběhla řada schůzek, na základě kterých byly stanoveny následující uživatelské požadavky:

Učitel

UP1 Vytváření jednorázových konzultačních termínů

UP2 Vytváření opakujících se konzultačních termínů

UP3 Odstranění konzultačních termínů

UP4 Zohlednění dnů pracovního klidu při vytváření konzultačních termínů

UP5 Zobrazení historie rezervací

UP6 Rezervace studenta na konzultační termín

UP7 Zrušení rezervace studenta na konzultační termín

UP8 Přiložení poznámky k rezervaci

UP9 Zanechání poznámky o samotných studentech

Student

UP10 Zobrazení historie rezervací

UP11 Rezervace na konzultační termín

UP12 Zrušení rezervace na konzultační termín

UP13 Přiložení poznámky pro učitele k rezervaci

UP14 Zobrazení jmen studentů přihlášených na konzultační termín

2.3 Možné řešení

Tato část je věnována vyhledávání a analýze systémů určených pro správu rezervací. Analýza bude zahrnovat prozkoumání procesu tvorby konzultačního termínu, identifikaci výhod a nevýhod a splnění stanovených uživatelských požadavků.

Začala jsem hledat možná řešení v internetovém prohlížeči Google a klíčovými slovy pro hledání byl "systém pro plánování konzultací" a "rezervace služeb". Google zobrazil kolem 2 000 000 výsledků.
Systémy měly:

- poskytovat funkcionalitu pro řízení konzultací,
- být určené pro vzdělávací účely,
- podporovat český jazyk,
- být bezplatné,
- být dostupné pro použití prostřednictvím prohlížeče.

Nepodařilo se mi najít systémy navržené speciálně pro učitele, všechny systémy, které jsem našla, byly navrženy pro lidi nebo firmy poskytující komerční služby. Vzhledem k tomu, že tyto systémy nejsou určeny k řešení konkrétního problému, ale poskytují obecné řešení pro různé oblasti služeb, je jejich funkčnost přetížena (provádění plateb, propagace na sociálních sítích, analýza zisků, poskytování statistik) a za toto přetížení byl účtován dodatečný poplatek. Zpočátku jsem preferovala bezplatné systémy, které podporují i češtinu, ale jelikož jsem bezplatný systém nenašla, zvolila jsem 3 komerční softwary, které poskytují svým potenciálním zákazníkům demoverzi, na základě které jsem provedla svou další analýzu. Jak jsem již uvedla výše, systém nebyl navržen tak, aby naplňoval naše konkrétní cíle, takže jsem použila pouze tu část funkce, která pokrývala naše požadavky.

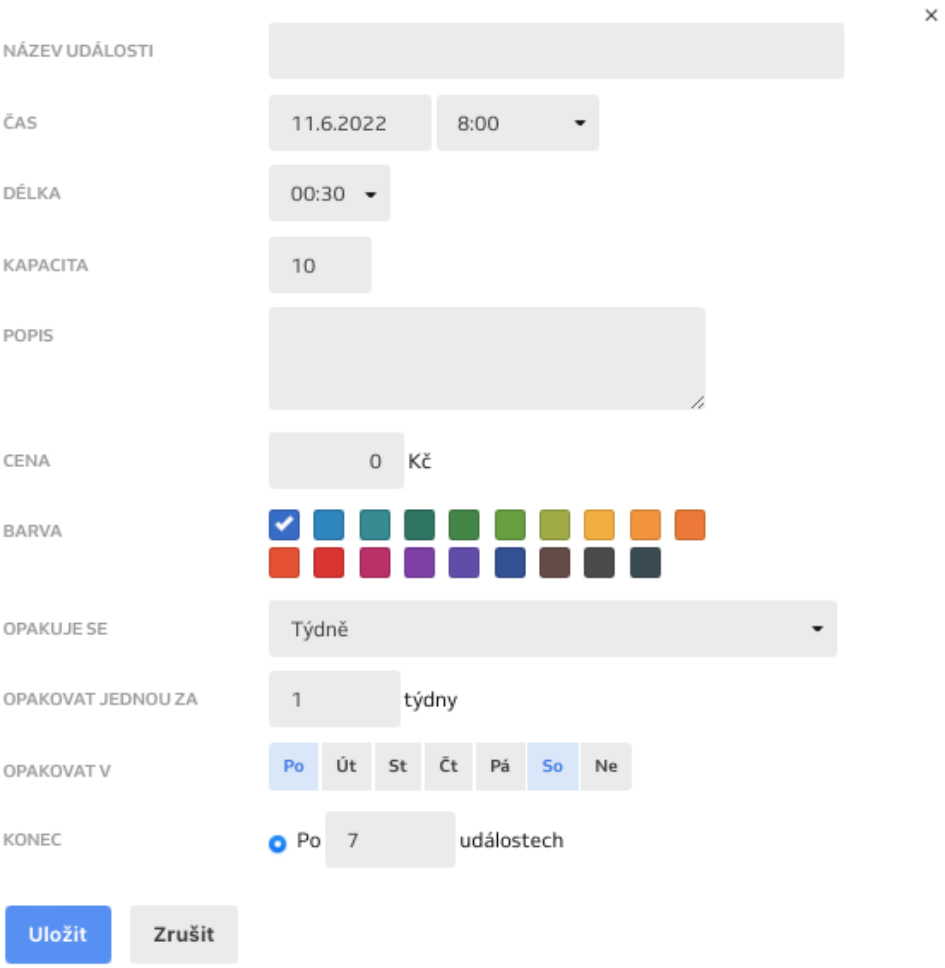
2.3.1 Reservio

Reservio[16] je komerční rezervační systém služeb.

Analýza byla provedena na bezplatném balíčku se sníženou funkčností, ale žádnou další funkci jsme kromě zvýšení počtu rezervací z placených balíčků nepotřebovali.

Pro vytvoření konzultačního termínu je nutné vyplnit formulář, který se otevře v modálním okně (viz obrázek 2.1). Velkou výhodou je, že si můžeme zvolit, které dny v týdnu a jak často chceme termín opakovat v určitém časovém rámci, jelikož řada vyučujících, včetně mého vedoucího, si vyčleňuje na konzultace určité dny v týdnu. Za zmínku ale také stojí, že zde můžeme vytvořit vždy pouze jednu konzultaci. Pokud tedy učitel vyčlenil například v pondělí hodinové okno od 9 do 12 hodin na konzultace, z nichž každá trvá 30 minut, v rámci tohoto systému bude muset samostatně vytvořit 6 konzultací. Při studiu systému jsem také zjistila tyto nevýhody:

- I když můžeme do systému přidat nepracovní dny, nelze je nakonfigurovat tak, aby se opakovaly každý rok. Když už mluvíme o svátcích, znamená to každoroční událost, v takovém případě je bude nutné každý rok ručně zadávat do systému.
- Uživatel může vytvořit rezervaci na termín bez registrace. K tomu bude muset uvést jméno, příjmení a e-mail. V tomto případě se nekontroluje pravost e-mailu, což vyvolává otázky ohledně bezpečnosti systému.



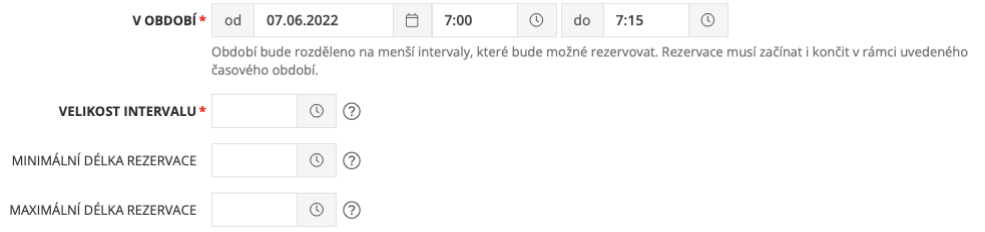
Obrázek 2.1: Snímek obrazovky modálního okna s formulářem pro vytvoření termínu – Reservio

2.3.2 Reenio

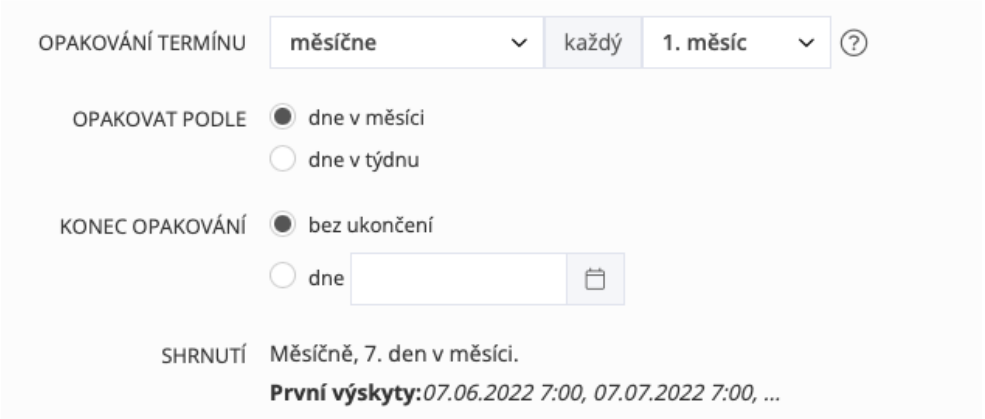
Reenio je komerční online rezervační systém určený pro všechny podnikatele a firmy, které nabízejí přímé služby zákazníkům nebo se s nimi setkávají.[17] Reenio je nabízen formou Saas(viz [3]).[18] Poskytuje 3 balíčky[18], které obsahují určité funkce. Naše analýza byla provedena na bezplatném balíčku, který slouží k seznámení se systémem a není časově omezen, ale má omezené funkce. Tato funkce nám ale stačí, až na to, že může být vyžadováno zvýšení počtu rezervací. Pro vytvoření konzultačního termínu je nutné vyplnit formulář, který se otevře v dialogovém okně (viz obrázky 2.2, 2.3, 2.4). Je na výběr ze 4 typů termínů (viz obrázek 2.2) a také je k dispozici dokumentace[4], ve které je každý z nich podrobně popsán. Po přečtení dokumentace jsem se rozhodla, že nejvýhodnější je pro nás intervalový termín, který představuje ča-

sový interval rozdělený na menší časové intervaly, na které se mohou studenti dále zapisovat. Dále musíme zvolit začátek časového intervalu, určit hodinový interval a také délku trvání každé konzultace (viz obrázek 2.3). Také zde můžeme nastavit minimální a maximální délku rezervace. To znamená, že pokud je minimální délka rezervace 30 minut a délka konzultace 15 minut, pak si student musí rezervovat dva po sobě jdoucí termíny. Stejný princip funguje také pro maximální délku rezervace. Diskutovala jsem o zavedení takových omezení v implementaci našeho systému, ale v našem případě se ukázaly jako nadbytečné. Systém nám také dává možnost vytvářet opakující se termíny (viz obrázek 2.4). Můžeme si vybrat, který termín chceme vytvořit (denní, týdenní, měsíční nebo roční). U opakujících se termínů musíme také dodatečně definovat konec časového intervalu a frekvenci opakování (např. každý týden nebo každé dva měsíce). Pokud jsme zvolili týdenní opakování, pak uvidíme 7 zaškrtávacích políček odpovídajících každému dni v týdnu. Pokud bychom zvolili měsíční opakování, potom musíme zvolit opakování po dnech v měsíci nebo dnech v týdnu. Například na začátku časového intervalu jsme zvolili 6. den libovolného měsíce. V prvním případě se termín bude opakovat každý 6. den. Ve druhém případě se termín bude opakovat každý týden v ten samý den.
Zde jsme při přidávání nepracovních dnů do systému opět narazili na problém, že nelze každoročně opakovat nepracovní den, zde však lze nastavit týdenní opakování.

Obrázek 2.2: Snímek obrazovky dialogového okna s formulářem pro vytvoření termínu - Reenio



Obrázek 2.3: Snímek obrazovky dialogového okna s formulářem pro vytvoření termínu - Reenio



Obrázek 2.4: Snímek obrazovky dialogového okna s formulářem pro vytvoření termínu webu Reenio

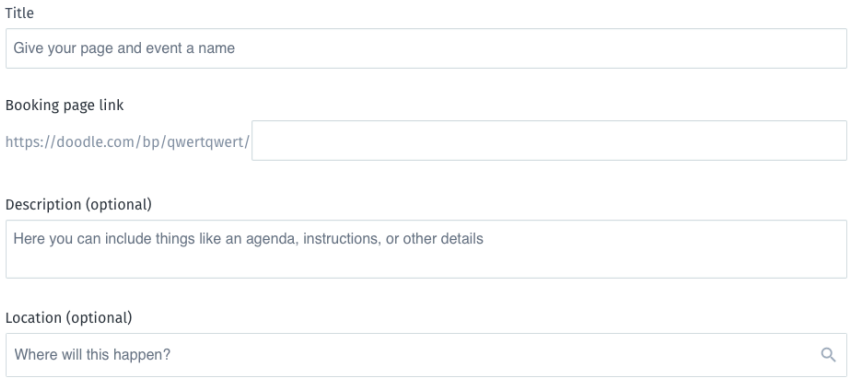
2.3.3 Doodle

Doodle je online kalendářní nástroj pro správu času a koordinaci schůzek.[19] Tento systém mi doporučil nastudovat vedoucí mé bakalářské práce. Jedná se o placenou aplikaci, která nám poskytuje 14denní zkušební verzi. Pro práci s tímto systémem je potřeba mít účet v Google nebo Microsoft 365, protože systém je již vázán na existující online kalendář a následné rezervace se zobrazí v jednom z vybraných kalendářů. Pro vytvoření termínu musíme vyplnit údaje ve formuláři, který se nám otevřel na webové stránce (viz obrázky 2.5, 2.6, 2.7). Na začátku musíme uvést všechny základní informace o termínu: kde se bude konat, z jakého odkazu se k němu lze dostat, informace o termínu samotném (viz obrázek 2.5). Dále definujeme dobu trvání a dostupnost termínu (viz obrázky 2.6, 2.7). Nastavíme délku konzultace. Můžeme použít navrženou hodnotu nebo si ji sami definovat. Při definování časového intervalu jej můžeme definovat uvedením data začátku a konce nebo jej nedefinovat vůbec. Ve druhém případě pak musíme definovat budoucí horizont rezervací, aby se studenti například nemohli přihlásit na termín, který je vzdálený více než týden. Dále zvolíme, ve které dny v týdnu chceme konzultace poskytovat a časové období. Je třeba poznamenat, že tento formulář je velmi vhodný pro vytváření termínů.
Také byla v průběhu studie zjištěna řada závažných nedostatků:

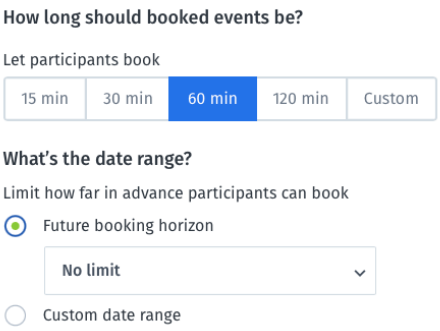
- Ačkoliv zde můžete nastavit jazyk, v rozbalovacím seznamu jazyků není čeština.

- Když student zruší rezervaci, termín se již neobjeví v seznamu dostupných záznamů, a ani s právy tvůrce termínu se mi jej nepodařilo znovu zpřístupnit.

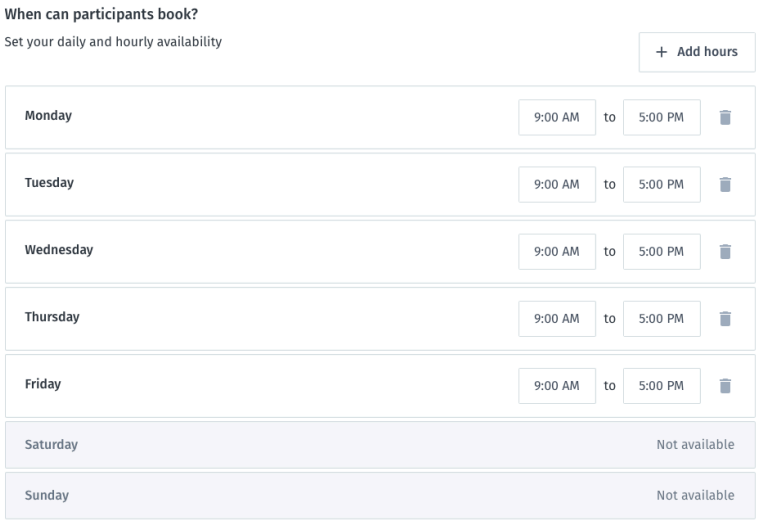
- Pro rezervaci termínu se uživatel nemusí přihlašovat do systému, stačí zadat údaje (které si můžeme sami určit), což má také negativní dopad na bezpečnost.
- Jakmile učitel vytvoří termín, nemůže jej smazat.



Obrázek 2.5: Snímek obrazovky s formulářem pro vytvoření termínu - Doodle



Obrázek 2.6: Snímek obrazovky s formulářem pro vytvoření termínu - Doodle



Obrázek 2.7: Snímek obrazovky s formulářem pro vytvoření termínu - Doodle

2.3.4 Shrnutí analýzy možných řešení

Ze všech systémů, které jsem zhodnotila výše, Reenio nejlépe vyhovoval našim potřebám, ačkoli systém nepokrýval všechny zadané uživatelské požadavky (viz 2.2). Za zmínku také stojí, že systém je poměrně drahý (225Kč/měsíc) a pokud používáte bezplatnou verzi, je zde limit na počet rezervací, na který v našem případě učitel nebude stačit. Doodle je pohodlný v tom, že se integruje se systémem Microsoft 365, který je na českých univerzitách aktivně využíván, ale z vlastní zkušenosti jsem viděla, že studenti se aktivně odhlašují z konzultačních termínů a pokud termín po odhlášení studenta zůstane nedostupný pro přihlášeni, tak to bude velký problém jak pro učitele, tak pro samotné studenty. Nejjednodušší na naučení se ukázalo Reservio, systém však nepočítá s rozdělením časového intervalu na menší časové intervaly. Žádný ze tří uvedených systémů nesplňoval všechny stanovené uživatelské požadavky (viz 2.2).

2.4 Shrnutí analýzy

V této části byla identifikována problémová doména současně používaná vedoucím systému. Jednalo se o proces plánování konzultačních termínů, následně bylo navrženo možné řešení tohoto problému – převedení tohoto procesu do jiného systému, který bude mít potřebnou funkcionalitu pro organizaci konzultačních termínů. Byly také identifikovány požadavky uživatelů na nový systém. Nebyl však nalezen žádný systém, který by plně pokryl před ním nastavené uživatelské požadavky. Proto bylo rozhodnuto vytvořit nový.

UID	Splnění
UP1	ano
UP2	ano
UP3	ano
UP4	částečně
UP5	ano
UP6	ano
UP7	ano
UP8	ne
UP9	ano
UP10	ano
UP11	ano
UP12	ano
UP13	ano
UP14	ne

(a) Reservio

UID	Splnění
UP1	ano
UP2	ano
UP3	ano
UP4	částečně
UP5	ano
UP6	ano
UP7	ano
UP8	ano
UP9	ano
UP10	ano
UP11	ano
UP12	ano
UP13	ano
UP14	ne

(b) Reenio

UID	Splnění
UP1	ano
UP2	ano
UP3	ne
UP4	ne
UP5	ano
UP6	ano
UP7	částečně
UP8	ano
UP9	ano
UP10	ano
UP11	ano
UP12	částečně
UP13	ano
UP14	ne

(c) Doodle

Tabulka 2.2: Splnění vypsaných uživatelských požadavků (viz sekci 2.1)

Kapitola 3

Návrh

Tato sekce je věnována návrhu našeho systému. Na základě uživatelských požadavků v předchozích kapitolách a analýze funkčnosti podobných systémů jsme stanovili požadavky na náš systém. Tyto požadavky byly rozděleny podle základní kategorizace na funkční a nefunkční. Pro podrobnější popis implementace funkčních požadavků v navrhovaném systému z pohledu uživatele byl vytvořen model případů užití.

3.1 Funkční požadavky

Hlavním úkolem funkčních požadavků je definovat funkčnost softwaru, kterou potřebujeme splnit.
Identifikovala jsem následující funkční požadavky pro náš systém:

F1 Evidence uživatelů

- Systém bude provádět identifikaci, autentizaci a autorizaci uživatelů v systému.
- Systém také uchová všechny osobní údaje uživatelů a umožní uživateli je v budoucnu spravovat.

F2 Evidence rezervací

- Systém bude ukládat informace o všech rezervacích, které může student nebo učitel učinit na konzultační termín.

F3 Evidence konzultačních termínů

- Systém bude ukládat informace o termínech konzultací a umožní je také spravovat.
- Při vytváření termínů systém zabrání kolizím a zohlední i dny pracovního klidu.

F4 Evidence čekací listiny

- Pokud se v určitém datu koná alespoň jeden termín konzultace, systém automaticky spojí čekací listinu s tímto datem.
- Uživatelé, kteří nejsou spokojeni s aktuálně dostupnými termíny konzultací, mohou být zařazeni na čekací listinu.
- Systém zkontroluje, zda je na datu volné místo, o které má student zájem, a zašle mu e-mailové upozornění.

F5 Evidence dnů pracovního klidu

- Systém bude ukládat informace o všech dnech pracovního klidu (název a časový interval).
- V systému lze spravovat dny pracovního klidu.
- Ve výchozím nastavení systém při vytváření termínu konzultace kontroluje, zda nejsou vytvořeny termíny, které připadají na dny pracovního klidu.

3.2 Model případů užití

Pro formalizaci funkčních požadavků budou použity diagramy použití (viz 3.1).
Diagramy použití jsou textové nebo grafické popisy interakce uživatele se systémem.
Základní prvky diagramu použití jsou:

Aktér

- Role objektu v jeho interakci se systémem,
- Aktérem může být člověk, jiný systémový software nebo hardwarové zařízení.

Případy užití

- Popisují posloupnost interakce mezi aktérem a systémem.

Vazby

- Sémantické vztahy mezi jednotlivými prvky modelu (viz 3.1).
  - Asociace

    Označuje, které případy užití mohou být použity jednotlivými aktéry.
  - Generalizace

Označuje, že některý prvek lze zobecnit na jiný prvek modelu případů užití.

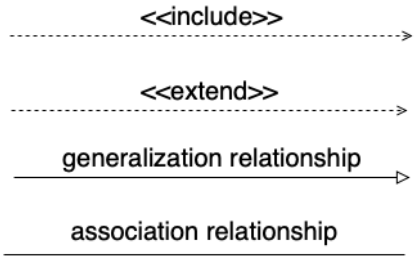
Prvek může být buď aktér, nebo případ užití.

* Zahrnout - Relace «Include»

  Označuje, že jeden případ užití zahrnuje další případ užití jako nedílnou součást.

* Rozšíření - Relace «Extend»

  Označuje, že jeden případ užití selektivně zahrnuje jiný případ užití.



Obrázek 3.1: Druhy vazeb

3.2.1 Aktéři

Aplikace je určena pro 3 skupiny uživatelů:

Student

– Oprávněný uživatel s právy studenta může v systému spravovat své rezervace a osobní údaje.

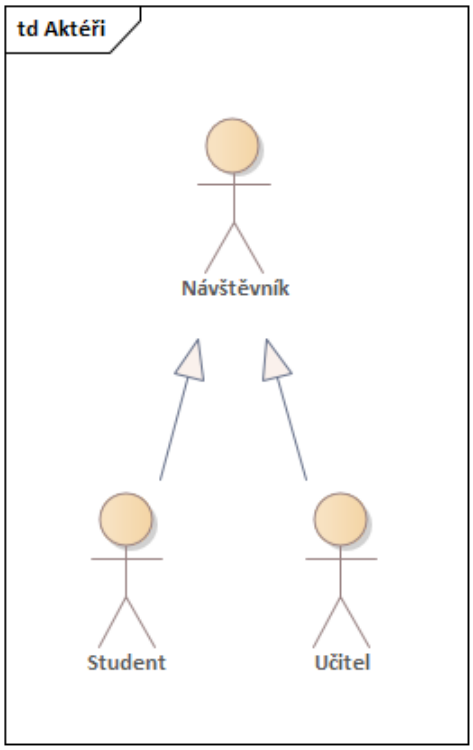
Učitel

– Oprávněný uživatel s právy učitele může řídit konzultační termíny a rezervace všech studentů a také spravovat vlastní osobní údaje.

Návštěvník

– Uživatel, který nebyl autorizován, je pro něj dostupná pouze registrační a přihlašovací stránka.

Učitel a student jsou s návštěvníkem v generalizačním vztahu. To znamená, že interagují se stejnou sadou případů použití jako návštěvník a interakce probíhá podle stejného scénáře (viz obrázek 3.2).



Obrázek 3.2: Vztahy mezi aktéry

3.2.1.1 Případy užití - učitel

V této části rozebereme všechny případy použití, kde je hlavním aktérem učitel (viz 3.3, 3.4, 3.5).

UC1 Zobrazení seznamu studentů – umožňuje zobrazit seznam všech studentů.

Příklad použití se spustí, když se učitel rozhodne zobrazit seznam všech studentů.

1. Systém vrátí stránku se seznamem všech studentů a jejich osobní údaje.

UC2 Přidání studenta – umožňuje definovat skupinu e-mailů studentů, kteří se mohou do systému zaregistrovat, a poslat jim registrační odkaz.

Příklad použití se spustí, když chce učitel studentovi poslat registrační odkaz.

1. Učitel stiskne tlačítko „Přidat studenta“.

2. Systém otevře formulář, který umožňuje vyplnit seznam e-mailů a také obsah dopisu, který student obdrží e-mailem s odkazem na registraci.
3. Učitel vyplní formulář.
4. Systém zkontroluje platnost vyplněných údajů a rozešle dopisy na emailové adresy s odkazem na registraci a poznámkou od vyučujícího.

UC3 Odstranění studenta – umožňuje odstranit studentský účet ze systému.

Příklad použití začíná, když chce učitel smazat studentský účet a všechny informace s ním spojené.

1. Učitel stiskne tlačítko „Odstranit“ vedle konkrétního studenta.
2. Systém smaže účet studenta a historii jeho rezervací.

UC4 Zobrazení osobních údajů – umožňuje zobrazit osobní údaje.

Příklad použití začíná, když se učitel rozhodne zobrazit své osobní údaje.

1. Systém učiteli otevře stránku, která obsahuje jeho osobní údaje jako je jméno, příjmení, telefonní číslo a emailová adresa

UC5 Změna osobních údajů – umožňuje změnit osobní údaje.

Příklad použití začíná, když se učitel rozhodne změnit osobní údaje.

1. Systém učiteli otevře formulář pro vyplnění jména, příjmení, email a telefonního čísla.
2. Učitel vyplní formulář.
3. Systém ověří formulář a změní osobní údaje učitele.

UC6 Zobrazení seznamu rezervací – umožňuje přístup k historii rezervací.

Příklad použití začíná, když se učitel rozhodne podívat se na historii rezervací.

1. Učitel si vybere rezervace konkrétního studenta nebo seznam všech rezervací.
2. Systém otevře historii rezervací a také odkazy na podrobnosti.

UC7 Zobrazení podrobnosti rezervace – umožňuje přístup k podrobnostem rezervace.

1. Učitel následuje odkaz s podrobnostmi rezervace.

2. Systém učiteli otevře stránku, která obsahuje následující informace o rezervaci:
   - **Student**
     * jméno studenta přihlášeného na konzultační termín.
   - **Stav**
     * stav aktuální rezervace (aktivní, zrušená, blokovaná).
   - **Vytvořil(a)**
     * Jméno uživatele, který vytvořil rezervaci.
   - **Zrušil(a)**
     * Jméno uživatele, který zrušil rezervaci.
   - **Datum**
     * čas zahájení konzultace
   - **Vytvořeno**
     * čas vytvoření rezervace
   - **Zrušeno**
     * čas zrušení rezervace

UC8 Vytvoření rezervací– umožňuje vytvoření rezervace

Příklad použití začíná, když se učitel rozhodne vytvořit rezervaci na konzultační termín.

1. Učitel klikne na tlačítko „Přihlásit“ vedle termínu, který ho zajímá.
2. Systém učiteli otevře modální okno s rozevíracím seznamem studentů.
3. Učitel si vybere studenta, kterého chce zapsat do konzultačního termínu.
4. Systém vytvoří rezervaci na vybraný termín konzultace.

UC9 Zrušení rezervací - umožňuje zrušení rezervaci.

Příklad použití začíná, když se učitel rozhodne zrušit rezervaci studenta.

1. Učiteli klikne na tlačítko „Odhlásit“ vedle termínu, který ho zajímá.
2. Systém zruší rezervaci.

UC10 Přidání poznámky k rezervaci – umožňuje zanechat poznámku týkající se konkrétní rezervace.

Příklad použití začíná, když chce učitel přidat poznámku týkající se rezervace.

1. Systém učiteli otevře formulář, kam může přidat poznámku nebo kde ji může upravit, pokud již existuje.

UC11 Přidání poznámky o studentovi – umožňuje zanechat poznámku týkající se konkrétního studenta.

Příklad použití začíná, když chce učitel přidat poznámku týkající se studenta.

1. Systém učiteli otevře formulář, kam může přidat poznámku nebo kde ji může upravit, pokud již existuje.

UC12 Zobrazení konzultačních termínů – umožňuje zobrazit rozvrh pro vybraný den.

Případ použití začíná, když se učitel rozhodne zobrazit rozvrh pro vybraný den.

1. Systém učiteli otevře kalendář pro aktuální měsíc.
2. Učitel vybere den v kalendáři.
3. Systém učiteli otevře rozvrh pro vybraný den.

UC13 Vytvoření konzultačních termínů – umožňuje vytvářet konzultační termíny.

Příklad použití začíná, když se učitel rozhodne vytvořit konzultační termín.

1. Systém učiteli otevře formulář s následujícími údaji:
   * **Časový interval**
   * **Hodinový interval**
   * **Opakování** (frekvence opakování, dny v týdnu)
   * **Student** (pokud chce uživatel okamžitě zapsat studenta)
   * **Délka konzultačního termínu**
   * **Počet konzultací**
   * **Zahrnutí dnů pracovního klidu**
2. Učitel vyplní formulář.
3. Systém ověří formulář a vytvoří konzultační termín.

UC14 Odstranění konzultačních termínů – umožňuje odstranit konzultační termíny.

Příklad použití začíná, když se učitel rozhodne odstranit termín konzultace.

1. Systém učiteli otevře formulář s následujícími údaji:
   * **Časový interval**

    - **Hodinový interval**
    - **Opakování** (frekvence opakování, dny v týdnu)
2. Učitel vyplní formulář.
3. Systém zkontroluje formulář a odstraní všechny termíny konzultace, které se překrývají se zadaným časovým intervalem, a také rezervace s nimi spojené.

UC15 Počítání počtu termínů konzultace nebo délky jejich trvání - umožňuje vypočítat počet termínů konzultace nebo dobu trvání jedné konzultace.

Příklad použití začíná, když uživatel před odesláním formuláře pro vytvoření termínu konzultace chtěl znát počet termínů nebo dobu trvání jedné konzultace.

1. Učitel klikne na tlačítko „Spočítat“.
2. Systém učiteli vrátí počet termínů nebo délku jejich trvání.

UC16 Blokování konzultačních termínů - umožňuje zablokovat termíny konzultace i aktivní rezervaci na tento termín (pokud existuje).

Příklad použití začíná, když se učitel rozhodne zablokovat termín.

1. Učitel klikne na tlačítko „Zrušit“.
2. Systém otevře modální okno s formulářem, ve kterém je potřeba vybrat alespoň jeden konzultační termín pro vybraný den a můžete zanechat poznámku pro studenty.
3. Učitel vyplní formulář.
4. Systém zkontroluje formulář a uzavře termíny pro záznam a pokud již byl student na termín přihlášen, tak jeho rezervace bude zablokována a o zablokování termínu obdrží upozornění s poznámkou vyučujícího.

UC17 Zobrazení čekací listiny - umožňuje zobrazit obsah (nebo spíše jména studentů) čekací listiny pro vybraný den.

Příklad použití začíná, když se učitel rozhodne prohlédnout jména studentů na čekací listině.

1. Systém učiteli otevře seznam všech konzultačních termínů pro vybraný den.
2. Učitel klikne na tlačítko „Pořadníky“.
3. Systém zobrazí učiteli seznam všech studentů na čekací listině.

UC18 Zobrazení dnů pracovního klidu - umožňuje zobrazit seznam dny pracovního klidů.

– Systém učiteli otevře seznam dnů pracovního klidu. Seznam obsahuje informace o názvu, začátku a konci časového intervalu.

UC19 Přidání dnů pracovního klidu - umožňuje přidat dny pracovního klidu.

Případ užití začíná, když se učitel rozhodne přidat do systému dny pracovního klidu.

1. Systém pošle učiteli formulář, ve kterém je třeba vyplnit následující pole:
   * **Název události**
   * **Časový interval**
2. Učitel vyplní formulář.
3. Systém je přidá do systému a nastaví jejich roční opakování.

UC20 Odstranění dnů pracovního klidu - umožňuje odstranit dny pracovního klidu.

Případ užití začíná, když se učitel rozhodne odstranit ze systému dny pracovního klidu.

1. Učitel klikne na tlačítko „Smazat“ vedle nepracovního dne.

3.2.1.2 Případy užití - student

V této části rozebereme všechny případy použití, kde je hlavním aktérem student (viz 3.6).

UC21 Zobrazení osobních údajů/rezervací - umožňuje zobrazit osobní údaje/seznam rezervací.

Příklad použití začíná, když se student rozhodne zobrazit své osobní údaje nebo seznam svých rezervací.

1. Systém studentovi otevře stránku, která obsahuje jeho osobní údaje, jako je jméno, příjmení, telefonní číslo a e-mailová adresa, a také seznam všech jeho rezervací. U každé rezervace v tomto seznamu je odkaz na podrobnosti této rezervace.

UC22 Změna osobních údajů - umožňuje změnit osobní údaje.

Příklad použití začíná, když se student rozhodne změnit osobní údaje.

1. Systém studentovi otevře formulář pro vyplnění jména, příjmení a telefonního čísla.
2. Student vyplní formulář.
3. Systém ověří formulář a změní osobní údaje studenta.

UC23 Vytvoření rezervací - umožňuje studentovi vytvářet rezervace.

Příklad použití začíná, když se student rozhodne vytvořit rezervaci na konzultační termín.

1. Student klikne na tlačítko „Přihlásit“ vedle termínu, který ho zajímá.
2. Systém vytvoří rezervaci na vybraný termín konzultace.

UC24 Zrušení rezervací - Umožňuje studentovi zrušit rezervaci.

Příklad použití začíná, když se student rozhodne zrušit rezervaci.

1. Student klikne na tlačítko „Odhlásit“ vedle termínu, který ho zajímá.
2. Systém zruší rezervaci.

UC25 Přenos rezervací - umožňuje přenos rezervaci.

Příklad použití začíná, když se student rozhodne přenést svou rezervaci na aktuální den (protože rezervaci na aktuální den nelze zrušit, pouze přenést).

1. Student klikne na tlačítko „Přesun“ vedle termínu, na který je přihlášen.
2. Systém mu otevře modální okno se všemi dostupnými konzultačními termíny pro aktuální den.
3. Student vybere termín.
4. Systém přenese jeho rezervaci na vybraný termín.

UC26 Přidání poznámky k rezervaci - umožňuje zanechat poznámku týkající se konkrétní rezervace.

Příklad použití začíná, když chce student přidat poznámku týkající se rezervace

1. Během vytváření rezervace student klikne na tlačítko „Přidat“. V modálním okně, které se otevře, se zobrazí pole pro přidání poznámky (kterou nelze v budoucnu upravit).

UC27 Zobrazení konzultačních termínů - umožňuje studentovi zobrazit rozvrh pro vybraný den.

Případ použití začíná, když se student rozhodne zobrazit rozvrh pro vybraný den.

1. Systém studentovi otevře kalendář pro aktuální měsíc.
2. Student vybere den v kalendáři.
3. Systém uživateli otevře rozvrh pro vybraný den.

UC28 Přidání uživatele do čekací listiny - umožňuje zařadit se na čekací listinu na konkrétní datum a přijímat e-mailová upozornění, pokud se uvolní místo.

Příklad použití začíná, když se student rozhodne zařadit do čekací listiny.

1. Systém uživateli otevře seznam všech termínů pro vybraný den.
2. Student klikne na tlačítko „Pořadník“.
3. Systém zařadí studenta do čekací listiny.
4. Systém odešle uživateli e-mail, pokud se místo uvolní, a bude odstraněn z čekací listiny.

3.2.1.3 Případy užití - návštěvník

V této části rozebereme všechny případy použití, kde je hlavním aktérem návštěvník (viz 3.7).

UC29 Registrace - umožňuje zaregistrovat uživatele do systému.

Příklad použití začíná, když uživatel obdrží e-mail s odkazem na registrační stránku

1. Uživatel klikne na odkaz v e-mailu s pozvánkou.
2. Systém otevře registrační formulář, který vám umožní zadat jméno, příjmení, e-mail, telefonní číslo (nepovinné), heslo a opakování hesla.
3. Uživatel vyplní formulář.
4. Systém zkontroluje platnost formuláře a pošle studentovi odkaz na zadaný e-mail pro dokončení registrace.
5. Student následuje odkaz, který obdržel na svůj e-mail.
6. Systém přidá do systému uživatelský účet.

UC30 Přihlášení - umožňuje uživateli přístup ke svému účtu v systému

Příklad použití začíná, když uživatel, který je v systému neoprávněný, vstoupí na jakoukoliv stránku poskytovanou systémem (kromě stránek Rezervace a Přihlášení).

Uživatel si pamatuje heslo ze svého účtu

1. Systém otevře formulář, který vám umožní zadat e-mail a heslo.
2. Uživatel vyplní formulář.
3. Systém zkontroluje platnost formuláře, po úspěšné validaci autorizuje uživatele v systému a přesměruje uživatele na stránku, na kterou se uživatel původně chtěl dostat.

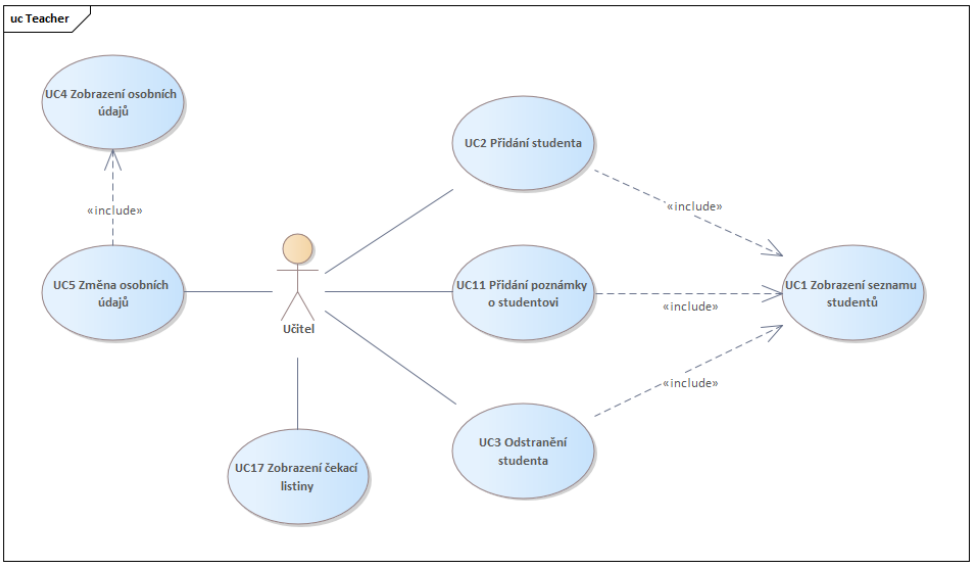
Uživatel zapomněl heslo k účtu

1. Systém uživateli otevře přihlašovací formulář, v jehož spodní části je odkaz "Zapomněli jste své heslo?".
2. Uživatel následuje odkaz.

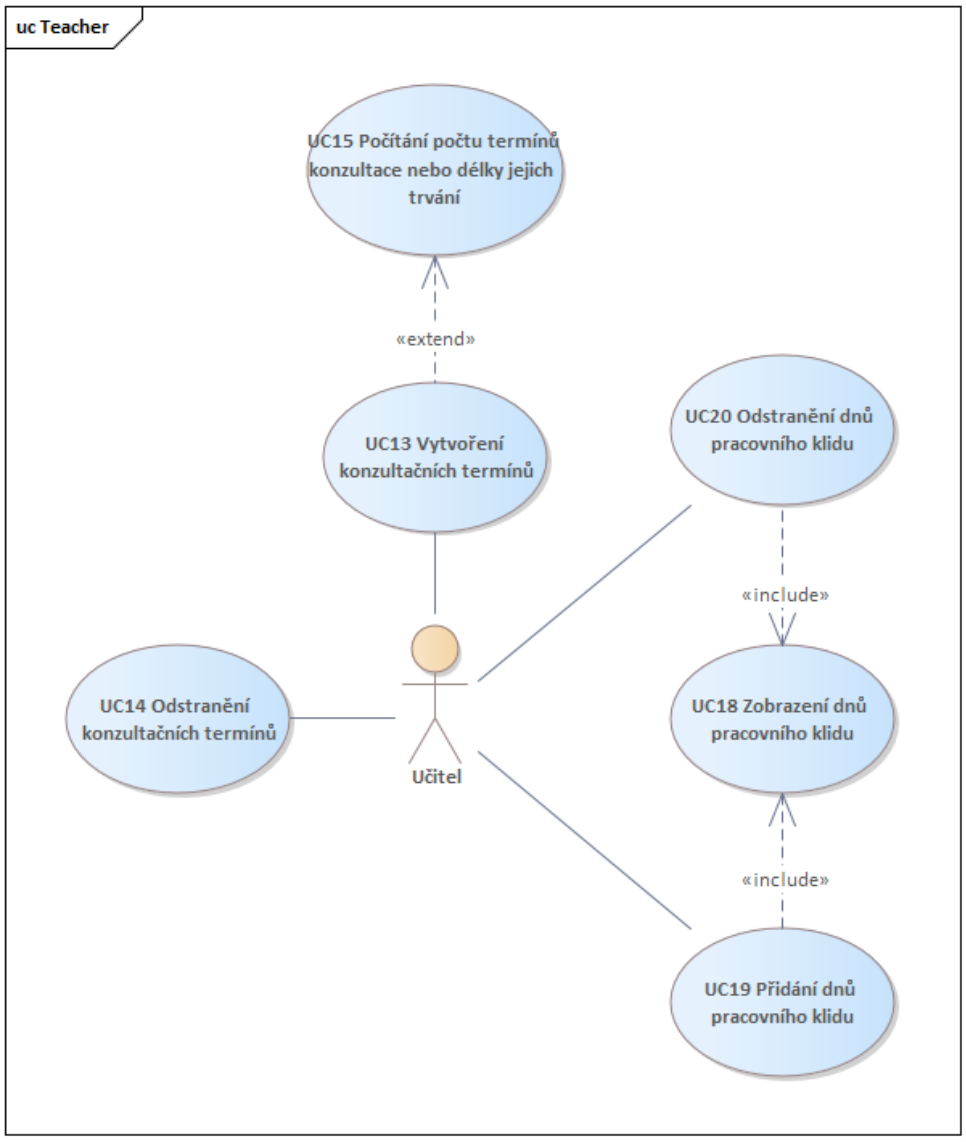
UC31 Resetování hesla - umožňuje uživateli resetovat aktuální heslo a nastavit nové v případě, že uživatel zapomene heslo k účtu.

Příklad použití začíná, když uživatel zapomene heslo do svého systému.

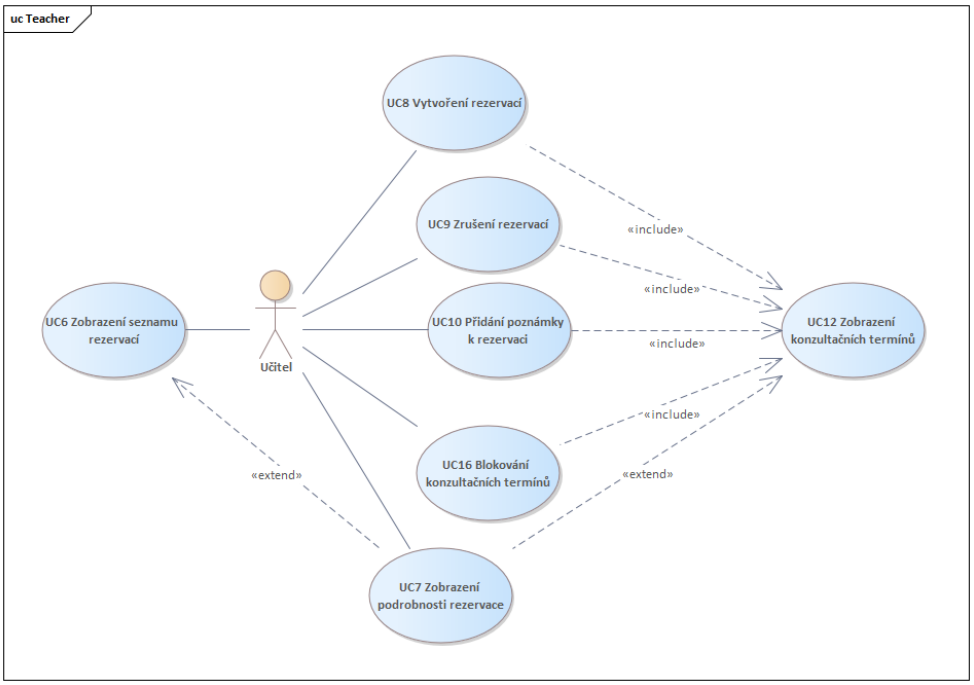
1. Systém otevře uživateli formulář, do kterého stačí zadat pouze e-mailovou adresu.
2. Student vyplní formulář.
3. Systém ověří formulář a odešle e-mail s odkazem na obnovení hesla a poté zobrazí upozornění, že by si student měl zkontrolovat svou e-mailovou adresu.
4. Uživatel obdrží e-mail s odkazem pro obnovení hesla a následuje jej.
5. Systém otevře formulář pro resetování hesla, kde je potřeba vyplnit pole s novým heslem.
6. Uživatel vyplní formulář.
7. Systém ověří formulář a změní heslo uživatelského účtu.



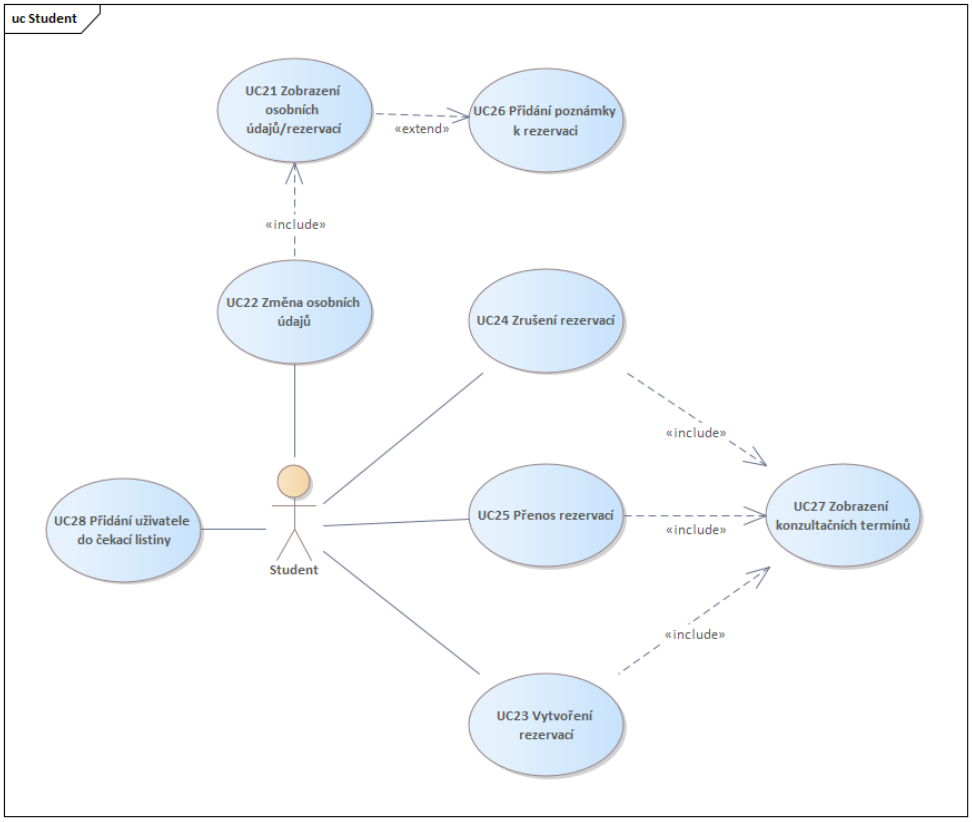
Obrázek 3.3: Diagram případů užití - učitel



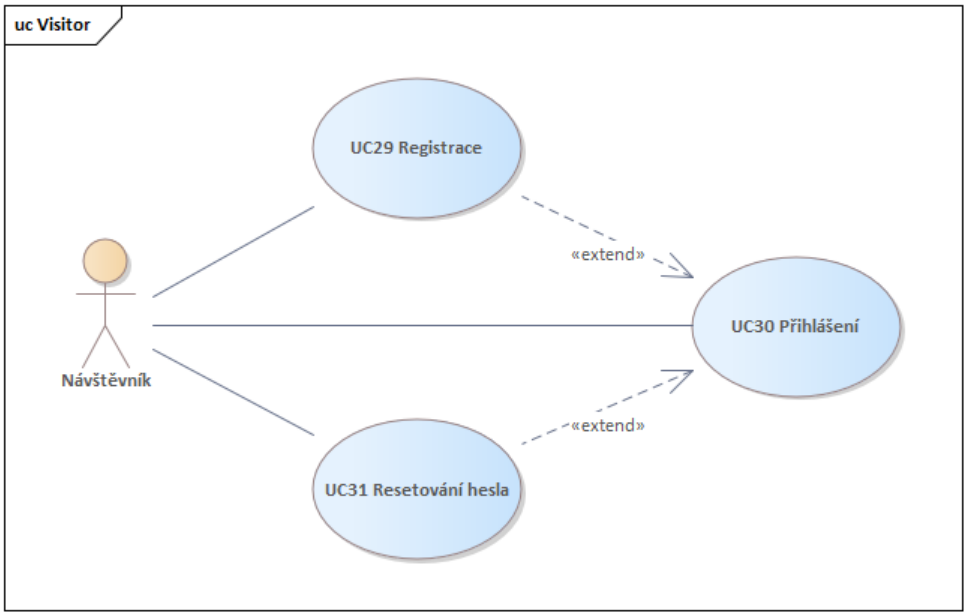
Obrázek 3.4: Diagram případů užití - učitel



Obrázek 3.5: Diagram případů užití - učitel



Obrázek 3.6: Diagram případů užití - student



Obrázek 3.7: Diagram případů užití - návštěvník

	F1	F2	F3	F4	F5
UC1	✓				
UC2	✓				
UC3	✓				
UC4	✓				
UC5	✓				
UC6		✓			
UC7		✓			
UC8		✓			
UC9		✓			
UC10		✓			
UC11	✓				
UC12			✓		
UC13			✓		
UC14			✓		
UC15			✓		
UC16		✓	✓		
UC17				✓	
UC18					✓
UC19					✓
UC20					✓
UC21	✓	✓			
UC22	✓				
UC23		✓			
UC24		✓			
UC25		✓			
UC26		✓			
UC27			✓		
UC28				✓	
UC29	✓				
UC30	✓				
UC31	✓				

Tabulka 3.1: Pokrytí funkčních požadavků případy užití

3.3 Nefunkční požadavky

Nefunkční požadavky popisují požadavky na vlastnosti našeho systému a také omezení, která musí splňovat.
Identifikovala jsem následující funkční požadavky pro náš systém:

N1 Bezpečnost

- Musí být přijata ochranná opatření k zabezpečení systému před nejběžnějšími typy kybernetických útoků (SQLI, CSRF).
- Systém by neměl umožnit uživatelům, kteří nemají potřebná práva, přístup k určitým zdrojům webové aplikace a provádění řady určitých akcí.

N2 Jednoduchá použitelnost

- Naším hlavním cílem je, aby byl návrh naší webové aplikace intuitivní. Předpokládá se zde, že se uživatel může soustředit na svůj úkol, aniž by byl rozptylován dlouhým hledáním určitých sekcí, funkcí. Usnadníme tak uživateli práci s naším systémem.

N3 Snadná provozovatelnost

- Nevyžaduje žádný speciální hardware pro provoz.
- K získání přístupu do systému bude uživateli stačit pouze webový prohlížeč a přístup k internetu.

N4 Bezplatné použití

Uživatelé mohou spouštět kód zdarma, bez nutnosti nákupu licence. Během implementace systému nebudou využívány programy s placenou licencí. Přednost bude dána bezplatnému a open source softwaru.

3.4 Shrnutí návrhu

Součástí návrhu bylo identifikovat funkční a nefunkční požadavky na systém a také případy užití. Na základě systémových požadavků, které byly identifikovány v této kapitole, můžeme vybrat vhodné technologie.

Kapitola 4

Výběr technologií

Tato sekce je věnována výběru vhodných technologií pro vývoj našeho softwaru.

Začněme tím, že náš systém bude webová aplikace. Pokud jde o webovou aplikaci, lze k ní přistupovat z jakéhokoli zařízení, které má webový prohlížeč a přístup k internetu.

Webové aplikace se skládají ze serverové části (backend) a klientské části (frontend). Uživatelé komunikují s klientskou stranou prostřednictvím rozhraní, které se zobrazuje v prohlížeči (Chrome, Firefox, Safari atd.). Klientská část na příkaz uživatele odešle požadavek na server přes internet. Server požadavek zpracuje a vrátí klientovi odpověď.[28]

V následujících kapitolách budou analyzovány nejoblíbenější technologie, které se používají k vývoji webových aplikací na straně serveru a klienta. Budeme také analyzovat hlavní typy kybernetických útoků a jak před nimi chránit náš systém. Výstupem této kapitoly budou technologie, které byly zvoleny pro implementaci naší webové aplikace. Výběr provedeme na základě srovnání výhod a nevýhod technologií a jejich souladu s našimi nefunkčními požadavky.

4.1 Backend

Pro vývoj webových aplikací na straně serveru se používá mnoho programovacích jazyků: Java, PHP, .NET, Python a další. Ale rozhodla jsem se zůstat u Pythonu, protože na webu je obrovské množství tutoriálů, díky kterým je velmi snadné se to naučit. Můžete si také všimnout následujících výhod tohoto jazyka:

- multiplatformní
- obrovské množství knihoven
- jednoduchá syntaxe

Poté, co jsme se rozhodli pro jazyk serveru. Musíme zvolit framework pro vývoj webu na straně serveru. Framework bude vybrán na základě následujících parametrů: otevřený software, nízký vstupní práh, velká komunita a podrobná dokumentace. Nejoblíbenější jsou Django, Flask a Pyramid. Dále se budeme podrobněji zabývat každým z nich.

4.1.1 Django

Django[22] je nejpopulárnější framework Pythonu. Obsahuje plně funkční prostředí pro vývoj aplikací. Plná funkčnost znamená, že většina nástrojů, které můžeme potřebovat k vytvoření aplikace, je již součástí frameworku (na rozdíl od flask nebo pyramid není potřeba zahrnout další knihovny).

Django podporuje návrhový vzor MVC. MVC umožňuje vývojářům pracovat samostatně na business logice a vizuální reprezentaci aplikace.

Tento rámec implementuje ORM. Avšak podle mého názoru je svou flexibilitou horší než SQLAlchemy, kterou lze snadno připojit k flask nebo pyramid pomocí speciálního rozšíření.

Z výhod lze také vypozorovat následující vlastnosti:

- dodržuje princip DRY,
- poskytuje ochranu proti útokům SQLI, XSS, CSRF.

Avšak pokud jde o malý projekt, těžká monolitická struktura tohoto rámce je nadbytečná.

4.1.2 Flask

Flask je mikrorámec Pythonu pro webové aplikace využívající sadu nástrojů Werkzeug a šablonovací engine Jinja2.

Koncept „microframework“ implikuje absenci nástrojů a knihoven, které jsou typické pro jiné frameworky. Nemá žádnou abstraktní databázovou vrstvu, žádné ověřování formulářů a další. Vzhledem k tomu, že Flask byl původně navržen jako rozšiřitelný, ponechal vývojářům svobodu vybrat si balíčky třetích stran nebo vytvořit vlastní. Komunita také předkládá sérii rozšiřujících balíčků pro Flask. Jejich název obvykle začíná na flask (flask-wtf, flask-login, flask-sqlalchemy).
Mezi výhody lze zařadit následující vlastnosti:

- flexibilita,
- podpora zabezpečených cookies,
- rozsáhlá dokumentace,
- automatické testování,

- vestavěné jednotkové testování, vývojový server, debugger a obsluha požadavků,
- kompatibilní s Google App Engine.

4.1.3 Pyramid

Jedná se o open source framework, který se používá k vývoji malých i velkých webových aplikací. Pyramidová struktura byla inspirována Zope, Pylons (verze 1.0) a Django.

- Koncepty rozšiřitelnosti, bypassu a deklarativní bezpečnosti byly vypůjčeny od Zope.
- Přístup k odesílání URL, žádná politika (uživatel se rozhoduje, kterou databázi nebo šablonovací systém použít) byl vypůjčen od Pylons.
- Koncept prezentace a rozsáhlé dokumentace byl vypůjčen od Django.[6]

Mezi výhody lze zaznamenat následující vlastnosti:

- rychlost
  - považován za nejrychlejší známý webový rámec Pythonu,
- škálovatelnost,
- vhodné pro projekty jakékoliv velikosti,
- vestavěné relace.

Hlavní nevýhodou je, že tento framework je méně populární než Flask a Django. Z tohoto důvodu má malou komunitu. To znamená, že je velká pravděpodobnost, že pokud se objeví problém ve fázi implementace softwaru, nelze najít řešení na internetu.

4.2 Frontend

4.2.1 Bootstrap 5

Ve svém projektu použiji Bootstrap 5[21] pro vývoj frontendu. Hlavním důvodem výběru jsou výborné předchozí zkušenosti. Bootstrap je open source a bezplatný HTML, CSS a JS rámec, který poskytuje webovým vývojářům sadu nástrojů pro rychlé vytváření responzivních návrhů webových stránek a webových aplikací.[31]
Bootsrap 5 obsahuje následující prvky:

- nástroje rozvržení – obalové kontejnery, mřížkové systémy, adaptivní třídy,
- hotové komponenty – tlačítka, formuláře, navigační lišty, modální okna, rozevírací seznamy atd.,
- třídy stylování obsahu – tlačítka, obrázky, tabulky, formuláře, textová pole atd.,
- další třídy zodpovědné za zarovnání textu, zobrazení a skrytí prvků, nastavení barev a odsazení.[5]

Hlavní výhody tohoto rámce jsou:

- Kompatibilita s prohlížečem

  Zobrazuje se stejně ve všech prohlížečích.

- Přizpůsobivost

  – Návrh webové stránky se zobrazuje správně při jakémkoli rozlišení obrazovky.

- Jednota stylů

  – Bootstrap prvky jsou vyrobeny ve stejném stylu, díky čemuž se vzájemně harmonicky kombinují.

4.3 Bezpečnost

Jedním z našich nefunkčních požadavků je bezpečnost. V této části analyzujeme hlavní kybernetické útoky, kterým mohou naše aplikace čelit, a jak jim předcházet.

4.3.1 CSRF

Podstatou CSRF útoku je, že útočník jménem uživatele autorizovaného v systému posílá příkazy na server. Tento typ kybernetického útoku může ovlivnit jakoukoli webovou aplikaci, která používá cookies, autorizační certifikáty a autentizaci prohlížeče.[7] Jinými slovy všechny aplikace, které k požadavku přidávají ověřovací údaje uživatele. Pojďme se podívat na nejčastější příklad – pomocí phishingového odkazu. Útočník poslal uživateli odkaz v e-mailu. Aby odkaz nevypadal podezřele, je v tagu obrázku zkrácen nebo skrytý. Kliknutím na odkaz prohlížeč uživatele spustil JS kód útočníka. Kód JS odeslal požadavek na server, kde je náš uživatel již přihlášen. Předpokládejme, že převedeme peníze z bankovního účtu. Bankovní server kontroluje cookies, aby se ujistil, že požadavek přišel od oprávněného uživatele a zpracovává jej. Požadavek tedy skutečně přišel od uživatele, ale odeslal jej neúmyslně.

4.3.1.1 CSRF token

Jedním ze způsobů ochrany proti útokům CSRF je použití CSRF tokenů. Server pro požadavek vygeneruje náhodný token a odešle jej do prohlížeče uživatele. Pokud chce uživatel provést nějaké změny, musí spolu s požadavkem poslat token. Pokud se token shoduje, relace pokračuje a server akci provede, jinak relaci přeruší.[7] Například HTML formuláře jsou odesílány na server pomocí metody požadavku POST. Metoda POST je považována za nebezpečnou, protože mění stav serveru. Proto se považuje za dobrou praxi předat token klientovi ve skrytém poli formuláře (viz 4.1). Token bude zahrnut jako parametr požadavku při odeslání formuláře na server.

```
1 <form method="post" action="">
2 {{ form.hidden_tag() }}
3 {{ form.old_password }}
4 {{ form.new_password }}
5 </form>
```

Listing 4.1: Formulář obsahující CSRF token ve skrytém poli

CSRF token musí mít následující vlastnosti:

- jedinečný pro každou operaci,
- jednorázový,
- velikost odolná proti útoku hrubou silou,
- generován kryptograficky bezpečným generátorem pseudonáhodných čísel,
- má omezenou životnost.[7]

4.3.2 SQLI

SQLI označuje kybernetické útoky založené na neoprávněném vkládání úmyslného kódu SQL do dat. Když uživatel odešle svá data na server, server je použije v dotazech SQL. Prostřednictvím těchto dat SQLI spustí svůj škodlivý SQL kód.

V této části se podíváme na tři hlavní typy útoků založených na SQLI:

- Classic SQLI
- Error-based SQLI
- Blind SQLI

4.3.2.1 Classic SQLI

Podstatou tohoto útoku je, že uživatel změní SQL výraz pomocí dat, která odešle na server (formuláře nebo URL). Řekněme, že se uživatel rozhodne přihlásit ke svému účtu. Server odešle do databáze následující dotaz 4.2. Dotaz vrátí všechny záznamy, které odpovídají danému výrazu.

```
1 SELECT * FROM user
2 WHERE username='some_username' and password='some_password';
```

Listing 4.2: SQL dotaz, který vrátí uživatele s odpovídajícím uživatelským jménem a heslem.

Nyní řekněme, že útočník zadal následující údaje:

username = random_username' OR 1=1;–

password = random_password

Kde **random_username**, **random_password** (viz výše) je náhodná sada znaků. Poté bude dotaz, který odešleme do databáze, vypadat takto: 4.3. Databáze vrátí všechny řádky, kde **username**='random_username' nebo 1=1 , protože druhý příkaz je vždy pravdivý, db vrátí všechny řádky. „- -“ tato sekvence znaků je vnímána jako začátek jednořádkového komentáře, to znamená, že část požadavku na ověření hesla byla jednoduše okomentována.

```
1 SELECT * FROM user
2 WHERE username='random_username' OR 1=1;--' and password=
    random_password;
```

Listing 4.3: SQL dotaz, který vrátí uživatele s odpovídajícím uživatelským jménem a heslem.

Útočníkovi takový útok utekl, protože požadavek byl vytvořen přímo z uživatelských dat (tzv. raw request). Za správnou praxi se považuje používání parametrizovaných dotazů. Při parametrizaci dotazů jsou všechny doslovné hodnoty obsažené v dotazu nahrazeny parametry. Díky tomu se nejprve v

databázi vytvoří plán provádění dotazu a poté jsou do databáze předány parametry, které již plán provádění nemohou ovlivnit, protože již byl vytvořen. Nyní SQL dotaz po parametrizaci vypadá takto: 4.4.

```
SELECT * FROM user
WHERE username=@username and password=@password;
```

Listing 4.4: Parametrizovaný SQL dotaz

4.3.2.2 Error-based SQLI

Error-based SQLI používá chybové zprávy vyvolané databázovým serverem. Způsobením takové chyby může útočník získat informace o struktuře databáze.
Řekněme, že stránka poskytuje uživateli adresář služeb (viz zdrojový kód 4.5). Uživatel následuje odkaz, aby získal podrobné informace o konkrétní službě, v tu chvíli server obdrží požadavek **GET** na url adresu **/service/<service_id>**. Kde je místo **<service_id>** nějaké číslo. Jde nám o dynamickou url adresu, protože se skládá z měnící se části – proměnné **service_id**. Dále je proměnná předána mapovací funkci **service_info**. Hodnota proměnné určuje jedinečný identifikátor služby. Na jeho základě chceme přistupovat k objektu třídy **Service**. Pokud ručně zadáme url adresu **/service/some_string** do adresního řádku, dostaneme chybu. Faktem je, že naše databáze očekává, že obdrží číselnou hodnotu, protože dříve jsme při vytváření tabulky **Service** určili, že **service_id** je číselná hodnota, nikoli řetězec.

```
@app.route('/service/<service_id>/')
def service_info(service_id):
  service = Service.query.get(service_id)
  service_description = service.description
    return 'f<h1>{service_description}</h1'>
```

Listing 4.5: Funkce zobrazení, která nám vrací informace o konkrétní službě.

Hlášení chyb by proto nemělo zveřejňovat citlivé informace. Zobrazení SQL dotazů, které způsobily chybu, je relevantní pouze ve fázi vývoje.

4.3.2.3 Blind SQLI

Pokud se vývojář postaral o to, aby se uživatelům nezobrazovaly informace o SQL dotazech, útočníci se mohou pokusit jednat slepě. Pojďme analyzovat příklad, který jsme zvažovali dříve (viz zdrojový kód 4.5). Řekněme, že adresář zobrazuje odkazy na 5 z 6 dostupných služeb. Iterací přes proměnnou **service_id** v url adrese může uživatel získat přístup k informacím o službě 6. Proto je nutné před použitím v SQL dotazech zkontrolovat data, která od uživatele obdržíme.

4.3.2.4 Ochrana proti SQL útokům

Abychom se uchránili alespoň před těmi nejzákladnějšími typy SQLI, měli bychom:

- ověřit všechna data přijatá od uživatele,
- nepoužívat "raw"SQL dotazy
  - dotazy musí být parametrizovány,
    - můžete použít vrstvu ORM, která také používá parametrizované dotazy,
- v chybových zprávách nezobrazovat důvěrné informace.

4.4 Shrnutí výběru technologií

Pro implementaci serverové části jsem preferovala rámec Flask. I když je Django populárnější, vzhledem k mým malým zkušenostem v oblasti vývoje webu je pro nezkušené vývojáře docela obtížné ho zvládnout. Proto zůstala volba mezi Flask a Pyramid. Ale protože Flask má mnohem větší komunitu, moje volba nakonec padla na něj. Pro zabezpečení proti útokům SQLI a CSRF připojíme k Flask následující rozšíření:

- Flask-SQLAlchemy
  - SQLAlchemy je známá svým ORM.
  - Jak bylo zmíněno v subsubsection 4.3.2.4, ORM nás chrání před určitými typy SQLI.
- Flask-WTF
  - Používá CSRF tokeny k ochraně formulářů před CSRF útoky (viz subsection 4.3.1).

Více podrobností o těchto rozšířeních lze nalézt v subsection 5.2.1 a subsection 5.2.2.

Co se týče frontendu, Bootstrap nám vyhovuje. Jediným nefunkčním požadavkem na návrh uživatelského rozhraní naší webové aplikace je "jednoduchá použitelnost". Na tomto frameworku bylo napsáno poměrně velké množství stránek. Vzhledem k šablonovité povaze frameworku jsou si všechny podobné (stejná struktura, navigace, tlačítka).[29] Uživatel tedy s největší pravděpodobností již měl zkušenost se stránkou implementovanou pomocí Bootstrapu, takže předchozí zkušenost mu může usnadnit orientaci na našem webu. V otázce intuitivnosti naší webové aplikace bude samozřejmě hodně záležet na nás.

Kapitola 5

Implementace

Díky předchozím kapitolám, kde jsme provedli návrh našeho systému a vybrali vhodné technologie, můžeme přistoupit k implementaci. V této části popíšeme proces vytváření klientské a serverové části naší aplikace a také strukturu zdrojového kódu našeho systému.

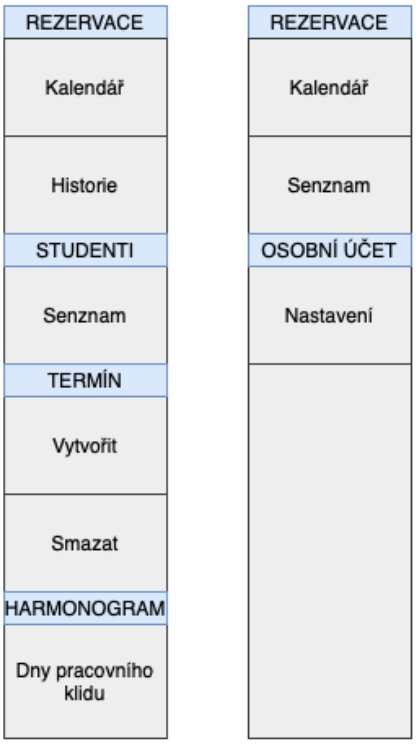
Pro vývoj backendu jsme zvolili Flask. Jak už ale bylo zmíněno dříve, Flask je mikrorámec a jeho možnosti nám na vývoj plnohodnotné webové aplikace stačit nebudou, proto využijeme i rozšíření, která budou také popsána v této kapitole.

5.1 Frontend

Pro vývoj klientské části byla použita již hotová Bootstrap 5 startovací šablona pro tvorbu webových aplikací[10]. V projektu byly také použity hotové objekty rozhraní a třídy frameworku **Bootstrap 5**[21]. **JS** byl použit pro přidání interaktivity do prvků webové aplikace. Pro zpracování šablon jsem použila Jinju, která je již vestavěná ve Flasku.

5.1.1 Jinja

Jinja[23] je šablonový engine pro programovací jazyk Python. Šablonový engine je nástroj, který zjednodušuje proces psaní značek, jejich rozdělení na komponenty a jejich vázání na data. Hlavní výhodou šablonových enginů je to, že eliminují potřebu vícekrát psát opakující se kód. Jedním z hlavních prvků Jinja, které jsem použila při psaní aplikace, byl prvek dědičnosti šablony. Téměř všechny stránky na našem webovém serveru mají společný boční navigační panel, ale protože má učitel k dispozici více sekcí, boční navigační panel pro studenty a učitele se výrazně liší (viz 5.1). Vytvořila jsem tedy dvě základní šablony pro studenta a učitele. Základní šablona obsahuje běžné prvky webu (skripty, styly, navigační lišta). Podřízené šablony dědí a rozšiřují základní šablonu pomocí příkazu **extends**.



Obrázek 5.1: Navigační lišta učitele a studenta

5.2 Backend

K vývoji serverové strany aplikace byl použit mikrorámec Flask. Jako mikrorámec nedokáže Flask sám o sobě téměř nic, ale jeho jádro lze škálovat pro různé úkoly. Za tímto účelem byly zahrnuty balíčky rozšíření. Tato část popisuje téměř všechny balíčky rozšíření, které byly dodatečně použity.

5.2.1 Flask-SQLAlchemy

Flask nemá specifický databázový systém ani ORM, proto jsme použili rozšíření Flask-SQLAlchemy[24].

Flask-SQLAlchemy je rozšíření Flask, které integruje SQLAlchemy do Flasku.

SQLAlchemy je programovací knihovna Pythonu pro práci s relačními databázemi pomocí technologie ORM.

ORM je způsob přístupu k relační databázi pomocí objektově orientovaného jazyka.

SQLAlchemy umožňuje popsat databázové struktury a způsoby interakce s nimi přímo v Pythonu. SQLAlchemy také podporuje následující typy databází:

MySQL, PostgreSQL, Oracle, MS-SQL, SQLite a další. Jedním z hlavních důvodů pro použití SQLAlchemy je její flexibilita a odolnost vůči SQLI.

5.2.1.1 Vytváření modelů

Model je třída v Pythonu, která představuje databázovou tabulku. Jeho atributy jsou mapovány do sloupců tabulky. Třída modelu dědí z db.Mobel a definuje sloupce jako instance třídy db.Column.[32] Všechny naše modely jsou uloženy v souboru models.py. Obrázek 5.1 ukazuje model třídy Consultationterm.

```
1  class ConsultationTerm(db.Model):
2      id = db.Column(db.Integer, primary_key=True)
3      date = db.Column(db.Date, nullable=False)
4      start_time = db.Column(db.Time, nullable=False)
5      end_time = db.Column(db.Time, nullable=False)
6      state = db.Column(db.String(20), nullable=False)
7
8      reservations = db.relationship('Reservation', backref='
       time_window', lazy=True, passive_deletes=True)
9
10     def __init__(self, date, start_time, end_time):
11         self.date = date
12         self.start_time = start_time
13         self.end_time = end_time
14         self.state = 'free'
```

Listing 5.1: Třída Consultationterm

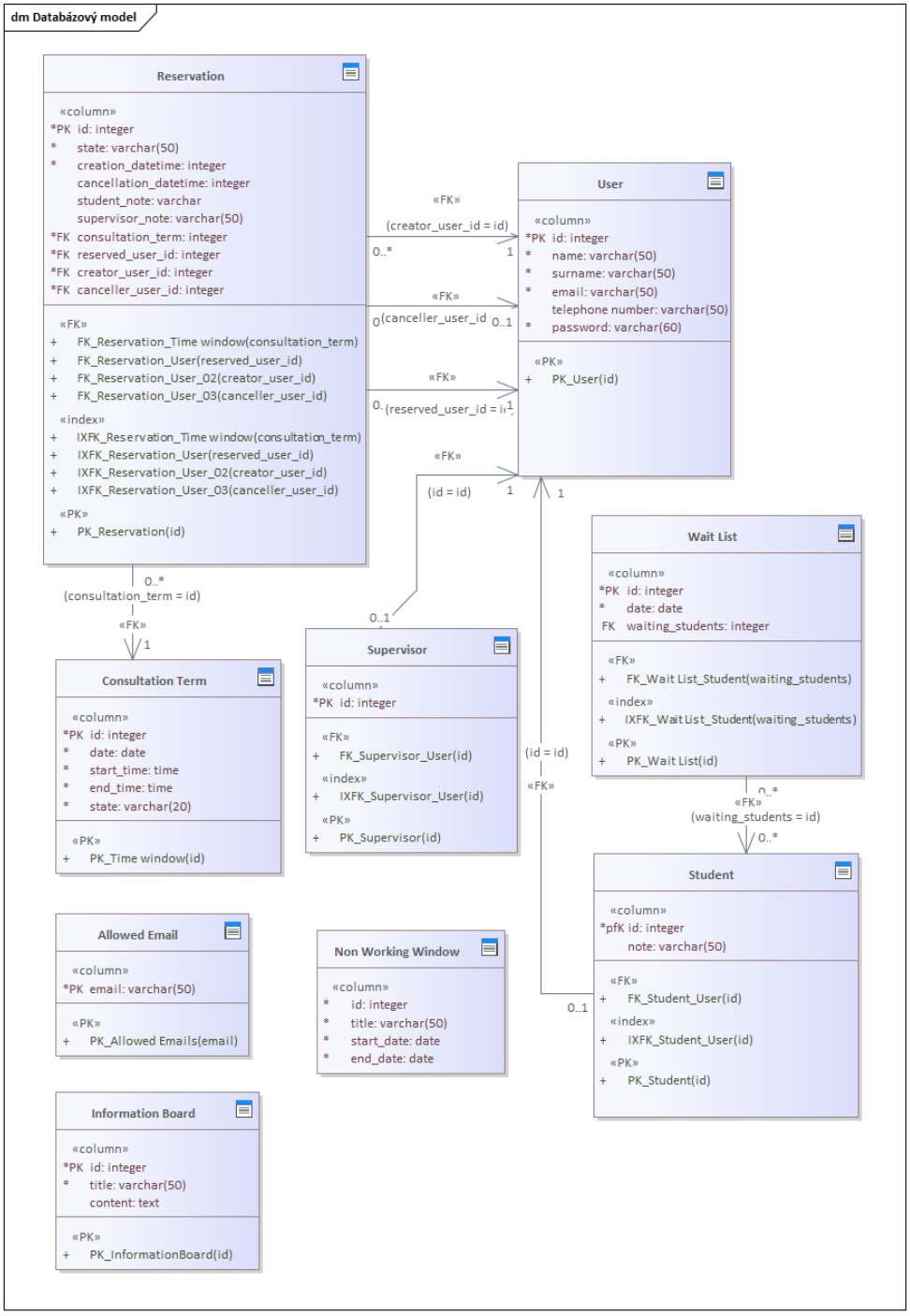
5.2.1.2 Databázový model

Obrázek 5.2 ukazuje model, na jehož základě jsme postavili databázové schéma v SQLAlchemy. Databáze byla vytvořena podle funkčních požadavků a jejich příslušných případů užití. Dále si stručně projdeme všechny tabulky zastoupené v tomto modelu a také se blíže podíváme na některé jejich atributy a hodnoty, které tyto atributy mohou nabývat.

- User – definuje osobní údaje uživatele: jméno, příjmení, e-mail, telefonní číslo a heslo k účtu.
- Supervisor - definuje uživatele s právy učitele.
- Student - definuje uživatele s právy studenta.
  - Atribut note specifikuje informace o pokroku studenta, které učitel zadává do databáze.
- Consultation Term - definuje konzultační termín.
  - Na jeden konzultační termín může být více rezervací, aktivní je však pouze jedna.

  - Termín konzultace může být v následujících stavech:
    * Volný – žádná aktivní rezervace na příslušný konzultační termín.
    * Obsazený – existuje aktivní rezervace na příslušný konzultační termín.
    * Blokovaný - učitel zablokoval tento konzultační termín a nyní není k dispozici pro přihlášení.

- Reservation - definuje rezervaci na konzultační termín.
  - Rezervace může být v následujících stavech:
    * Aktivní - student je přihlášen na konzultační termín odpovídající této rezervaci.
    * Zrušená - student nebo učitel zrušil rezervaci na příslušný termín konzultace.
    * Blokovaná - učitel rezervaci zablokoval, což znamená, že vyučující nebude moci se studentem v určeném termínu konzultovat.
- Non Working Window - definuje dny pracovního klidu.
- Wait list - definuje čekací listinu na konkrétní datum.
- Allowed Email - definuje seznam e-mailových adres, jejichž uživatelé se mohou do systému registrovat.
  - Když učitel přidá studenty do systému, zadá do formuláře e-mailové adresy, na které studenti obdrží pozvánky s odkazem na registrační stránku. Faktem je, že systém do této tabulky ukládá e-mailové adresy uvedené učitelem ve formuláři. Když se uživatel zaregistruje, systém zkontroluje, zda učitel pozval uživatele s e-mailem uvedeným v registračním formuláři, a pokud tam není, uživatel nebude moci pokračovat v registraci do systému.
- Information Board - definuje informační bloky, které učitel sdílí se studenty.
  - Používá se pouze na hlavní stránce a definuje základní pokyny pro používání kalendáře konzultací.
  - Snad v budoucnu vznikne sekce, kde bude učitel sdělovat potřebné informace studentům.



Obrázek 5.2: Databázový model

5.2.2 Flask-WTF

Část dokumentu, která umožňuje uživateli zadávat informace, které systém dále zpracovává, se nazývá formulář. Zpracování formuláře zabere spoustu času a pro zjednodušení tohoto procesu bylo zahrnuto rozšíření Flask-WTF[25]. Flask-WTF je rozšíření Flask, které integruje WTForms do Flasku. WTForms je knihovna pro ověřování a vykreslování formulářů pro vývoj webu v Pythonu. Formuláře jsou definovány jako třídy Pythonu, které rozšiřují třídu FlaskForm.

V rámci třídy jsou definice polí ve formě proměnných třídy a provádějí se vytvořením objektu, který je přidružen k typu pole.

Na obrázku 5.2 je formulář, s jehož pomocí uživatel odešle žádost o resetování hesla. **EmailField** a **SubmitField** jsou dvě třídy, které jsou typy polí, které jsme importovali z balíčku wtform.

- **EmailField** – definuje pole pro e-mailovou adresu.
- **SubmitField** - definuje tlačítko pro odeslání, které odešle všechny hodnoty formuláře do obsluhy formuláře.

Při vytváření výše uvedených polí předáváme argumenty konstruktoru.
validators – je seznam validátorů. Například **DataRequired()** je vestavěný validátor a kontroluje, zda byl poskytnut vstupní formulář. **validate_email()** – je vlastní validátor, který zkontroluje, zda e-mailová adresa existuje, a pokud se neshoduje, zobrazí chybu.

```
1 class RequestResetForm(FlaskForm):
2     email = EmailField('E-mail', validators=[DataRequired()])
3     submit = SubmitField('Obnovit heslo')
4 
5     def validate_email(self, email):
6         user = get_user_by_email(email.data)
7         if not user:
8             raise ValidationError('Špatný email')
```

Listing 5.2: Formulář pro resetování hesla.

Všechny formuláře jsou definovány v modulech forms.py.

5.2.3 Flask-Login

I když Flask dovolí vytvářet vlastní autentizační systém pomocí hashů a cookies, abych si nekomplikovala svůj úkol, obrátila jsem se na rozšíření Flask-Login.

Flask-Login je rozšíření, které se specializuje na správu aspektu systému ověřování uživatelů, aniž by bylo vázáno na konkrétní mechanismus ověřování.

Funkce tohoto rozšíření nám umožní provádět úkoly, jako je přihlášení, odhlášení a také zapamatování uživatelských relací.

Také v models.py jsme definovali zpětné volání funkce **user_loader** (viz 5.3). Hlavním účelem této funkce je znovu načíst uživatele z ID uživatele uloženého v relaci. Funkce **load_user()** bere jako parametr ID uživatele uložené v relaci a vrací odpovídající objekt uživatele. Třída, která reprezentuje uživatele, je reprezentována v models.py a nazývá se **User**. Abychom vrátili objekt třídy **User** odpovídající identifikátoru, odeslali jsme dotaz do naší databáze.

```
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
```

Listing 5.3: Funkce load_user

Abychom ochránili některá zobrazení před neoprávněnými uživateli, použili jsme dekorátor **@login_required** jako dekorátor nejvnitřnější funkce zobrazení. Jak ukazuje příklad 5.4.

```
@users.route("/reservation_calendar",methods=["GET","POST"])
@login_required
def reservation_calendar():
    ....
```

Listing 5.4: Funkce zobrazení reservation_calendar

5.2.4 Flask-Mail

V některých situacích je nutné, aby systém zasílal e-mailová upozornění.

Zatímco ve standardní knihovně pythonu existuje balíček smtplib, který lze použít k odesílání e-mailů, my použijeme rozšíření Flask-Mail, které nám poskytuje přístupné rozhraní pro nastavení SMTP s naší aplikací a zjednodušuje celkový proces odesílání zpráv.

Při výběru poštovní služby jsem vycházela z následujících faktorů:

- bezplatný,
- podporuje protokol SMTP,
- povolení nezabezpečených aplikací.

Ve výsledku moje volba padla na poštovní službu Gmail[26]. Gmail je bezplatná e-mailová služba od společnosti Google. Podporuje také použití poštovních klientů využívajících protokoly SMTP, POP a IMAP.[11]

Gmail má určitá omezení pro odesílání e-mailů. Jejich počet by neměl přesáhnout 500. (viz [12]) V našem případě je překročení této hranice nepravděpodobné. Pokud však útočník pošle žádost o resetování hesla 500krát, může tohoto limitu dosáhnout. Jelikož při resetování hesla uživatel obdrží email s odkazem na formulář pro nastavení nového hesla.

5.2.5 ItsDangerous

ItsDangerous[27] je knihovna pro bezpečné podepisování libovolných dat. Rozhraní podepisuje pouze bajty. Pro podepisování jiných typů importujeme modul TimedJSONWebSignatureSerializer, který poskytuje rozhraní pro dumps/loads podobné modulu Python json, který serializuje objekt do řetězce a poté jej podepisuje.

Pomocí tohoto modulu systém vygeneruje bezpečný, časově citlivý token, který může použít pouze uživatel, který má přístup k e-mailu.

Modul TimedJSONWebSignatureSerializer použijeme ve dvou případech:

- při resetování hesla z uživatelského účtu,
- po potvrzení registrace.

V obou případech uživatel obdrží e-mailem odkaz, ve kterém je token předán jako dynamická proměnná url dotazu.

Příklad odkazu 5.3, který byl uživateli zaslán e-mailem pro potvrzení e-mailu a dokončení registrace.

Registraci dokončíte kliknutím na odkaz:

https://supervision-support-system.herokuapp.com/registration/eyJhbGciOiJIUzUxMiIsImlhdCI6MTY1NTE2NzM2NiwiZXhwIjoxNjU1MTY4MjY2fQ.eyJlbWFpbCI6Imd1bGl5ZmkxQGN2dXQuY3oiLCJuYW1lIjoiRmlkYW4iLCJzdXJuYW1lIjoiR3VsaXlldmEiLCJ0ZWxlcGhvbmVfbnVtYmVyIjoiIiwicGFzc3dvcmQiOiIkMmIkMTIkbHlQa2JzUWpnNGpCZ2J1cWpsTGZwdVRjcGYuSGZCOTExTnRTTVNGQTE0OTBUS0ZZNkRCdk8ifQ.XU3adjPlG5VMaX96Zi7xeYPTa2Eph9FPRbDh2Y_eluEOonHMV2MZ3so_CAVt7-3pliQ-o7OEtqXYNAF6ziBwBQ

Obrázek 5.3: Obsah e-mailu s odkazem na dokončení registrace.

5.2.6 Flask-Bcrypt

Pro přihlášení k účtu musí uživatel zadat své uživatelské jméno a heslo. Dále server porovná data zadaná uživatelem s daty uloženými v databázi. Téměř všechna data uložená v databázi jsou veřejná. V případě hesla čelíme naléhavé potřebě uložit je v zašifrované podobě, protože pokud útočník najde mezeru a získá přístup k datům z databáze, nebude pro něj obtížné vstoupit do uživatelského účtu. K vyřešení tohoto problému používáme hash. Nyní bude muset uživatel uhodnout hesla. Jeho úspěch bude také záviset na rychlosti výpočtu hashovací funkce. Podle článku [13] je pro nás nejbezpečnější možností hašovací funkce bcrypt. Pro naše účely použijeme rozšíření poskytované flaskem Flask-Bcrypt.

Použili jsme dvě metody třídy Bcrypt:

- **create_password_hash** – aplikuje hashovací funkci na heslo a vrátí hashovanou hodnotu.

- **check_password_hash** - zkontroluje, zda zadané heslo odpovídá hashované hodnotě uložené v databázi.

5.3 Struktura

Flask nemá žádná omezení, pokud jde o strukturování aplikací. Vývojář si může zvolit strukturu projektu sám. Použila jsem nejoblíbenější a nejflexibilnější strukturu flask aplikace, která je nám nabízena v oficiální dokumentaci flask frameworku [8].
Strukturu projektu můžeme vidět na obrázku 5.4.

- `flask_app` ..... balíček, který obsahuje kód a soubory pro naši aplikaci.
- `requirements.txt` soubor, který ukládá informace o všech závislostech a jejich verzích.
- `run.py` ..... soubor, ve kterém se aplikace spouští na serveru.
- `static` .. složka, která obsahuje statické soubory aplikace, které používají šablony.
- `venv` ..... virtuální prostředí aplikace, kde je nainstalován Flask a další závislosti.

Obrázek 5.4: Struktura projektu

5.3.1 flask_app/

flask_app je balíček, který obsahuje kód a soubory pro naši aplikaci.
Strukturu flask_app můžeme vidět na obrázku 5.6.

5.3.1.1 Blueprint moduly

Flask blueprinty slouží ke vhodnému seskupování jednotlivých pohledů do samostatných celků ve velkých webových aplikacích.[9] Na obrázku 5.5 můžeme detailněji vidět strukturu samostatného blueprint modulu.
auth, **errors**, **reservation**, **terms**, **users** jsou blueprint moduly naší webové aplikace, které jsou zodpovědné za konkrétní procesy v našem systému (viz 5.5).

- **auth** blueprint modul, který je zodpovědný za autorizaci v našem systému (registrace, přihlášení, resetování hesla);
- **errors** blueprint modul, který je zodpovědný za zpracování síťových chyb.
  - Flask poskytuje aplikační mechanismus pro vytváření vlastních chybových stránek. K deklaraci vlastní obslužné rutiny chyb se používá dekorátor **@errorhandler**. Pro odpovídající chybu server vrátí obsah šablony odpovídající chybě (viz 5.5).

```
1       @errors.app_errorhandler(404)
2       def error_404(error):
3         return render_template('errors/404.html'), 404
4
```

Listing 5.5: Příklad funkce vracející chybovou stránku 404.

- **reservation** blueprint modul, který zodpovídá za veškeré manipulace související s rezervacemi.
  - vytvoření, zrušení, přenos a blokování
- **terms** blueprint modul, který zodpovídá za veškeré manipulace související s konzultačními termíny.
  - vytváření a odstranění
- **users** blueprint modul, který je zodpovědný za manipulaci s osobními údaji všech uživatelů.

`forms.py` ..... modul pro definování formuláře
`utils.py` ..... modul definující obslužné funkce
`routes.py` ..... modul definující cesty
`__init__.py` je prázdný modul, jehož jediným účelem je říci Pythonu, že blueprint modul je balíček.

Obrázek 5.5: Struktura blueprint modulu

5.3.1.2 templates/

templates je složka obsahující šablony aplikací. Šablony byly rozděleny do následujících tří složek:

- General
  - zde uloženy šablony, které se zobrazují neoprávněným uživatelům (stránka registrace, přihlášení, resetování hesla).
- Supervisor
  - zde jsou uloženy šablony, které se zobrazují pouze učitelům
- Student
  - zde jsou uloženy šablony, které se zobrazují pouze studentům
- Errors

– zde jsou uloženy šablony, které se uživatelům zobrazí, když dojde k chybě sítě.

- `auth`
- `users`
- `terms`
- `reservations`
- `errors`
- `templates`
- `static` .. složka, která obsahuje statické soubory aplikace, které používají šablony.
- `__init__.py`
- `models.py` ..... modul obsahuje definici modelů.
- `config.py` ..... modul obsahuje nastavení konfigurace aplikace.
- `__init__.py` .... modul, ve kterém je vytvořena instance aplikace flask se všemi potřebnými nastaveními.

Obrázek 5.6: Struktura flask_app balíčků.

5.4 Shrnutí implementace

Součástí implementace byl popis vývoje serverové a klientské části naší webové aplikace. Byla také popsána struktura zdrojového kódu. Výstupem této sekce je zdrojový kód, který je připraven k testování.

Kapitola 6

Testování

Tato sekce je věnována procesu testování našeho systému.

Testování je důležitým krokem ve vývoji softwaru. V této fázi se kontroluje soulad výsledků našeho softwaru se zadanými kritérii. S ohledem na objekt testu je poté můžeme rozdělit na dva hlavní typy testování: funkční a nefunkční.

Funkční testování je založeno na kontrole splnění funkčních požadavků stanovených pro náš software a také na kvalitě jejich implementace. Nefunkční testování se zase týká kontroly našeho souladu s nefunkčními požadavky našeho systému.

Dále si podrobněji popíšeme všechny typy testování, které byly na našem systému provedeny, a jejich výsledky. Je třeba poznamenat, že všechny testy uvedené v této části byly provedeny ručně, jinými slovy, nebyly použity žádné další softwarové nástroje.

6.1 Funkční testování

V průběhu funkčního testování byla identifikována a opravena řada chyb, z nichž většina souvisela s nesprávným zněním dotazů zasílaných do databáze. Také bylo během testování zjištěno, že všechny akce, které byly doprovázeny odesíláním e-mailů, byly pomalé. Jak se ukázalo, metoda, která byla v systému použita k odesílání e-mailových zpráv, zablokovala na několik sekund provedení funkce zobrazení. Tato chyba byla opravena pomocí vláken, nyní jsou e-maily odesílány v samostatném vláknu.

6.2 Nefunkční testování

6.2.1 Bezpečnostní testování

Provedla jsem bezpečnostní testování, jehož účelem bylo identifikovat zranitelnost našeho softwaru vůči určitým typům SQLI. Během testování bylo nalezeno velké množství zranitelností a následně byly přidány další funkce pro ověření dat přijatých od uživatele.

6.2.2 Uživatelské testování

Aby bylo možné vyhodnotit interakci uživatele se systémem zvenčí, bylo provedeno uživatelské testování. Díky této testovací metodě vývojář zlepšuje uživatelskou zkušenost tím, že analyzuje zpětnou vazbu od koncového uživatele.

Ve fázi implementace softwaru jsem nejprve implementovala uživatelské rozhraní. Paralelně mohli koncoví uživatelé hodnotit hotové modely a provádět úpravy. To bylo provedeno s cílem poskytnout jim představu o funkčnosti systému a teprve když prototyp vyhovoval všem, začala jsem funkcionalitu implementovat.

Poté, co jsem dokončila implementaci softwaru, tak jsem provedla uživatelské testování na základě testovacích scénářů (viz C). Tyto scénáře byly založeny na modelech případů použití. Účastníky testování byli 3 studenti vysoké školy. Můj výběr je odůvodněn tím, že všichni již měli zkušenosti se psaním závěrečných prací a mají od systému svá očekávání. Dva studenti, stejně jako já, zvolili jako vedoucího Pavla Náplavu a mají zkušenosti se současně používaným systémem Microsoft OneNote.

Na konci testování vyplnili uživatelé formulář zpětné vazby. Skládal se z následujících otázek:

- S jakými testovacími scénáři máte potíže a proč?
- Co byste chtěli přidat do systému?
- Byli jste spokojeni se svými zkušenostmi s tímto systémem?

Stanovili jsme si následující cíle:

1. Zjistit, jaké problémy má uživatel při práci se systémem
2. Zjistit, jak systém naplňuje očekávání uživatelů.

6.2.2.1 Shrnutí uživatelského testování

Během výsledků testu se ukázalo, že hlavní problémy s porozuměním způsobovaly scénáře C.A5 a C.A6. Toto jsou scénáře pro učitele. Se studentskými scénáři nebyly žádné problémy.

Proces vytváření a odstraňování opakujících se termínů nesplnil očekávání uživatelů. Například jeden z uživatelů při vytváření opakujícího se termínu zvolil frekvenci opakování 1x týdně, ale neurčoval dny v týdnu. V důsledku toho systém vůbec nevytvářel konzultační termíny, uživatel však očekával, že se konzultační termín bude opakovat každých 7 dní od začátku zadaného časového intervalu. Zdálo se mi ale, že učitel nepotřebuje, aby mu systém určil den v týdnu, kdy se termín opakuje, takže jsem s tím nic nedělala.

Během uživatelského testování byl změněn design formuláře pro vytvoření termínu. Dříve musel uživatel definovat frekvenci opakování, pokud chtěl, aby se termín opakoval ve vybrané dny v týdnu. Ve výchozím nastavení ne byla frekvence opakování termínu definována. Uživatel si tak mohl vybrat dny v týdnu a nemusel specifikovat frekvenci.V tomto případě systém ignoroval vybrané dny v týdnu a termín opakoval každý den v zadaném časovém intervalu. Nyní uživatel nemůže vybrat dny v týdnu bez určení frekvence.

6.3 Shrnutí testování

V této části byly popsány typy funkčních a nefunkčních testů prováděných na našem systému a jejich výsledky, v průběhu testování byla opravena řada chyb. Nyní je náš systém připraven k provozu.

Kapitola 7

Zhodnocení vytvořeného systému

7.1 Náklady

Tato kapitola popisuje náklady na vývoj a provoz naší webové aplikace.

7.1.1 Vývoj

V tabulce 7.1 byl spočítán čas strávený na jednotlivých fázích vývoje softwaru a s přihlédnutím k průměrné mzdě juniorského fullstack programátora (200 korun za hodinu) byly vypočteny přibližné náklady na vývoj této aplikace.

Tabulka 7.1: Náklady na vývoj

Činnost	člh	Cena bez DPH
Analýza	80	16 000
Návrh funkčnosti	120	24 000
Implementace	240	48 000
Testování	80	16 000
Nasazení	8	1 600
Součet	520	105 600

7.1.2 Provoz

Jelikož je náš systém určen pro učitele, kteří ve většině nemají serverovnu, můžeme zvážit dvě možnosti provozu aplikace:

- nasadit aplikaci na localhost
  - tato možnost nám nevyhovuje, protože aplikace musí fungovat 24/7.

- použít webhosting
  - weboví hostitelé poskytují zdroje pro hostování informací na serveru.[14]

Cena nasazení bude přímo záviset na ceně zvoleného webhostingu. K nasazení aplikace jsem použila webhosting Heroku[15]. Heroku je kontejnerová cloudová platforma, která poskytuje přizpůsobené prostředí pro hostování webových stránek. Heroku nám nabízí 4 různé plány, my jsme použili plán Free & Hobby. Také jedna ze služeb, kterou jsem využívala, poskytovaná cloud serverem a současně zahrnuta v námi zvoleném tarifu, je databáze jako služba založená na PostgreSQL. Bezplatný tarif nám poskytuje maximálně 10 000 řádků v databázi a kapacitu úložiště 1 GB.

Jedním z hlavních nástrojů pro nasazení aplikace na Heroku je GIT. Náš projekt má soubor requirements.txt, díky kterému se po nasazení aplikace nainstalují příslušné závislosti.

Mezi výhody tohoto cloudového serveru patří:

- snadné použití a nasazení,
- dobrá dokumentace,
- poskytuje doplňkové služby (Add-ons) (databáze, poštovní služba atd.).

Můžeme také nasadit naše aplikace na následujících hostingech, které poskytují bezplatné sazby:

- Vercel[33]
- Qovery[34]

7.2 Přínosy použití systému

Tento systém nepovažujeme za náhradu současně používaného systému Microsoft OneNote. Naším cílem bylo zjednodušit procesy související s plánováním konzultačních termínů. Abychom mohli vyhodnotit efektivitu naší webové aplikace, porovnáme posloupnost akcí, kterou je nutné dodržet k vyplnění konkrétního procesu spojeného s konzultačním termínem.

- Vytvoření konzultačního termínu
  - Microsoft OneNote - přidání buňky do tabulky.
  - Náš systém - vyplnění formuláře pro vytvoření termínu.
- Odstranění konzultačního termínu
  - Microsoft OneNote - vyplnění formuláře pro odstranění termínu.
  - Náš systém - odstranění odpovídajících buněk tabulky.

- Blokování termínu
  - Microsoft OneNote - vybarvením sloupce červeně a odesláním hromadného upozornění na zrušení termínu.
  - Náš systém - vyplnění formuláře pro blokování termínu.
- Přihlášení studenta na konzultační termín
  - Microsoft OneNote - zadání jména studenta do tabulky.
  - Náš systém - stisknutím tlačítka „Přihlásit“.
- Odhlášení studenta z konzultačního termínu
  - Microsoft OneNote - odstranění jména studenta z tabulky.
  - Náš systém - stisknutím tlačítka „Odhlásit“.
- Přidávání poznámky k rezervacím studentů
  - Microsoft OneNote - vytváření samostatného poznámkového bloku.
  - Náš systém - vyplnění formuláře pro přidání nebo úpravu existující poznámky týkající se konkrétní rezervace studenta.
- Přidání poznámky o studentovi
  - Microsoft OneNote - vytvoření samostatného poznámkového bloku.
  - Náš systém - vyplnění formuláře pro přidání nebo úpravu existující poznámky týkající se konkrétního studenta.
- Zobrazení historie rezervací
  - Microsoft OneNote - prohledávání historie verzí dokumentu.
  - Náš systém - přechod do speciální sekce, která zobrazí seznam všech dokončených a zrušených rezervací konkrétního studenta nebo všech studentů dohromady.

Jak vidíme, provádění většiny procesů se stalo mnohem efektivnější, učitelům nyní stačí kliknout na tlačítko nebo vyplnit formulář. Náš systém také poskytuje studentům možnost zařadit se na čekací listinu. Pokud se pak v den, který si zvolí student, uvolní místo, obdrží upozornění e-mailem. Tato možnost není v systému Microsoft OneNote dostupná.

7.3 Shrnutí zhodnocení vytvořeného systému

V této části jsme spočítali náklady na vývoj a provozu našeho systému a také jeho výhody oproti stávajícímu systému Microsoft OneNote. Na základě analýzy nákladů a efektivity našeho systému můžeme dospět k následujícím závěrům:

- Systém je vhodný pro učitele, kteří se svými studenty pravidelně konzultují.
- Systém není vhodný pro velký počet studentů, protože bezplatný hostingový plán, který jsme použili, nás omezuje v počtu požadavků a v množství alokované paměti v databázi.
- Systém je vhodný pro použití jedním vyučujícím, pokud tento systém používá více vyučujících, je potřeba jej spustit na různých adresách.

Závěr

V rámci své bakalářské práce jsem vytvořila systém, který slouží k podpoře vedení závěrečných prací. Před zahájením implementace bylo nutné analyzovat požadavky na systém. Po tomto kroku bylo třeba podobné systémy prostudovat a v průběhu studia zjistit, zda splňují všechny naše požadavky, a inspirovat se těmito systémy během naší vlastní implementace. Dále bylo nutné vytvořit návrh systému a tento návrh implementovat pomocí vhodných nástrojů a prostředí systému. Dále bylo zapotřebí provést uživatelské testování, vyhodnotit přínosy systému a porovnat je s náklady na vývoj a provoz. Na závěr došlo k rozhodnutí, zda je vhodné tento systém provozovat a za jakých podmínek.

Na začátku své práce jsem studovala aktuálně používaný systém pro podporu závěrečných prací Microsoft OneNote. Během studie byla nalezena problémová doména tohoto systému – neefektivita splnění procesů spojených s plánováním konzultačních terminů. Bylo navrženo řešení tohoto problému, a to převedením tohoto procesu do jiného systému, který bude mít potřebnou funkcionalitu pro plánování konzultačních termínů. Uspořádala jsem také sérii schůzek s vedoucím a studenty, jejichž hlavním účelem bylo zjistit očekávání ohledně funkčnosti nového systému. Identifikovala jsem uživatelské požadavky stanovené na náš systém. Poté jsem začala na internetu hledat software, který by je mohl částečně nebo úplně pokrýt. Téměř všechny systémy, které se mi podařilo najít, byly komerční, ale vzhledem k tomu, že poskytovaly demoverzi, mohla jsem si jejich funkčnost podrobně prostudovat. Tyto systémy zahrnují systémy jako Reservio, Reenio a Doodle. Při analýze se ukázalo, že žádný z těchto systémů zcela nepokrývá naše uživatelské požadavky. V důsledku toho bylo rozhodnuto vytvořit nový systém. Dále jsem spočítala celkové náklady na vývoj a nasazení našeho systému, také jsem porovnala náš systém se současným používaným systémem Microsoft OneNote, vyzdvihla výhody a určila případy, kdy je provoz našeho systému vhodný.

Výstupem práce je bezplatný systém určený pro zjednodušení plánování konzultací a evidenci pokroku studentů během procesu vedení závěrečných prací. Do budoucna je možné funkčnost tohoto systému rozšířit – přidat sekci

s nástěnkou, zejména pro tyto účely máme v databázi odpovídající tabulku.

Všechny cíle stanovené v kapitole 1 byly splněny a systém je dostupný na adrese: `https://supervisionsupport-system.herokuapp.com`

Literatura

[1] WIEGERS, Karl Eugene a Joy BEATTY. Software requirements. 3rd ed. Redmond, WA: Microsoft press, c2013. Best practices. ISBN 0735679665.

[2] Microsoft OneNote [online]. [cit. 2022-06-15]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/onenote/digital-note-taking-app

[3] VARLAMOV, Alexej. CHto takoe SaaS servis prostymi slovami [online]. 20. listopadu 2019 [cit. 2022-06-12]. Dostupné z: https://wezom.com.ua/blog/saas-prilozheniya

[4] Termíny a jejich nastavení. Reeservio [online]. [cit. 2022-06-12]. Dostupné z: https://manual.reenio.cz/cs/napoveda/terminy-a-jejich-nastaveni

[5] Pyramid Introduction. The Pyramid Web Framework [online]. Agendaless Consulting, 2022 [cit. 2022-06-13]. Dostupné z: https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/introduction.html

[6] Přehled rámce Bootstrap s užitečnými hacky pro vývojáře. In: Liquidhub [online]. [cit. 2022-06-13]. Dostupné z: https://liquidhub.ru/blogs/blog/obzor-freymvorka-bootstrap

[7] Mezhsajtovaya poddelka zaprosa: zashchita ot CSRF atak. Tproger [online]. [cit. 2022-06-13]. Dostupné z: https://tproger.ru/articles/mezhsajtovaja-poddelka-zaprosa-zashhita-ot-csrf-atak/

[8] Project Layout. Flask [online]. [cit. 2022-06-13]. Dostupné z: https://flask.palletsprojects.com/en/2.1.x/tutorial/layout/

[9] Webové aplikace: Flask. Nauč se Python! [online]. [cit. 2022-06-13]. Dostupné z: https://naucse.python.cz/lessons/intro/flask/

[10] SB Admin. Start Bootstrap Themes Templates Bundles [online]. [cit. 2022-06-13]. Dostupné z: https://startbootstrap.com/template/sb-admin

[11] Gmail. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-06-14]. Dostupné z:https://ru.wikipedia.org/wiki/Gmail

[12] Limits for sending & getting mail. Gmail Help [online]. [cit. 2022-06-14]. Dostupné z: https://support.google.com/mail/answer/22839?hl=en#zippy=%2Cyou-have-reached-a-limit-for-sending-mail

[13] Kak nado heshirovat' paroli i kak ne nado. Habr [online]. [cit. 2022-06-14]. Dostupné z: https://habr.com/ru/post/210760/

[14] Hosting. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-06-14]. Dostupné z: https://ru.wikipedia.org/wiki/Hosting

[15] Heroku [online]. [cit. 2022-06-14]. Dostupné z: https://dashboard.heroku.com/apps

[16] Reservio [online]. [cit. 2022-06-15]. Dostupné z: https://www.reservio.cz/CZ/Praha

[17] Reenio [online]. [cit. 2022-06-15]. Dostupné z: https://reenio.cz

[18] Balíčky a ceny. Reenio [online]. [cit. 2022-06-15]. Dostupné z: https://reenio.cz/cs/balicky#PricingPlan

[19] Doodle (webová stránka). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-06-15]. Dostupné z: https://en.wikipedia.org/wiki/Doodle_(website)

[20] In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-06-15]. Dostupné z: https://ru.wikipedia.org/wiki/

[21] Bootstrap 5 [online]. [cit. 2022-06-15]. Dostupné z: https://getbootstrap.com/docs/5.0/getting-started/introduction/

[22] Django [online]. [cit. 2022-06-15]. Dostupné z: https://www.djangoproject.com

[23] Jinja [online]. [cit. 2022-06-15]. Dostupné z: https://jinja.palletsprojects.com/en/3.1.x/

[24] Flask-SQLAlchemy [online]. [cit. 2022-06-15]. Dostupné z: https://flask-sqlalchemy.palletsprojects.com/en/2.x/

[25] Flask-WTF [online]. [cit. 2022-06-15]. Dostupné z: https://flask-wtf.readthedocs.io/en/1.0.x/

[26] Gmail [online]. [cit. 2022-06-15]. Dostupné z: https://www.google.com/intl/cs/gmail/about/

[27] ItsDangerous [online]. [cit. 2022-06-15]. Dostupné z: https://itsdangerous.palletsprojects.com/en/2.1.x/

[28] Webové aplikace: typy, architektura a principy fungování. HIGHLOAD [online]. [cit. 2022-06-19]. Dostupné z: https://highload.today/veb-prilozheniya/

[29] Pljusy i minusy Bootstrap. Cheapsender [online]. 19. září 2018 [cit. 2022-06-19]. Dostupné z: https://timeweb.com/ru/community/articles/plyusy-i-minusy-bootstrap-1

[30] Python. Urok 13. Moduli i paketi. Devpractise [online]. [cit. 2022-06-20]. Dostupné z: https://devpractice.ru/python-lesson-13-modules-and-packages/

[31] Podborka: 6 poleznyh instrumentov dlya sozdaniya sajtov na Bootstrap. Vc.ru [online]. [cit. 2022-06-23]. Dostupné z: https://vc.ru/services/161862-podborka-6-poleznyh-instrumentov-dlya-sozdaniya-saytov-na-bootstrap

[32] #14 Sozdanie baz dannyh vo Flask. PythonRu [online]. [cit. 2022-06-23]. Dostupné z: https://pythonru.com/uroki/14-sozdanie-baz-dannyh-vo-flask

[33] Vercel [online]. [cit. 2022-06-23]. Dostupné z: https://vercel.com

[34] Qovery [online]. [cit. 2022-06-23]. Dostupné z: https://www.qovery.com

Příloha A

Seznam použitých zkratek

CSRF Cross-site request forgery

CSS Cascading Style Sheets

DRY Don't repeat yourself

HTML Hypertext Markup Language

IMAP Internet Message Access Protocol

JS JavaScript

MVC Model-view-controller

ORM Objektově relační mapování

POP Post Office Protocol

SaaS Software jako služba

SMTP Simple Mail Transfer Protocol

SQLI Structured Query Language injection

SQL Structured Query Language

URL Uniform Resource Locator

XSS Cross-site scripting

Příloha B

Slovník

Opakující se konzultační termín opakujícím se termínem konzultace se zde rozumí termín, který se bude v určitém časovém intervalu opakovat. Řekněme každý den, jednou týdně nebo jednou za měsíc atd.

Uživatelský požadavek -je úkol, který musí být v systému schopny provádět určité třídy uživatelů. [1]

Identifikace -je proces rozpoznávání subjektu podle jeho identifikátoru.

Autentizace -je proces ověřování uživatele.

Autorizace -je "udělení určité osobě nebo skupině osob oprávnění k provádění určitých úkonů; stejně jako proces kontroly (potvrzení) těchto práv při pokusu o provedení těchto akcí."[20]

Modul v Pythonu -je soubor s příponou .py. Moduly jsou navrženy tak, aby ukládaly často používané funkce, třídy, konstanty a tak dále. Jsou určeny k importu do jiných programů. [30]

Balíčky v pythonu -jsou adresáře, které obsahují další adresáře a moduly, ale také obsahují soubor ___init___.py.[30]

Příloha C

Testovací scénáře

A.1 Registrace

Prerekvizity - uživatel by měl již dříve obdržet e-mail s odkazem na registrační stránku.

Testovací scénář:

1. V e-mailové schránce je potřeba najít dopis s odkazem na registraci a následovat ho.

2. Poté se otevře registrační stránka s formulářem, který je potřeba vyplnit, povinná pole jsou jméno, příjmení, emailová adresa, heslo a potvrzení hesla, volitelně můžete přidat telefonní číslo.

3. Dále bude na zadaný e-mail zaslán odkaz, kliknutím na něj bude registrace dokončena.

A.2 Resetování hesla

Prerekvizity - uživatel musí být registrován v systému a mít přístup k e-mailu.

Testovací scénář:

1. Neoprávnění uživatelé jsou automaticky přesměrováni na přihlašovací stránku.

2. Musíte sledovat odkaz „Zapomněli jste své heslo?“

3. Poté se otevře stránka s formulářem, do kterého budete muset zadat svůj e-mail.

4. Na vaši e-mailovou adresu bude zaslán e-mail s odkazem na obnovení hesla.
5. Během následujících 15 minut (když je odkaz aktivní) musíte odkaz sledovat.
6. Po kliknutí na odkaz se otevře nový formulář, kde je potřeba zadat nové heslo.

A.3 Přidání studenta

Prerekvizity - uživatel musí znát vstupní data do účtu učitele.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Seznam v sekci STUDENTI.
2. Klikněte na tlačítko „Přidat studenta“ a do otevřeného formuláře zadejte poštovní adresy studentů, kliknutím na tlačítko „Poznámka“ můžete k obsahu dopisu přidat i poznámku.
3. Na zadané e-mailové adresy bude zaslán e-mail s odkazem na registrační stránku.

A.4 Odstranění studenta

Prerekvizity - uživatel musí znát vstupní data do účtu učitele.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Seznam v sekci STUDENTI.
2. V seznamu studentů klikněte na tlačítko „Odstranit“ u konkrétního studenta.
3. Veškeré informace o studentovi a jeho rezervacích budou ze systému vymazány.

A.5 Vytvoření termínu konzultace

Prerekvizity - uživatel musí znát vstupní data k účtu učitele.

Testovací scénář:

1. Stránka pro vytvoření konzultačního termínu se nachází v sekci TERMÍN v záložce Vytvořit

2. Pro vytvoření termínu musíte vyplnit formulář s následujícími údaji:

   - **Časový interval**
   - **Hodinový interval**
   - **Opakování**

     Tato volba otevře další prvky formuláře, kde je potřeba dodatečně vybrat dny v týdnu a frekvenci opakování termínu konzultace.
   - **Student**

     Tato volba umožňuje zapsat studenta na vytvořený konzultační termín. K tomu je třeba vybrat jednoho studenta z rozevíracího seznamu.
   - **Délka konzultačního termínu**

     Toto pole definuje dobu trvání jednoho konzultačního termínu. V daném hodinovém intervalu bude vytvořen maximální počet termínů dané doby trvání.
   - **Počet konzultací**

     Toto pole specifikuje počet konzultačních termínů. V daném hodinovém intervalu se vytvoří daný počet termínů, přičemž délka každého z nich bude maximální možná.
   - **Dny pracovního klidu**
     - Tato volba vytvoří konzultační termín v nepracovní den
     - nebo naopak, nevytvářet konzultační termín, který se protíná s nepracovním dnem.

A.6 Vymazání konzultačního termínu

Prerekvizity - uživatel musí znát vstupní data do účtu učitele.

Testovací scénář

1. Stránka pro vytvoření konzultačního termínu se nachází v sekci TERMÍN v záložce Smazat

2. Chcete-li termín smazat, vyplňte formulář s následujícími údaji:

   - **Časový interval**
   - **Hodinový interval**
   - **Opakování**

     Tato volba otevře další prvky formuláře, kde je potřeba dodatečně vybrat dny v týdnu a frekvenci opakování.

3. Systém odstraní všechny termíny, které se překrývají s vybranými časovými intervaly.

A.7 Blokování konzultačního termínu

Prerekvizity - uživatel musí znát vstupní data do účtu učitele.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si v kalendáři příslušný den.
2. V rozvrhu, který se otevře, musíte kliknout na tlačítko „Zrušit“.
3. Po kliknutí se otevře modální okno, ve kterém můžete vybrat jeden nebo více výrazů, které chcete zablokovat. Kliknutím na tlačítko „Poznámka“ můžete také určit obsah e-mailu, který student obdrží poštou, pokud měl rezervaci na zablokovaný termín.
4. Po odeslání formuláře bude termín zablokován a nebude možné se do něj zapsat, a pokud do něj byl student zapsán, tak bude jeho rezervace zrušena a bude mu zasláno upozornění.

A.8 Vytvoření rezervace

Prerekvizity - uživatel musí znát vstupní data do účtu učitele.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si v kalendáři příslušný den.
2. V rozvrhu, který se otevře, je potřeba kliknout na tlačítko „Přihlásit“ u požadovaného termínu.
3. Otevře se vám modální okno s rozevíracím seznamem studentů. Je třeba vybrat jednoho z nich.

Prerekvizity - Uživatel musí znát vstupní data do studentského účtu.

Testovací scénář

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si v kalendáři příslušný den.
2. V rozvrhu, který se otevře, je potřeba kliknout na tlačítko „Přihlásit“ u termínu zájmu.

A.9 Zrušení rezervace

Prerekvizity -uživatel musí znát vstupní data k účtu učitele. Vybraný den nesmí být zastaralý. Na vybraný den musí být rezervace.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si v kalendáři příslušný den.

2. V rozvrhu, který se otevře, musíte kliknout na tlačítko „Odhlásit“ vedle termínu.

3. Rezervace bude zrušena a termín bude znovu dostupný pro přihlášení.

Prerekvizity - Uživatel musí znát vstupní data do studentského účtu. Vybraný den nesmí být starý ani dnešní. Ve vybraný den musí mít uživatel aktivní rezervaci.

Testovací scénář

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si v kalendáři příslušný den.

2. V rozvrhu, který se otevře, musíte kliknout na tlačítko „Odhlásit“ vedle termínu.

3. Rezervace bude zrušena a termín bude znovu dostupný pro přihlášení.

A.10 Přesun rezervace

Prerekvizity - uživatel musí znát vstupní data do studentského účtu. Vybraný den musí být dnešní.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat aktuální den v kalendáři.

2. V otevřeném rozvrhu musíte kliknout na tlačítko „Posun“ vedle vaší rezervace, kterou musíte přesunout.

3. Rezervace bude přesunuta.

A.11 Přidání studenta do pořadníku

Prerekvizity - uživatel musí znát vstupní data do studentského účtu. Vybraný den může být pouze v budoucnosti.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si den v kalendáři.
2. V otevřeném rozvrhu je potřeba kliknout na tlačítko „Pořadník“.
3. Jakmile bude pro vybraný den volné místo, obdržíte upozornění e-mailem.

A.12 Prohlížení čekací listiny

Prerekvizity - uživatel musí znát vstupní data k účtu učitele.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Kalendář v sekci REZERVACE a vybrat si den v kalendáři.
2. V otevřeném rozvrhu je potřeba kliknout na tlačítko „Pořadníky“.
3. Modální okno, které se otevře, zobrazí seznam všech studentů, kteří byli přidáni do pořadníku.

A.13 Zobrazení historie rezervací studenta

Prerekvizity - uživatel musí znát vstupní data k účtu učitele.

Testovací scénář:

1. Nejprve je potřeba přejít na záložku Seznam v sekci STUDENTI
2. V seznamu studentů přejděte na odkaz se jménem a příjmením studenta, o kterého máte zájem.
3. Otevře se stránka obsahující jeho osobní údaje a seznam všech jeho rezervací.
4. Kliknutím na rezervaci o ní získáte podrobné informace.

Prerekvizity - uživatel musí znát vstupní data do studentského účtu.

Testovací scénář:

1. Je potřeba přejít na záložku Seznam v sekci Rezervace.
2. Otevře se stránka obsahující vaše osobní údaje a seznam všech vašich rezervací.
3. Kliknutím na rezervaci o ní získáte podrobné informace.

A.14 Zobrazení historie všech rezervací

Prerekvizity - uživatel musí znát vstupní data k účtu učitele.

Testovací scénář

1. Musíte přejít na záložku Historie v sekci REZERVACE.
2. Otevře se stránka se seznamem studentských rezervací.

A.15 Úprava osobních údajů v účtu

Prerekvizity - uživatel musí znát vstupní data k účtu učitele.

Testovací scénář:

1. Musíte kliknout na ikonu osoby v pravém horním rohu stránky a vybrat sekci „Osobní účet“.
2. Stisknutím tlačítka „Upravit“ můžete změnit jméno, příjmení, poštovní adresu a telefonní číslo.
3. Stisknutím tlačítka „Změnit heslo“ můžete změnit stávající heslo, k tomu je potřeba zadat aktuální heslo a heslo nové.

Prerekvizity - uživatel musí znát vstupní data do studentského účtu.

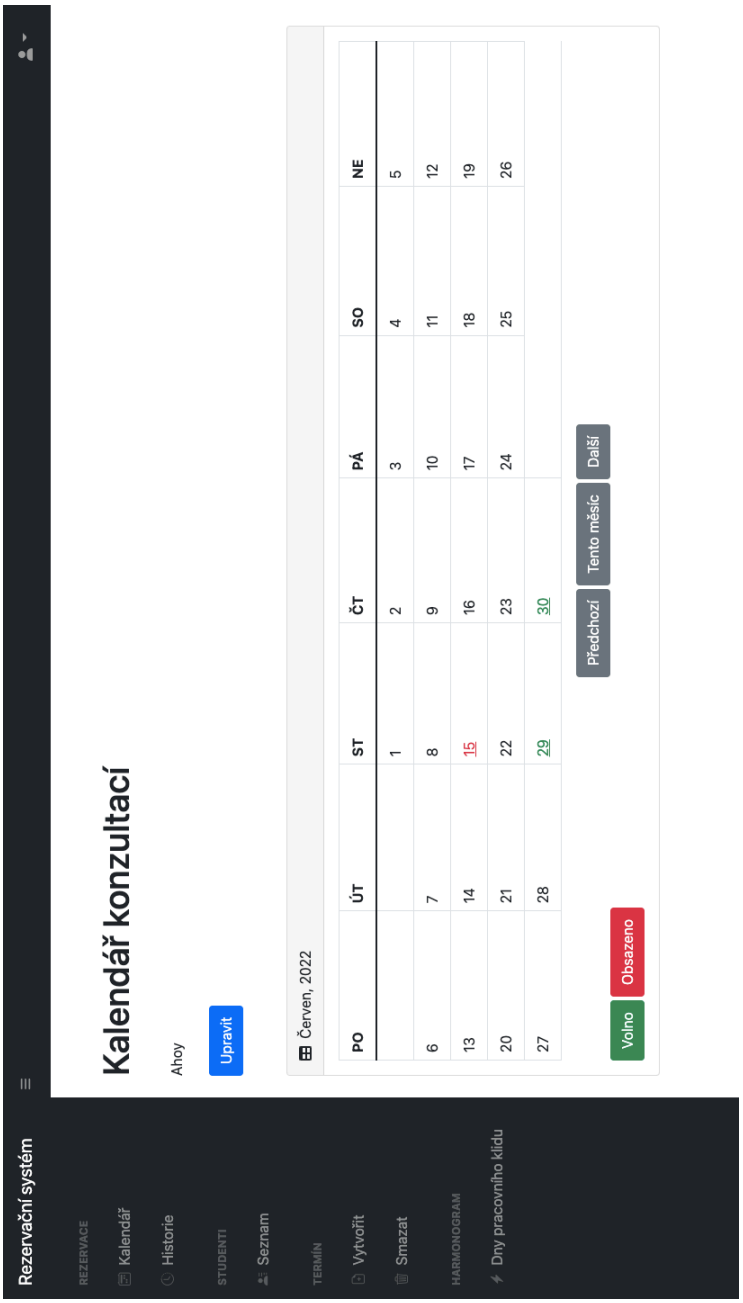
Testovací scénář:

1. Musíte kliknout na ikonu osoby v pravém horním rohu stránky, vybrat sekci „Osobní účet“ nebo přejít na záložku Nastavení v sekci Osobní účet.
2. Stisknutím tlačítka „Upravit“ můžete změnit jméno, příjmení a telefonní číslo.
3. Stisknutím tlačítka „Změnit heslo“ můžete změnit stávající heslo. K tomu je potřeba zadat aktuální heslo a nové.

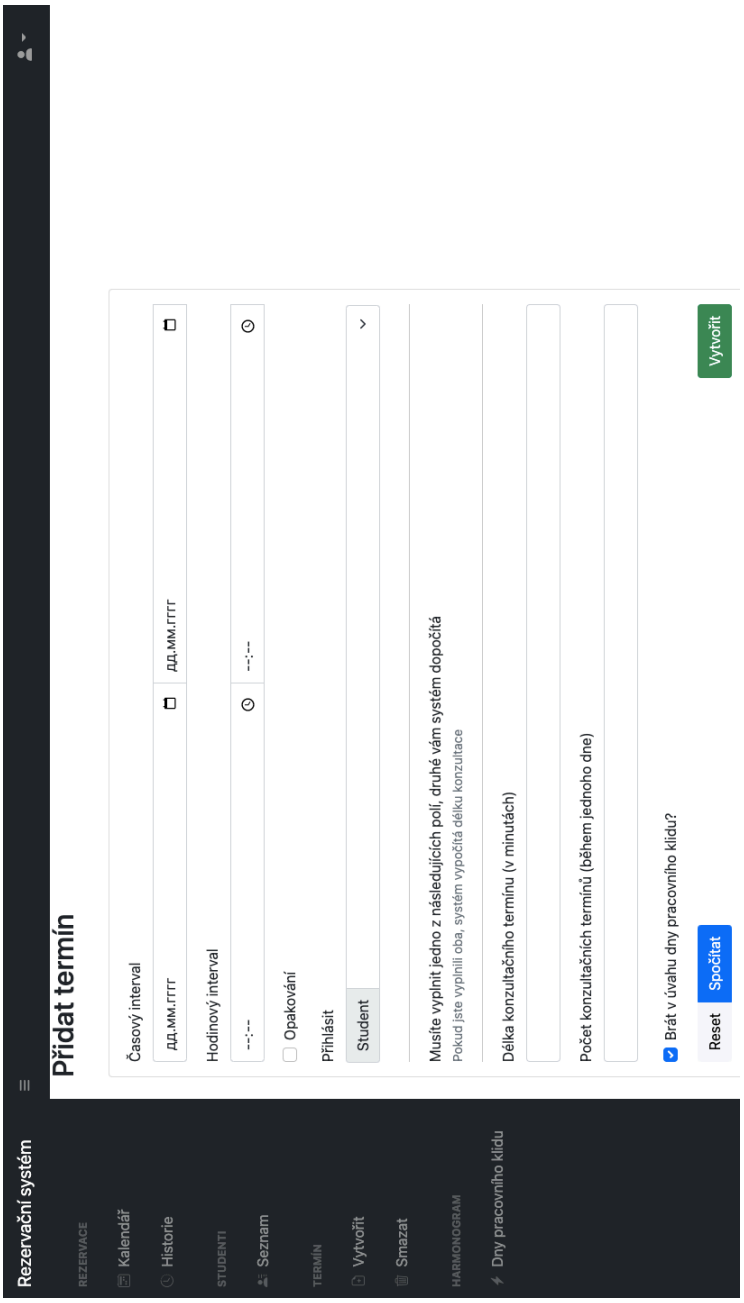
Příloha D

Ukázky aplikace

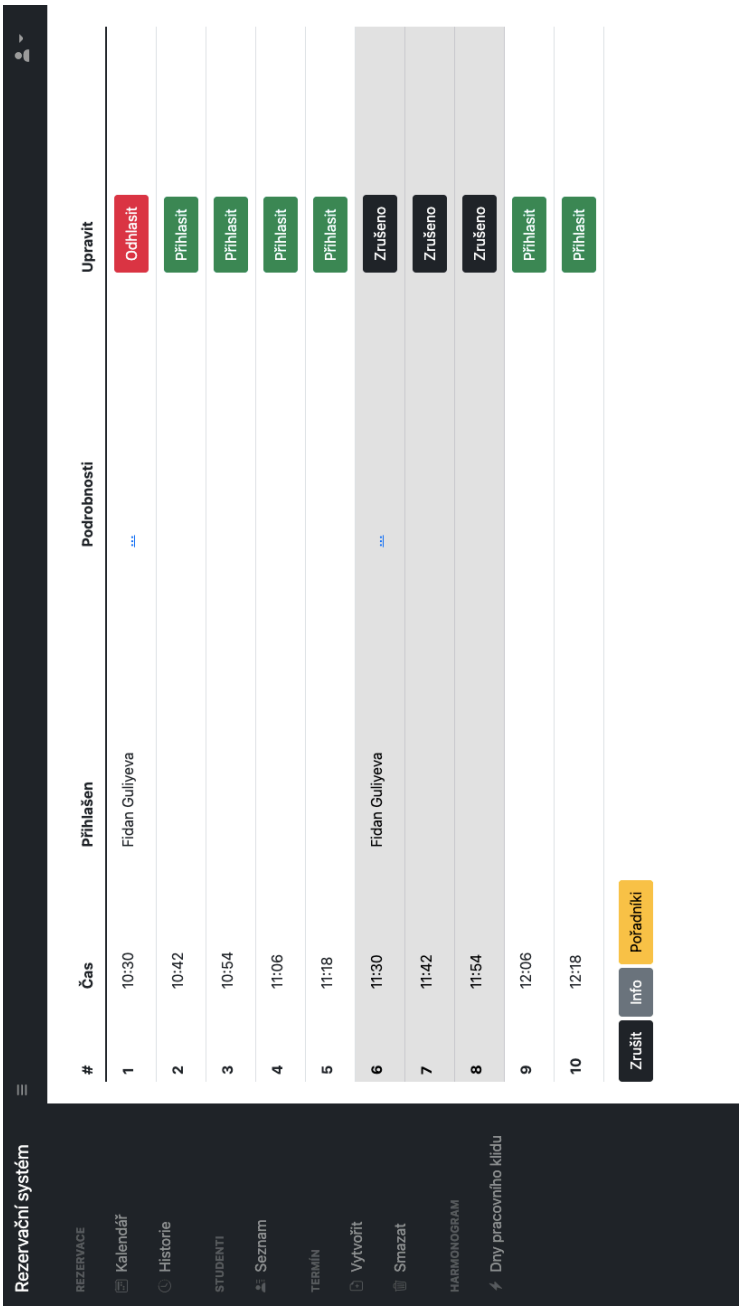
Obrázek D.1: Hlavní stránka aplikace.



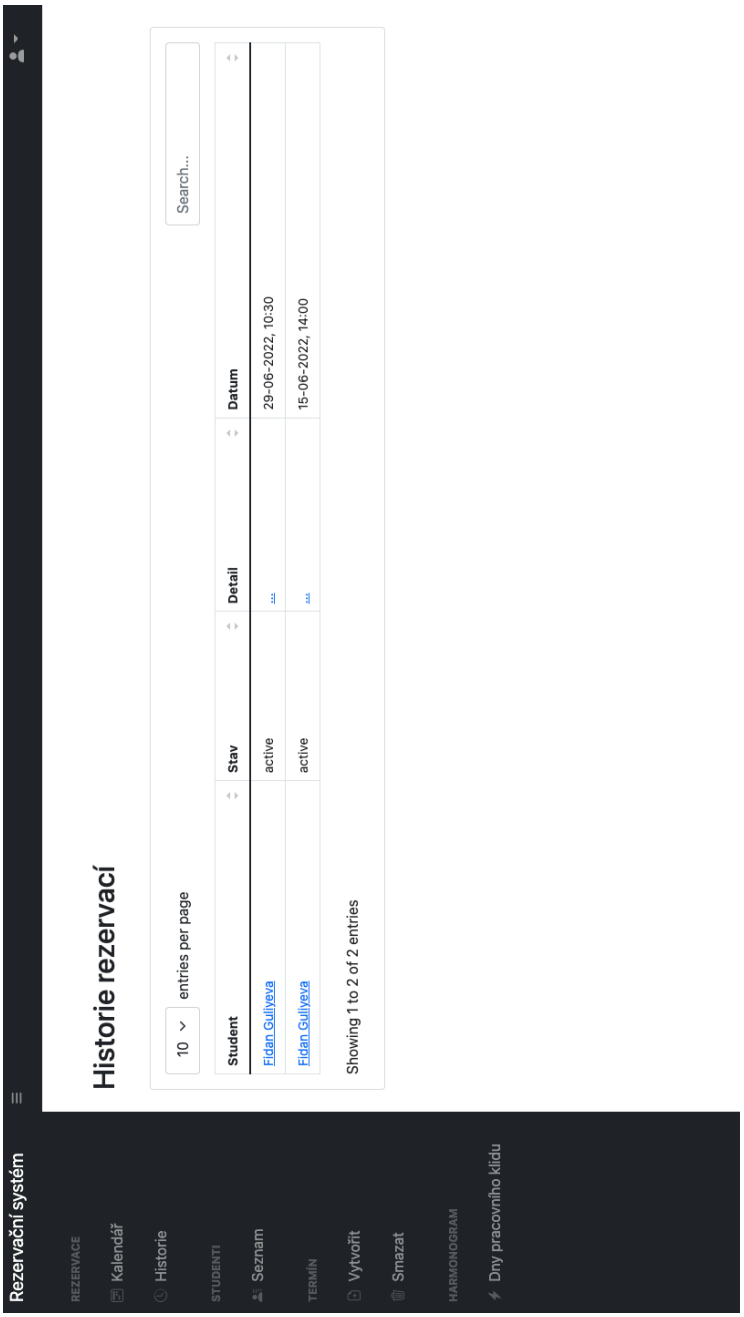
Obrázek D.2: Stránka pro vytváření termínů konzultací.



Obrázek D.3: Stránka pro správu rezervací.



Obrázek D.4: Stránka se seznamem všech rezervací a odkazy na jejich podrobnosti.



Příloha E

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `Instalační příručka.pdf` ..... instalační informace
  - `supervision_support_system` ..... zdrojové kódy implementace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `tex` ..... zdrojová forma práce ve formátu LaTeX