Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Školní informační systém

Martin Czakó

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Martin Czakó

Systémové inženýrství a informatika
Informatika

Název práce

Školní informační systém

Název anglicky

School information system

Cíle práce

Cílem práce je navrhnout školní informační systém.
Do systému budou zakomponovány tyto funkce: klasifikace, docházka a rozvrh hodin. Tyto funkce by měly sloužit k základní evidenci tříd a žáků ve škole.
Rozhraní systému bude rozděleno na dvě verze: pro žáky a pro učitele. Žáci budou mít možnost se podívat na vedené záznamy a učitelé budou mít právo tyto záznamy upravovat, či přidávat.

Metodika

Informační systém bude navržen použitím objektového paradigmatu. Nejdříve bude provedena analýza a návrh systému za použití UML. Následně bude systém implementovám za použití objektového programovacího jazyku.

K tvorbě webového rozhraní použiji HTML5. Stylizace stránky bude zajištěna jazykem CSS.

Data budou uložena do databáze a k jejich manipulaci budou použity jazyky SQL, PL/SQL. Data budou interpretována na základě typu uživatele (žák nebo učitel).

Do systému bude zakomponována i statistika. Data v databázi bude možnost zobrazit z pohledu žáka, či z pohledu celé třídy.

Jednotlivé funkce systému budou navrženy k co nejjednoduššímu použití ze strany uživatele.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

webový informační systém, školní informační systém, databáze, SQL, UML

Doporučené zdroje informací

ARLOW, J. – NEUSTADT, I. *UML 2 a unifikovaný proces vývoje aplikací : objektově orientovaná analýza a návrh prakticky.* Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

BRUCKNER, T. *Tvorba informačních systémů : principy, metodiky, architektury.* Praha: Grada, 2012. ISBN 978-80-247-4153-6.

KOSEK, J. *PHP – tvorba interaktivních internetových aplikací : podrobný průvodce.* Praha: Grada, 1999. ISBN 80-7169-373-1.

LUBBERS, P. – ALBERS, B. – SALIM, F. *HTML5 : programujeme moderní webové aplikace.* Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.

SUMMERFIELD, M. *Python 3 : výukový kurz.* Brno: Computer Press, 2010. ISBN 978-80-251-2737-7.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 13. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Školní informační systém" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2021 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) Ing. Marku Píckovi, Ph. D. za jeho vedení a rady k mé bakalářské práci.

Školní informační systém

Abstrakt

Tato bakalářská práce seznamuje čtenáře s pojmem informační systém. Pojednává se zde o tom, co to vlastně informační systém, k čemu slouží a také o jeho typech. Dále je zmíněn životní cyklus informačního systému a nejpoužívanější metodiky při jeho vývoji. Nakonec se čtenář dozví něco o modelovém jazyku UML a o technologiích použitých k vývoji praktické části této práce jako je například HTML, React a PHP.

Praktická část této práce obsahuje analýzu požadavků a s funkcemi, které má informační systém obsahovat. Zde se specifikují jednotlivé funkce pomocí Use Case diagramu a poté definování scénářů. Na analýzu navazuje návrhová část, kde je specifikován návrh databáze a jednotlivých tabulek. Dále je zde k nahlédnutí předběžný vzhled systému za použití wireframů. V implementační části je pospána výsledná MySQL databáze a funkce systému, jako například přihlášení do systému, nebo registrace nového uživatele. Je zde také zobrazen výsledný vzhled aplikace.

Klíčová slova: UML, informační systém, školství, HTML, PHP, React, MySQL

School information system

Abstract

This bachelor thesis introduces the reader to the concept of information system. It discusses what an information system is, what it is used for, and also its types. The life cycle of the information system and the most used methodology in its development are also mentioned. Finally, the reader will learn something about the UML model language and the technologies used to develop the practical part of this work, such as HTML, React and PHP.

The practical part of this work contains an analysis of requirements and functions that the information system should contain. Here, the individual functions are specified using a Use Case diagram and then scenarios are defined. The analysis is followed by the design part, which specifies the design of the database and individual tables. There is also a preliminary look of the system using wireframes. The implementation part describes the resulting MySQL database and system functions, such as logging into the system or registering a new user. The final appearance of the application is also displayed here.

Keywords: UML, information system, school, HTML, PHP, React, MySQL

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní době je používání internetu nedílnou součástí života velké části obyvatel. Lidé si zvykli mít potřebné informace na dosah ruky a poskytuje jim zábavu ve chvílích nudy.

V dnešní době je zapotřebí pracovat hodně z domova a je tedy potřeba umožnit zaměstnancům tuto možnost umožnit. K tomu slouží webové informační systémy, které poskytují vzdálený přístup datům firmy zaměstnance a dovolují mu s nimi manipulovat.

Tento přístup si osvojilo i školství, které už ve většině případů nepoužívá papírové třídní knihy na evidenci docházky, nebo žákovské knížky na evidenci známek studentů. Jak studenti, tak i učitelé mají přístup do školního informačního systému, který nahrazuje ono fyzické zpracování dat. Učitel do systému zadá data a student (či jeho rodič) si může tyto známky prohlédnout z pohodlí svého domova.

Teoretická část této bakalářské práce seznamuje s hlavním pointou informačních systému, dále představuje nejznámější typy informačních systémů, problematiku při jejich vývoji a technologie potřebné k vývoji, společně s jejich výhodami a nevýhodami.

Praktická část se zabývá analýzou a návrhem školního informačního systému. Systém obsahuje 3 rozhraní, dle typu uživatele: student, učitel a administrátor. Informační systém je následně implementován pomocí technologií pro tvorbu webový stránek: HTML s CSS, Javascript (React) a PHP.

Motivace k vytvoření této práce byla vlastní nemilá zkušenost autora se školním informačním systémem, který byl zaveden při jeho studiích na střední škole. V té fázi se ještě používala papírová třídní kniha současně s třídní knihou v informačním systému. Autor měl na starosti zadávání docházky do systému, což nebylo moc uživatelsky přívětivé.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je vytvoření informačního systému pro základní a střední školy. Informační systém bude rozdělen na tři části kvůli požadavku mít oddělené funkce podle typu uživatele (student, učitel a administrátor). Systém umožní administrátorovi spravovat uživatele, třídy, předměty a rozvrh hodin. Učitelům bude zpřístupněno zadávání známek, zkoušek a docházky. Studenti si potom tato data budou moci pouze zobrazit.

Dílčím cílem práce bude analýza a návrh systému. Návrh systému bude proveden za použití jazyka UML.

2.2 Metodika

Teoretická část bakalářské práce obsahovat studium odborných informačních zdrojů. Zde bude popsáno, co je to informační systém, nejznámější typy informačních systémů, jakými stádii prochází informační systém při jeho vývoji a jaké modely jsou k tomu používány.

V praktické části nejdříve proběhne analýza potřeb pro systém. Budou zde sepsány všechny funkce, které jsou od systému očekávány a jaký má být jejich výsledek. K analýze bude použit modelový jazyk UML pro vytvoření diagramů a postupů, které budou potřeba při implementaci. Dále proběhne návrh aplikace. Tato část bude popisovat návrh databáze pro celý systém, jaké atributy budou mít jednotlivé tabulky a jak na sebe budou vázat. Budou zde zobrazeny také wireframy pro obecnou představu, jak bude nakonec informační systém vypadat. Nakonec proběhne samotná implementace návrhu řešení informačního systému. Databázová stránka bude řešena pomocí databáze MySQL. Komunikace s ní bude probíhat přes objektově orientovaný jazyk PHP (back-end). Přední stránka aplikace (front-end) bude tvořena značkovacím jazykem HTML a s ním spojenými kaskádovými styly CSS. Logická část stránky bude provedena přes JavaScript a jeho knihovnu React. Komunikace mezi front-end a back-end částí bude realizována pomocí knihovny AXIOS, která zavolá script v PHP souboru a příjme od nich výsledná data. React tato data následně vykreslí do prohlížeče.

Po dokončení projektu bude moct uživatel přidávat data (známky, docházku, absenci) a následně si je i zobrazit.

3 Teoretická východiska

3.1 Informační systém

Informační systém je soubor komponent pro sběr, ukládání a zpracování dat pro poskytování informací, znalostí a digitálních produktů.

Podniky a korporace používají informační systémy k interakci se svými dodavateli a zákaznickou základnou k provádění svých operací, ke správě své organizace a provádění svých marketingových kampaní. (1)

3.1.1 Komponenty informačních systémů

Mezi hlavní komponenty informačních systémů se řadí hardware a software, telekomunikace, databáze a datové sklady, lidské zdroje a zavedené postupy (procedury). Hardware, software a telekomunikace společně tvoří Informační technologie (IT), které se nachází dneska skoro ve všech organizacích. (1)

3.1.2 Využití informačních systémů

Informační systémy jsou dneska skoro na každém rohu. Používají je například univerzity, knihovny, banky nebo nemocnice. V knihovnách slouží k evidenci knih, jejich výpůjček, databázi čtenářů apod., v nemocnicích slouží zase k evidenci pacientů z jednotlivých odděleních. Mají své využití i v podnicích či ve firmách (2):

- Lidské zdroje – evidence zaměstnanců, nábory, docházka zaměstnanců, školení, aktivita zaměstnanců apod.
- Správa majetku – přehled o technickém vybavení (komu bylo zařízení přiřazeno, jeho stav a záruka)
- Správa budov – evidence poboček a budov ve vlastnictví
- Bezpečnost dat – přehled o přístupových právech k jednotlivým sadám informací
- Logistika a doprava – přehled zboží na skladech, výroba nových produktů, rozvoz zboží, dodací lhůty

3.1.3 Klasifikace informačních systémů

Hledisek pro klasifikaci systémů je několik – od komplexnosti, přes účel až po vztah k systému řízení uživatele. Podle vztahu k systému řízení uživatele dělíme systémy na:

3.1.3.1 Transakční systém (TPS)

Tento systém představuje systémy automatizující zpracování typických úloh, jako je například účetnictví, skladové systémy, různé evidence, rezervační systémy. Jedná se tedy o systémy, kde převážná část práce s daty je provedena při vložení a výsledek je na první pohled viditelný. S tímto typem systému se setkáváme nejčastěji. (3)

3.1.3.2 Informační systém pro řízení (MIS)

Hlavním cílem MIS je zobrazení různých přehledů či soustav, ať už jsou to počty objednávek, nebo zisk v jednotlivých měsících. Používají je zejména řídící pracovníci, především ti v oblasti kontroly výkonnosti svých podřízených, až celého oddělení. (3)

3.1.3.3 Systém pro podporu rozhodování (DSS)

Jedná se o nadstavbu MIS. Umožňuje řídícím pracovníkům provádět různé analýzy pro přijmutí důležitých rozhodnutí. Pro lepší přehlednost tento systém umožňuje prezentaci pomocí grafických výstupů. (3)

3.1.3.4 Informační systém pro vrcholové řízení (EIS)

Funkce tohoto systému je být jakýsi pomyslný integrační prvek mezi všemi třemi typy informačními systémy. Jeho úkolem je poskytnout důležité informace vedoucím pracovníkům, na základě kterých by oni mohli učinit rozhodnutí prospěšnější pro budoucnost organizace. (3)

3.2 Vývoj informačního systému

Tato kapitola popisuje fáze vývoje, kterými systém prochází od započetí prací na jeho vzniku, až po samotný zánik systému. Tomuto časovému úseku (také „život“ IS) se říká životní cyklus informačního systému.

3.2.1 Předběžná analýza (specifikace cílů)

První fáze životního cyklu, která má na starost shromáždit požadavky uživatelů a cíle organizace. Dále sem patří i analýza současného stavu systému, kde se stanový jeho nedostatky a navrhnou se změny. Požadavky se dále rozeberou a odhadne se doba realizace a její náklady. Cílem je tedy pouze náčrt základního rámce požadavků, cílů a funkcí, bez podrobného zkoumání. Výsledek této fáze by tedy měl být následující (4):

- Odhad doby trvání vývoje systému
- Seznam potřebných zdrojů na vývoj systému
- Odhad funkčnosti, rozsahu systému a odhad návratnosti investice

3.2.2 Analýza systému (specifikace požadavků)

Tato fáze navazuje bezprostředně na fázi předchozí. V této části se provede detailní rozbor výsledků z předešlé části za účelem odhalení všech případných chyb ve struktuře dat i v systému samotném. Chyby, které se zde neodhalí jde později odstranit velmi těžce. (4)

3.2.3 Projektová studie (návrh)

Tato část je výsledkem analýzy systému. Výsledkem projektové studie je podklad pro obsah smlouvy s externí firmou, očekávaný časový harmonogram, cena vyvíjeného systému, postup a podmínky zavádění IS, podmínky pro záruční servis apod.

Dále by zde měly být uvedeny základní informace o tvůrcích systému, specifikace zákaznické organizace, globální (logický) a detailní (fyzický) návrh IS, který obsahuje funkční analýzu systému, datovou analýzu, popis veškerých datových toků atd. (4)

3.2.4 Implementace

V této fázi jde na řadu vlastní programování systému. Podkladem pro vývoj jsou všechny podklady shromážděné předchozími etapami. Výsledkem by tedy měl být hotový systém, který se následně ověří a připraví se testovací data, obsahující maximální procento konečných reálných dat. (4)

3.2.5 Testování

Podle názvu je již jasné, že v této fázi se provádí testování hotového informačního systému. Principem této fázi je otestovat veškeré možné funkce, ověřit jejich reakce na zadávaná data a zaznamenat nedostatky. Tyto nedostatky následně půjdou k opravě. Testování má zabránit vypuštění chybující aplikace do provozu, což by mohlo mít rozsáhlé následky. (4)

3.2.6 Zavádění systému

V této fázi dochází k zavádění systému v organizaci. Zavedením se myslí jeho instalace, převedení datové základy do podoby využitelné novým systémem a dále také školení nových uživatelů. Je možné zvolit z více postupů zavedení IS (4):

- **Souběžná strategie** – nový systém je provozován v souběžném jetí se starým systémem. Tento provoz trvá několik cyklů, dokud nejsou uživatelé s novým systémem dostatečně seznámeni a systém nepracuje spolehlivě. Jedná s o **bezpečnou** metodu, avšak náročnou na provoz – všechny operace se musí provádět dvakrát.
- **Pilotní strategie** – nový systém se zavede do vybrané části podniku a po jeho ověření se nasadí do celé organizace.
- **Postupná strategie** – využívá se u složitých systémů. Nejprve se nasadí primární části IS a až po jejich ověření se postupuje stejným způsobem až po zavedení celého systému.
- **Nárazová strategie** – spočívá v odstranění starého systému a nasazení nového. Jedná se o velice riskantní strategii.

3.2.7 Zkušební provoz

V této fázi má poskytoval systému za úkol zajistit okamžitý servis, odstranit chyby objevené při provozu, a nebo dořešit požadavky uživatelů v rámci původního návrhu. (4)

3.2.8 Rutinní provoz a údržba

Poslední fáze životního cyklu systému, ve které je systém provozován a používán. Spadá zde také údržba systému (zajištění správného provozu), úprava parametrů nebo vytvoření nových funkcí podle přání uživatele. (4)

3.2.9 Hodnocení

Jedná se o přehodnocení požadavků na systém a pokud požadavky nelze splnit jednoduchou úpravou, pak se životní cyklus opakuje. (4)

3.3 Základní modely životního cyklu systému

3.3.1 Model vodopád

Model vodopád, někdy také SWD (Systém Development Method) má charakteristiku ve zpracování jednotlivých etap životního cyklu a to jednu za druhou. Každá etapa se realizuje podle přesného plánu, po jejím dokončení předá výstup etapě následující a zpětně se k ní už nevrací (stejně jako tok vody vodopádu).

Tento model se používá jako základní model životního cyklu pro stavbu automatizovaných systémů řízení už od 70. let. Jeho cílem vzniku bylo zavedení hierarchické dekompozice a minimalizace potencionálních chyb. (4)

Je vhodný k použití při návrhu systému, kde je přesně známý problém a způsob jeho řešení.

Výhody:

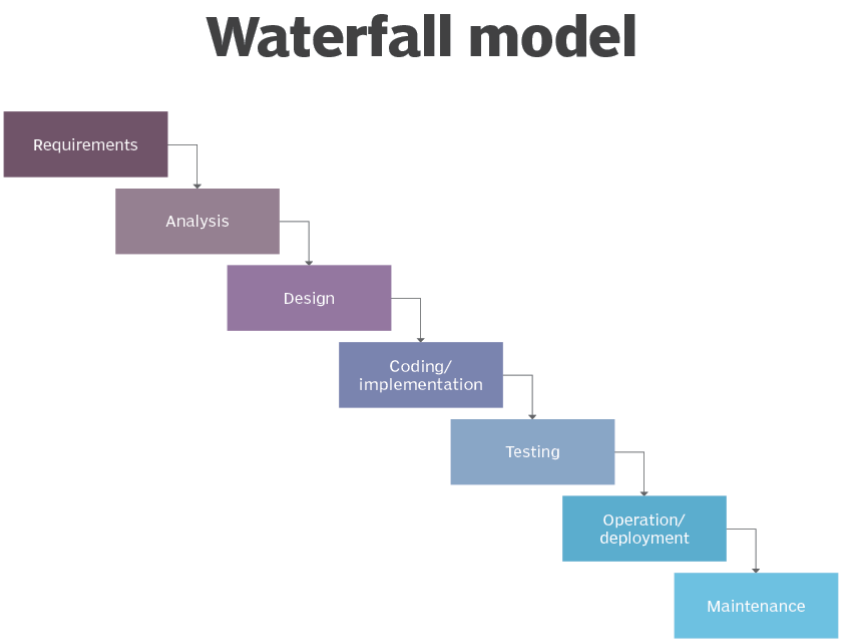
- Rychlost (pokud je snadno definovaný cíl a způsob jeho dosažení)
- Nízké náklady

Nevýhody:

- Pouze malé množství projektů jde řešit tímto modelem
- Úspěšnost systému zjistíme až po dokončení poslední fáze, tedy až po jeho předání
- Předání systému je až po delší době

Model lze chápat jako univerzální model, který je lepší použít i přes své nevýhody, než nepoužít žádný model.

Obrázek 1 - Model vodopád



Zdroj: https://cs.photo-555.com/4611920-waterfallmodel

3.3.2 Prototypový model

Na rozdíl od modelu vodopád je charakteristika tohoto modelu předpoklad změn výchozích požadavků ze strany zákazníka a následná reakce na tyto změny, což dělá tento model více dynamičtější. Hlavním cílem tohoto modelu je urychlení vývoje IS za pomoci použití připravených prototypů (zjednodušená implementace systému / plná implementace systému). Lze tedy zákazníkovi předvést v co nejkratším čase první verze systému a na základě jeho připomínek následně systém upravit. (4)

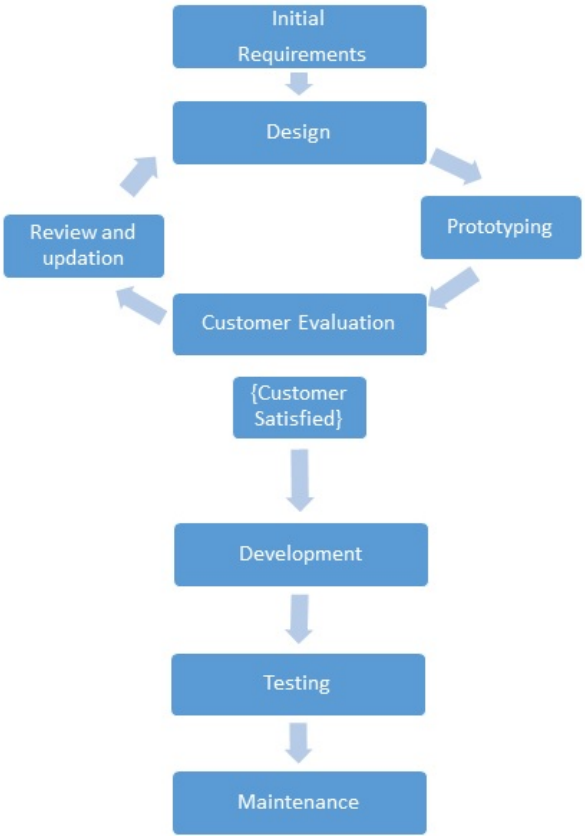
Výhody:

- Umožňuje reagovat na zákazníkovi změny v požadavcích

Nevýhody:

- Znační náročnost při použití u větších projektů

Obrázek 2 - Prototypový model



Zdroj: https://www.includehelp.com/basics/the-prototyping-model-software-engineering.aspx

3.3.3 Model spirála

V roce 1998 vytvořil B. M. Boehm spirálový model kombinací prototypového přístupu a analýzy rizik. Podstatou modelu je neustálé opakování vývojových kroků tak, že v každém dalším kroku se na ověřenou část přibalí část na vyšší úrovni. Postup vývoje je poté stejný, jako u modelu vodopád a každý další krok se skládá z následujících část (4):

- Komunikace se zákazníkem (specifikace cílů a určení plánu řešení) a následně určení plánu řešení
- Vyhodnocení alternativ řešení a analýza rizik s daným řešením souvisejících
- Vývoj prototypu určité úrovně a jeho analýza
- Zpětná revize požadavků zákazníka (validace) – test správné funkčnosti prototypu
- Kontrola, zda výstup daného kroku souhlasí s požadavky zákazníka

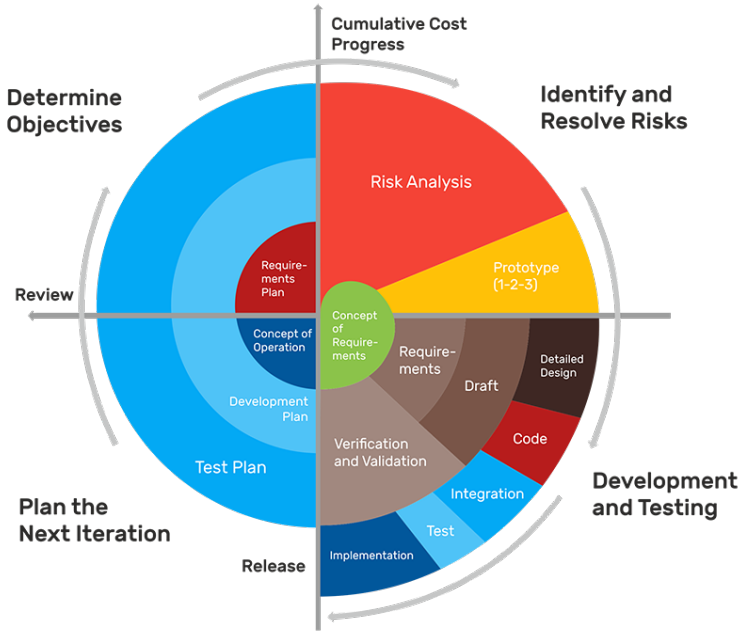
Výhody:

- Minimální riziko chyb díky využití ověřeného postupu vývoje a analýzy
- Možnost modifikace požadavků zákazníkem v konkrétních krocích
- Sledování vývoje systému a možnost ho zhodnotit

Nevýhody:

- Nevhodné pro vývoj systém na dálku – je zde potřeba neustálá komunikace se zákazníkem
- Nelze určit přesný termín dokončení vývoje, cenu a jednotlivé výstupy
- Neodhalení chyb v počátečních fází projektu má vliv na celý systém (stejně jako u modelu vodopád)

Obrázek 3 - Spirálový model



Zdroj: https://blog.codegiant.io/software-development-life-cycle-the-ultimate-guide-2020-153d17bb20fb

3.4 Unified Modeling Language (UML)

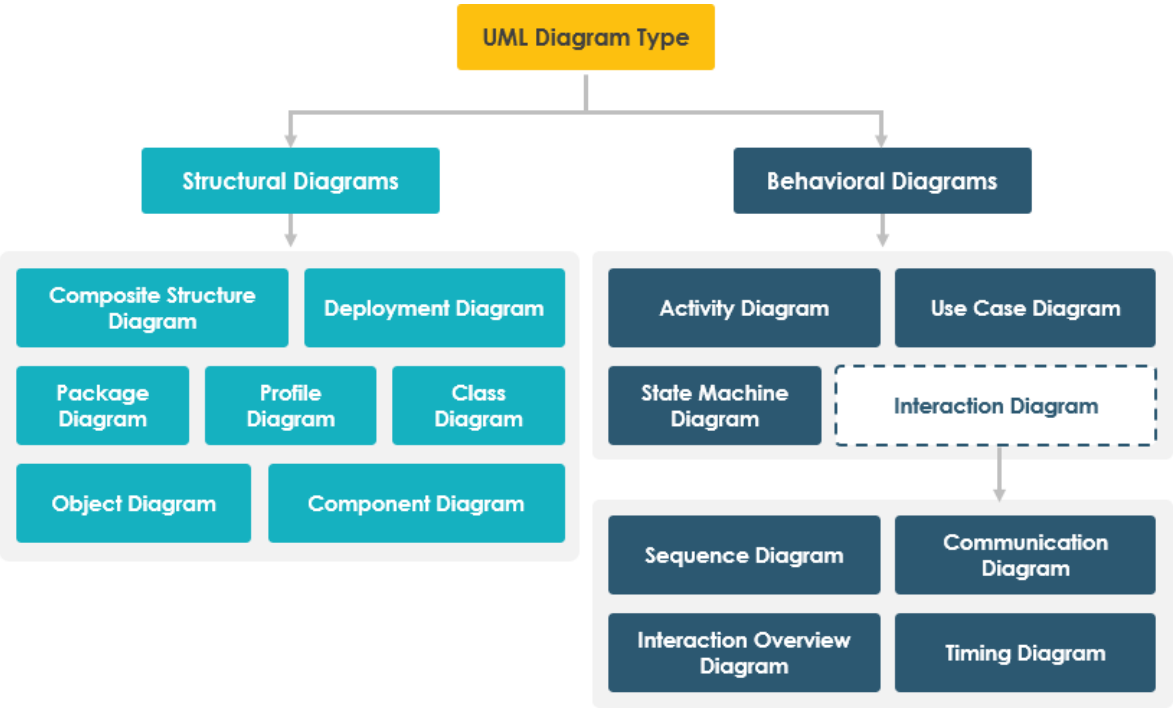
Jednotný modelovací jazyk, jenž se skládá ze souboru grafických notací používaných při vývoji softwaru. Používá se pro vizualizaci, specifikaci, navrhování a dokumentaci navrhovaných systému. (5)

UML se stal standardem pro objektivně orientovanou analýzu a design a používaný dodnes. Od verze 2.0 obsahuje celkem 14 diagramů, které se dělí na strukturní diagramy a diagramy chování, které dále obsahují ještě diagramy interakcí. (5)

Strukturální diagramy popisují strukturu systému, nezávisle na čase. Jedná se o konkrétní objekty pro sestavení modelu, jako jsou třídy, objekty, rozhraní a fyzické komponenty. (6)

Diagramy chování charakterizujeme jako stavy vzájemného působení okamžitých vlivů a jejich průběh v čase. Tyto diagramy mají ještě podskupinu diagramů interakcí, popisující interakci mezi jednotlivými částmi systému. (6)

Obrázek 4 - Typy diagramů UML

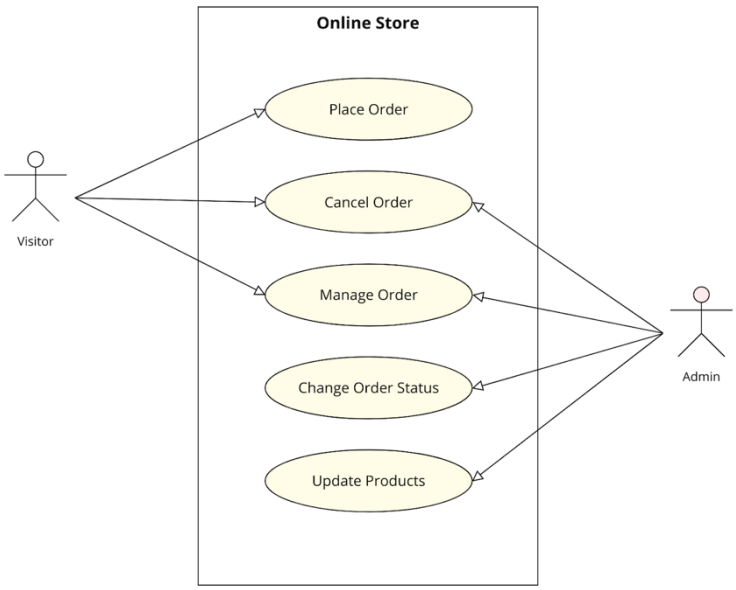


Zdroj: https://warren2lynch.medium.com/a-comprehensive-guide-to-14-types-of-uml-diagram-affcc688377e

3.4.1 Diagram užití (Use Case Diagram)

Tento diagram zobrazuje chování systému z pohledu uživatele. Jedná se o sadu několika akcí k dosažení určitého cíle. Může jít například o registrování nového uživatele, přidání nového článku, nebo výpis dat. Výsledkem je tedy popis jedné funkce, kterou by měl systém umět. Do diagramu však už nepatří logika schovaná za popisovanou funkcí. Nejčastěji je znázorněn elipsou s názvem funkce uvnitř. Dále jsou v diagramu potřeba aktéři (uživatelé), kteří komunikují se systémem a inicializují případ užití (např. přidání nového komentáře). (7)

Obrázek 5 - Příklad Use Case diagramu



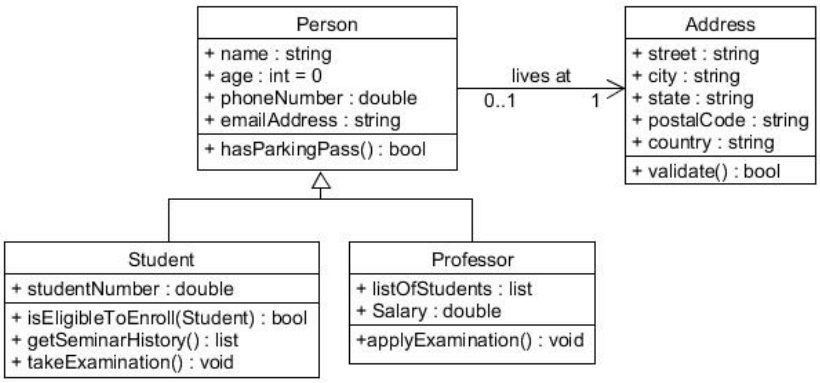
Zdroj: https://moqups.com/templates/diagrams-flowcharts/uml-diagrams/use-case-diagram/

3.4.2 Diagram tříd (Class Diagram)

Diagram tříd poskytuje statický náhled na základní systémovou strukturu. Jedná se o digram implementace, což znamená, že pokud ho programátor přepíše do kódu, kód musí fungovat. Diagram musí být tedy úplný. Třídy budou obsahovat všechny atributy a metody. Jelikož má každý programovací jazyk jinačí typy atributů a metod, tak i diagram tříd je pro každý jazyk jinačí. Třídy se následně spojují jedním z těchto vztahů (8):

- Realizace – mezi interface a třídou, která toto interface implementuje. Třídy reprezentující interface mají tzv. stereotyp, který se píše do dvojitých špičatých závorek.
- Asociační třída – třída zprostředkovávají vztah mezi 2 entitami.

Obrázek 6 - Příklad Class diagramu

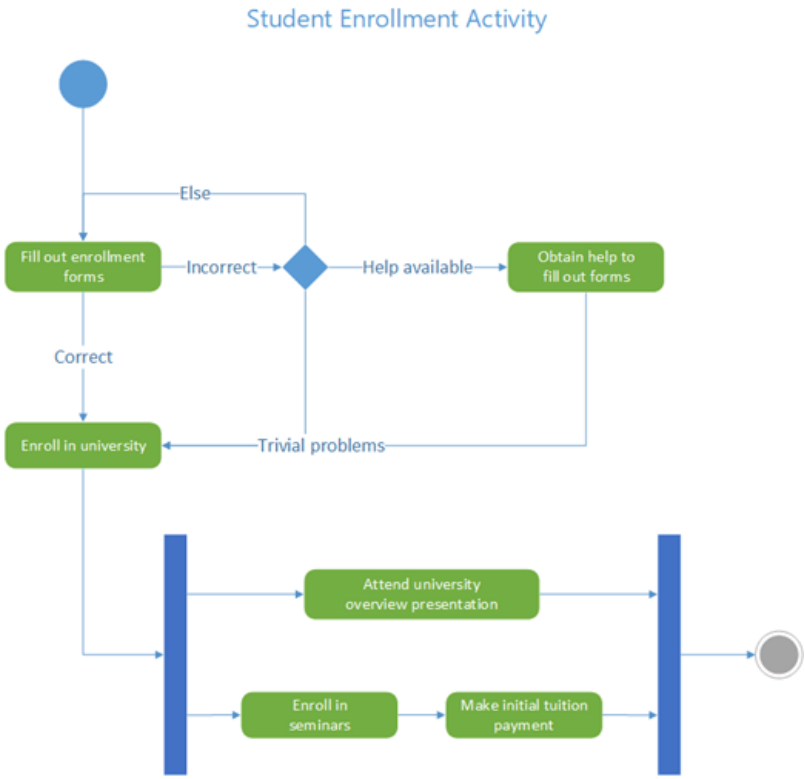


Zdroj: https://www.researchgate.net/figure/Simple-UML-class-diagram_fig1_314175216

3.4.3 Diagram aktivit (Activity Diagram)

Diagram aktivit modeluje procesy jako aktivity (sekvenční i paralelní), které se skládají z uzlů, který jsou vzájemně propojený hranami. Popisuje nejčastěji procedurální logiku, byznys procesy, nebo pracovní postupy. (9)

Obrázek 7 - Příklad Activity diagramu



Zdroj: https://support.microsoft.com/de-de/office/create-a-uml-activity-diagram-19745dae-2872-4455-a906-13b736f01685

3.5 Použité technologie

3.5.1 HTML

HTML je zkratka pro Hyper Text Markup Language. Je to základní stavební kámen při tvorbě webových stránek. HTML je kombinací hypertextů a značkovacího jazyka. Hypertext definuje propojení webových stránek. Značkovací jazyk definuje text dokumentu uvnitř prvku, který společně s ostatními prvky tvoří strukturu stránky. Většina značkovacích jazyků je čitelná pro lidi. Aktuální verze jazyka HTML je verze 5.2. (10)

3.5.1.1 Historie HTML

První verzi jazyka HTML vytvořil v roce 1991 Tim Berners-Lee. Mělo se jednat o součást projektu WWW. Tato první verze nikdy nebyla myšlena pro běžné používání. Projekt WWW byl určen pro vědce zabývající se fyzikou vysokých energií, aby si mohli na velké vzdálenosti přenášet informace o svých vědeckých projektech a poznatcích. Celý projekt vznikal v CERNu (Centre Européeenne pour la Recherche Nucléaire, Evropské centrum jaderného výzkumu) nacházející se na švýcarsko-francouzských hranicích nedaleko Ženevy. Myšlenka byla, aby výsledek používání HTML jazyka dokázal přečíst i člověk neznající jeho strukturu. (11)

První verze jazyka byla popsána v dokumentu „HTML tags“. Tehdy jazyk HTML dokázal pouze rozčlenit text do několika logických úrovní, použít základní stylizaci (zvýraznění textu) a přidat do textu odkazy a obrázky. (11)

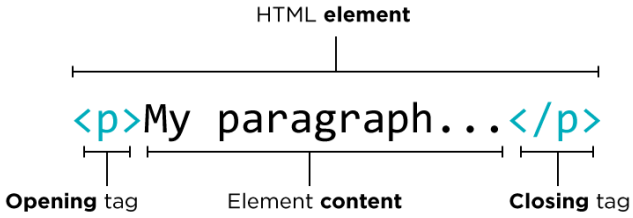
V roce 1991 byl CERNem vyvinut software včetně specifikací jazyka HTML a stal se veřejně dostupným. Kvůli narůstajícím požadavkům od uživatelů na WWW bylo HTML obohacováno producenty různých prohlížečů a nové prvky. Aby ale byla zachována kompatibilita, vytvořil Berners-Lee nový návrh standardu HTML 2.0 pod hlavičkou IETF, který zahrnoval starou verzi HTML a všechny nadstavby z prohlížečů. Do této verze byly přidány nově i formuláře. (11)

Další menší verze pod názvem HTML+ obsahovala rozšíření HTML o možnost vytváření tabulek a matematických vzorců. Začaly se zde objevovat i prvky komplexnější stylizace textu. Tuto verzi následně formalizoval Dave Ragett z laboratoří Hawlett-Packard a vytvořil jeho definici jako DTD (Document Type Definition). Následně na jaře roku 1995 vznikl nový standart HTML 3.0. (11)

3.5.1.2 Základní struktura HTML

HTML používá předpřipravené prvky a elementy, které prohlížeči řeknou, k čemu jaký prvek slouží a jak ho zobrazit. Většina prvků je párových (musí se na jejich konci uzavřít), najde se ovšem i pár nepárových. (10)

Obrázek 8 - Párový HTML prvek



Zdroj: https://studio.code.org/docs/csd/html_tags/index.html

Následující obrázek obsahuje základní prvky HTML, na kterých je webová stránka postavena.

Obrázek 9 - Startovací HTML kód

```
<!DOCTYPE html>
<html>
 <head>
  <title>Titulek stránky</title>
 </head>
 <body>
  <h1>Nadpis stránky</h1>
  <p>Toto je <a href="http://example.com/">odkaz</a> v odstavci.</p>
  <!-- toto je komentář -->
 </body>
</html>
```

Zdroj: https://sites.google.com/site/jaknatosnikolneu/web-a-html/jak-na-html-zkratky-kde-tvorit

<!DOCTYPE html> - deklarace dokumentu jako HTML

<html> - root element, který obsahuje všechny ostatní prvky

<head> - hlavička dokumentu obsahující dodatečné prvky definující výslednou stránku. Obsahuje například element <tag>, který nám říká, jaký bude název stránky v panelu záložek prohlížeče. Dále také může obsahovat element <link>, který definuje cestu k souboru s kaskádovými styly. Pokud programátor nechce mít zvlášť soubor na kaskádové styly, může do hlavička přidat párový element <style>, do kterého se následně vloží kaskádové styly.

<body> - tělo celé stránky. Všechno uvedené v tomto elementu se následně zobrazí uživateli v prohlížeči.

3.5.2 CSS

Kaskádové styly, v angličtině Cascading Style Sheets (CSS) je jazyk pro stylizaci HTML elementů. V dnešní době jde HTML a CSS ruku v ruce, jelikož se lidé začali dívat spíše na estetickou stránku webových stránek, než na obsahovou. Hlavním cílem CSS je umožnit vývojáři oddělit vzhled dokumentu od jeho struktury a obsahu.

Obrázek 10 - Způsoby psaní CSS

Zdroj: https://www.bitdegree.org/learn/how-to-link-css-to-html

Pomocí CSS můžete kontrolovat barvu textu, styl fontů, řádkování textu v odstavcích, rozložení sloupců a řádku v tabulkách, barvu pozadí apod.

Výhody CSS:

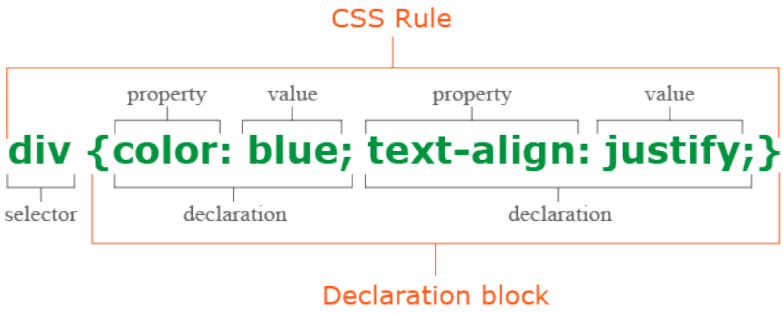
- **Šetření času** – jeden soubor se styly jde použít v dalších HTML stránkách, na rozdíl od vkládání stylů přímo do .html souboru
- **Rychlejší načítání stránek** – díky specifikaci stylů přímo pro prvek šetříte místem a tím pádem ulehčíte práci prohlížeči
- **Více možností stylizace** – oproti HTML nabízí CSS mnohem více možností, jak prvky stylizovat

První verze CSS level 1 byla vydána v prosinci 1996 organizací W3C. Tato verze popisuje jazyk CSS a jednoduché modely formátování pro všechny HTML prvky. Následovala verze CSS level 2 v roce 1998 a poté zatím poslední verze CSS3 v roce 2016, do které byly přidány např. prvky pro práci s multimédii, barvené modely RGBA, HSL, HSLA, podporu zaoblených rohů a stínování. (12)

3.5.2.1 Syntaxe

Základní syntaxí CSS je tzv. set pravidel, které popisují formátování (změnu zobrazení) jednotlivých elementů na webové stránce. Jedno pravidlo se musí skládat ze selectoru (název elementu, který chceme upravit) a z deklaračního bloku.

Obrázek 11 - Ukázka definice CSS stylu



Zdroj: https://puzzleweb.ru/en/css/1_css_syntax.php

Jak je vidět na obrázku, tak selektorem je v tomto případě element *div*. Do složených závorek se poté píšou jednotlivé deklarac. Každá deklarace se skládá z vlastnosti a její hodnoty. V našem případě je to vlastnost *color* a nastavujeme jí hodnotu *blue*. Tím se elementu *div* nastaví barva písma na modrou. Dále je zde definice pro vlastnost *text-align* a její hodnota *justify*, což nastaví délku každé věty na stejnou hodnotu.

3.5.3 Javascript

Javascript je multiplatformní, objektově orientovaný scriptovací jazyk používaný při vývoji webových stránek za účelem obohacení HTML prvků interakcemi. Javascriptem se dají naprogramovat různé funkce tlačítek po jejich kliknutí, dále komplexní animace, popup menu ad.

Javascript obsahuje standardní knihovnu objektů, jako je např.: Array (pole), Date (datum), Math (možnost provádění matematických opearací).

Javascript vyvinul Brendan Eich v roce 1995 pod názvem LiveScript a poprvé se objevil v tehdy populárním prohlížeči Netscape. Souvislost mezi JavaScriptem a programovacím jazykem Java žádná není. Javascript je hodně ovlivněn programovacím jazykem C. (13)

Přidání scriptu do html dokumentu je stejně jednoduché, jako přidání CSS. Existují 2 způsoby: externě a interně. Při externím přístupu stačí dát do hlavičky .html souboru element **<script type="text/javascript" href=".javascript.js"></script>** a následně se

může script psát do uvedeného souboru v href parametru. Interně se myslí, že script bude součástí .html souboru. Do dokumentu se vloží stejný element **<script></script>** a script se píše do něj.

Javascript podporuje import knihoven. Knihovna obsahuje předpřipravený JavaScript kód, který následně může být použit při vývoji. Využití knihoven ulehčí čas při programování, jelikož nemusíte vymýšlet kód, který už byl někým sepsán. (14)

3.5.3.1 React

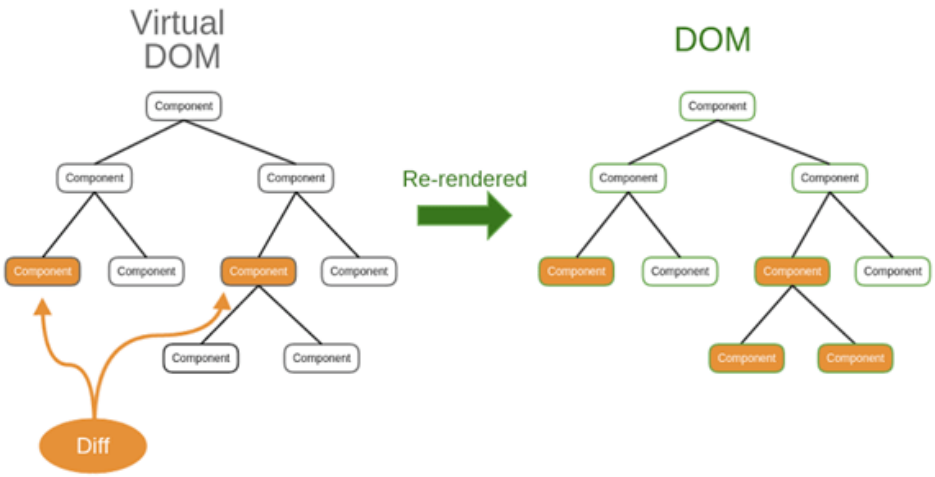
React je JavaScript knihovna, která pomáhá programátorům při tvorbě rychlého a interaktivního uživatelského rozhraní (user interface, neboli UI). Uživatelské rozhraní je ve webové terminologii kolekce menu, vyhledávacích lišt, tlačítek a čehokoliv dalšího, s čím může uživatel mít nějakou interakci.

React vytvořil Jordan Walke, softwarový inženýr ve společnosti Facebook. (15)

Výhody:

- **Lehká tvorba dynamických aplikací** – React ulehčuje tvorbu dynamický webových aplikací, protože nevyžaduje tolik kódování a nabízí více funkcí.
- **Vylepšený výkon** – React používá Virtual DOM a tím vytvoří aplikaci rychleji. Virtual DOM porovnává předchozí stavy komponent a aktualizuje pouze ty, které byly nějak upraveny.
- **Znovu používání komponent** – webová React aplikace je tvořena komponentami, kde každá komponenta má svoji logiku a je možné jednu komponentu použít vícekrát.
- **Lehký k naučení** – jelikož se React podobá hodně JavaScriptu a HTML, nemají programátoři sběhlý s těmito technologiemi problém přejít na React.

Obrázek 12 - Princip Reactu



Zdroj: https://dev.to/migueloop/my-collection-of-react-interview-questions-part-13-1d0b

Oproti JavaScriptu obsahuje React rozšíření **JSX** (JavaScript eXtension), které umožňuje Reactu používat HTML prvky ve stejném souboru.

Obrázek 13 - Ukázka React a JSX

```
const name = 'Simplilearn';
const greet = <h1>Hello, {name}</h1>;
```

Zdroj: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

3.5.4 PHP

PHP (Hypertext Preprocessor) je scriptovací jazyk, který se zpracovává na straně serveru (server-side), neboli back-end aplikace. Je používán k přijímání a manipulaci dat z formulářů, dokáže pracovat s databázemi, umí generovat dynamický obsah stránky, umí pracovat s cookies a session atd. (16)

3.5.5 MySQL

MySQL je relační databáze typu DBMS (Database Managment Systém), vlastnění společností Oracle. Jde o multiplatformní databázi, která komunikuje prostřednictvím jazyka SQL. Běžně se používá v kombinaci s PHP, Linuxem a Apache. MySQL dokáže pracovat s různými daty (obrázky, text, apod.) a umožňuje vytvářet tabulky včetně editace a mazání jejich záznam. Nejčastějším nástrojem pro správu databáze je phpMyAdmin, který umožňuje správu přes webové rozhraní. (17)

4 Vlastní práce

Cílem práce je navrhnout informační systém pro školy, které předchází z papírového systému na systém elektronický. Tento systém není dělaný pro specifickou školu.

Přední strana (front-end) systému je navržena za pomocí jazyků HTML a CSS v kombinaci se skriptovacím jazykem JavaScript a knihovnou React. Zadní strana (back-end) obsahuje programovací jazyk PHP, který komunikuje s databází MySQL.

Skrze informační systém budou moci učitelé předávat svá data studentům. Budou moci přidávat známky a docházku do databáze a nebo tato data spravovat. Student v systému nebude moci provádět žádné úpravy, bude si moct data pouze zobrazit.

4.1 Požadavky systému

Hlavním požadavkem pro informační systém je zobrazení povolení konkrétních funkcí jenom určité roli uživatele. S tímto požadavkem souvisí i požadavek na přihlášení uživatele, při kterém se určí, o jaký typ uživatele jde.

Požadavky pro roli „Student“:

- Možnost zobrazit svoji klasifikaci
- Možnost zobrazit rozvrh hodin
- Možnost zobrazit svoji nepřítomnost na hodinách (docházku)

Požadavky pro roli „Učitel“:

- Přidání nové zkoušky z předmětu, který učitel vyučuje
- Zadání známek, nebo jejich úprava, z dříve založených zkoušek
- Zadání docházky studentů

Požadavky pro roli „Administrátor“:

- Manipulace s uživateli – registrace, smazání
- Přidání tříd a předmětů
- Správa rozvrhů hodin pro třídy

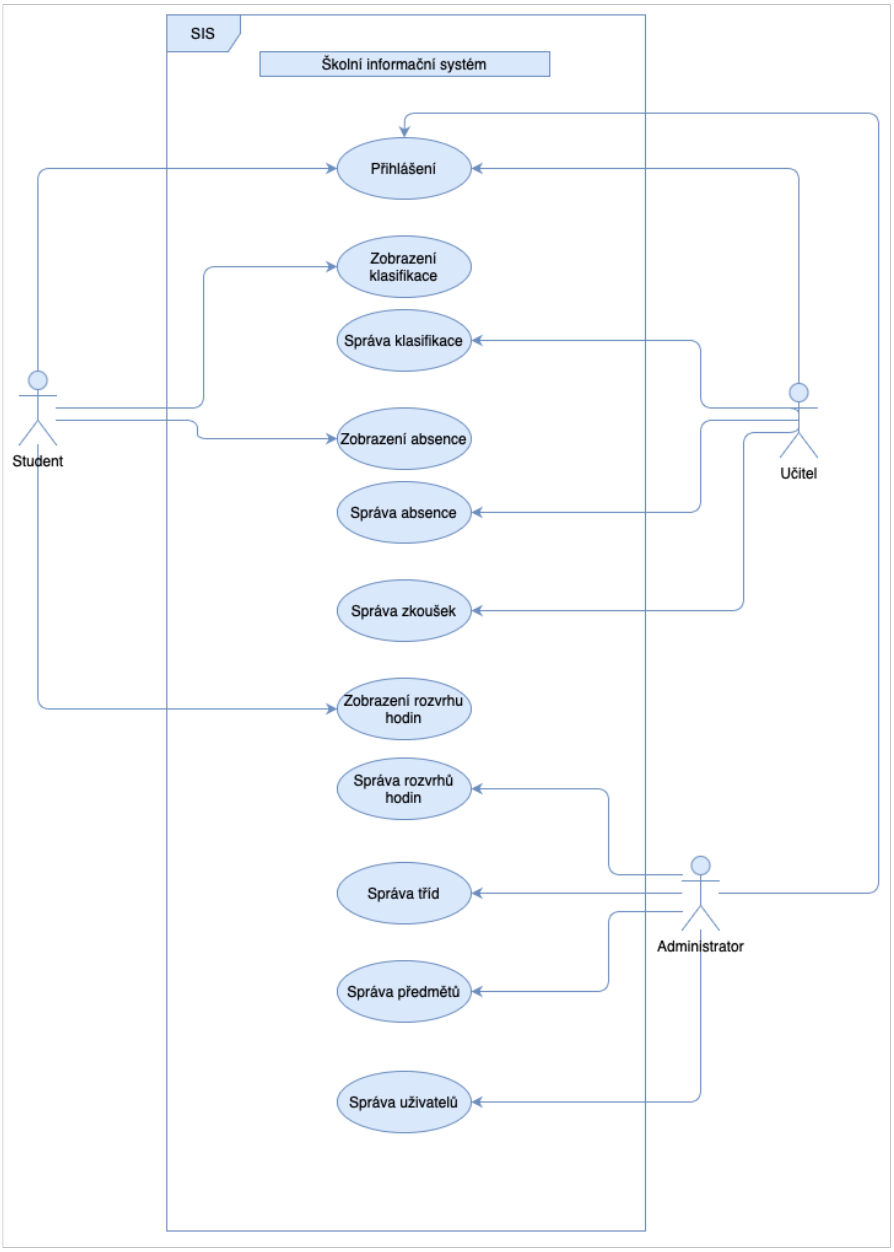
4.2 Analýza IS

Na základě stanovených požadavků v předchozí kapitole byla provedena analýza.

4.2.1 Use Case diagram

V diagramu jsou uvedeny tři aktéři: Administrátor, Student a Učitel. Následující diagram definuje jejich činnosti.

Obrázek 14 - Use Case diagram IS



Následující tabulky obsahují ukázku několika hlavních scénářů, pro lepší pochopení konkrétní situace.

4.2.2 Use Case scénáře

1 - UC1: Přihlášení uživatele

Krátký popis	Přihlášení uživatele do systému
Aktéři	Administrátor, Student, Učitel
Podmínky pro spuštění	Uživatel se chce přihlásit
Základní scénář	1) Systém načte přihlašovací formulář 2) Uživatel vyplní své přihlašovací údaje 3) Uživatel klikne na tlačítko „Přihlásit“ 4) Systém načte uživatelské rozhraní podle typu uživatele
Alternativní scénář	4.1) Systém zobrazí hlášku o neúspěšném přihlášení, uživatel zadal špatné přihlašovací údaje
Podmínky pro dokončení	Uživatel je úspěšně přihlášen

2 - UC2: Registrace nového uživatele

Krátký popis	Registrace nového uživatele (studenta, učitele, administrátora)
Aktéři	Administrátor
Podmínky pro spuštění	Administrátor chce zaregistrovat nového uživatele
Základní scénář	1) Uživatel klikne v menu na „Přidat uživatele“ 2) Systém načte menu pro přidání nového uživatele 3) Uživatel vybere roli nového uživatele 4) Uživatel vyplní všechna ostatní pole potřebná pro založení uživatele 5) Uživatel klikne na tlačítko „Potvrdit“ 6) Systém zobrazí hlášku o úspěšném přidání uživatele
Alternativní scénář	6.1) Systém zobrazí hlášku o neúspěšné registraci, emailová adresa již existuje v databázi 6.2) Systém zobrazí hlášku o neúspěšné registraci, uživatelské jméno již existuje v databázi 6.3) Systém zobrazí hlášku o neúspěšné registraci, telefonní číslo již existuje v databázi 6.4) Systém zobrazí hlášku o neúspěšné registraci, nebyla vyplněna všechna potřebná pole
Podmínky pro dokončení	Uživatel je úspěšně registrován.

3 - UC3: Přidání předmětu

Krátký popis	Přidání předmětu
Aktéři	Administrátor
Podmínky pro spuštění	Administrátor chce přidat předmět
Základní scénář	1) Uživatel klikne v menu na „Přidání předmětu“ 2) Systém načte menu pro přidání nového předmětu 3) Uživatel vybere ze seznamu učitele pro nový předmět 4) Uživatel vyplní všechna ostatní pole 5) Uživatel klikne na tlačítko „Potvrdit“ 6) Systém zobrazí hlášku o úspěšném přidání předmětu
Alternativní scénář	3.1) Systém zobrazí hlášku o prázdné databázi učitelů 6.1) Systém zobrazí hlášku o špatně vyplněných údajích
Podmínky pro dokončení	Předmět úspěšně přidán do databáze

4 - UC4: Úprava rozvrhu

Krátký popis	Upravení rozvrhu hodin pro třídu
Aktéři	Administrátor
Podmínky pro spuštění	Administrátor chce upravit rozvrh třídy
Základní scénář	1) Administrátor klikne v menu na „Rozvrh hodin“ 2) Systém načte menu pro sestavení rozvrhu hodin 3) Administrátor vybere třídu 4) Systém načte rozvrh hodin třídy 5) Administrátor klikne na hodinu, kterou chce upravit 6) Administrátor vybere předmět, učitele a místnost 7) Administrátor klikne na tlačítko „Potvrdit“ 8) Systém zobrazí hlášku o úspěšném přidání/upravení hodiny
Alternativní scénář	8.1) Systém zobrazí hlášku o neúspěšném přidání/upravení hodiny, uživatel nevyplnil všechna pole 8.2) Systém zobrazí hlášku o nepřítomnosti tříd v databázi 8.3) Systém zobrazí hlášku o nepřítomnosti předmětů v databázi
Podmínky pro dokončení	Rozvrh hodin úspěšně upraven

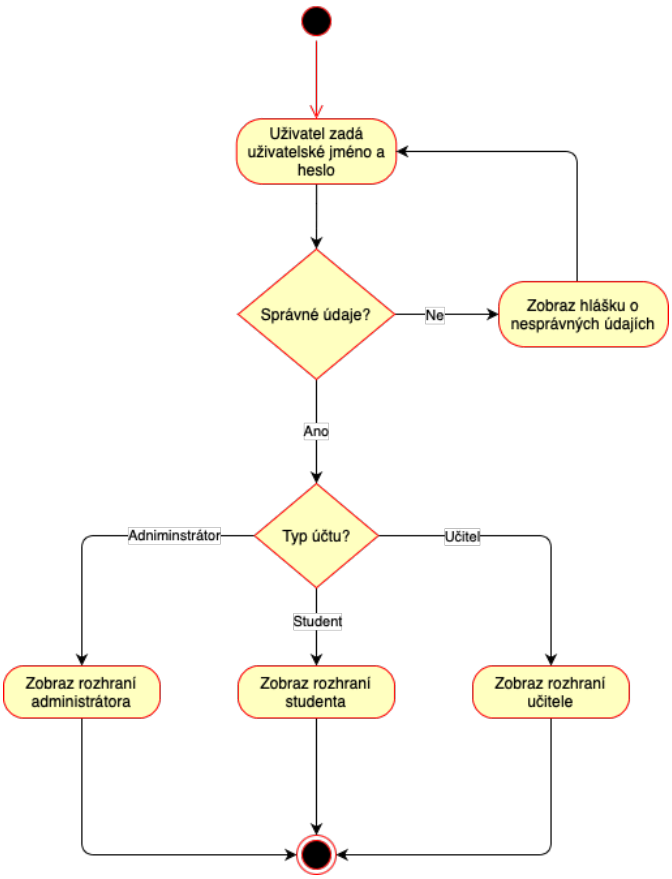
5 - UC5: Přidání docházky

Krátký popis	Přidání docházky
Aktéři	Učitel
Podmínky pro spuštění	Učitel chce přidat docházku k předmětu
Základní scénář	1) Učitel klikne v menu na „Docházka“ 2) Systém načte menu pro přidání docházky 3) Systém načte třídy v databázi 4) Učitel vybere třídu 5) Systém načte předměty třídy, které učí přihlášený uživatel 6) Učitel vybere předmět a vybere týden z kalendáře 7) Systém načte všechny hodiny předmětu v uvedeném týdnu 8) Učitel vybere hodinu 9) Systém načte studenty a jejich docházku k vybraným parametrům 10) Učitel zadá docházku pro studenty 11) Učitel klikne na tlačítko „Potvrdit“ 12) Systém zobrazí hlášku o úspěšném přidání docházky
Alternativní scénář	4.1) Systém nenajde žádnou třídu v databázi 6.1) Systém nenajde žádný z předmětů třídy, které učí přihlášený učitel 10.1) Systém nenajde žádné studenty pro uvedenou třídu 10.2) Systém zobrazí hlášku o neúspěšném přidání docházky, učitel nevyplnil docházku žádného studenta
Podmínky pro dokončení	Docházka přidána do databáze

4.2.3 Diagram aktivit

Následující diagram aktivit zobrazuje proces přihlášení uživatele, kdy se nejdřív ověří jeho přihlašovací údaje a typ účtu a na základě toho se poté zobrazí dané uživatelské rozhraní. V případě zadání nesprávných přihlašovacích údajů se o této skutečnosti zobrazí hláška uživateli.

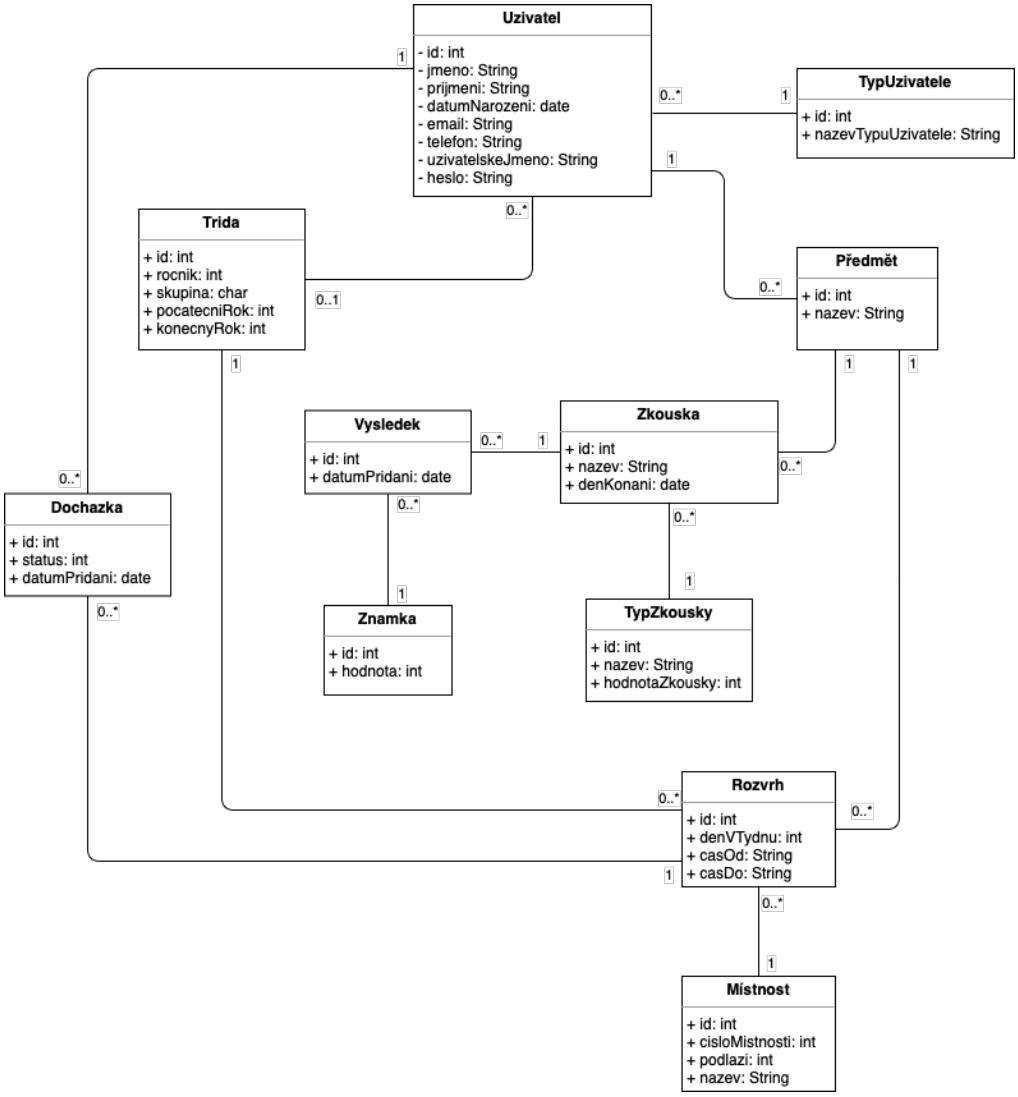
Obrázek 15 - Diagram aktivity - přihlášení



4.2.4 Diagram tříd

Diagram tříd pro zde navrhovaný informační systém obsahuje celkem jedenáct tříd. Hlavní třídou je třída *Uživatel*, ve které se uchovávají informace o registrovaných uživatelů. Každý uživatel musí mít specifikovaný jeho typ, což nám ukazuje napojení jeho třídy na třídu *TypUzivatele*. Dále se zde nachází třída *Předmět*, která je napojena na třídu *Uživatel*, *Zkouška* a *Rozvrh*. Každý předmět totiž musí mít nějakého uživatele (v tomto případě učitele), nemusí mít ovšem zkoušku a rozvrh, naopak zkouška a rozvrh musí nějaký předmět mít. Ze zkoušky vede vztah na třídu *Výsledek*, která v sobě uchovává výsledky studentů. Každý výsledek musí mít nějakého uživatele (studenta) a nějakou známku, což je vidět z napojení třídy na třídu *Score*. Dále je v diagramu třída *Třída*, která obsahuje třídy ve škole a která shrnuje studenty do jedné skupinky. Třída má potom svůj rozvrh, což je vidět z napojení na třídu *Rozvrh*, který musí mít dále napojení na *Předmět*. A pro studenty je důležitá třída *Docházka*, která je napojená na třídu *Rozvrh*. A na závěr třída *TypZkoušky* a *Místnost*. Jde o dodatečné informace k rozvrhu a zkouškám.

Obrázek 16 - Diagram tříd



4.3 Návrh IS

Na základě analýzy z předchozí kapitoly byl proveden návrh informačního systému.

4.3.1 Datový slovník

User – uživatel systému. Kromě údajů o uživateli obsahuje tabulka ještě cizí klíč na *UserType*, čímž se určí role uživatele a dále obsahuje cizí klíč na Grade tabulku který určí, do jaké třídy uživatel patří. Pouze uživatel s rolí *Student* může mít přidělenou třídu.

Tabulka 6 - Tabulka User

Atribut	Typ	Délka	Klíč	Popis
userId	int	255	primární	ID uživatele
firstName	varchar	25		Křestní jméno
lastName	varchar	25		Příjmení
dayOfBirth	date			Datum narození
gender	char	1		Pohlaví
phone	varchar	13		Telefonní číslo
email	varchar	50		Email
username	varchar	20		Uživatelské jméno
password	varchar	255		Heslo (hash)
isActive	tinyint	1		Stav uživatele
userTypeId	int	255	cizí	ID uživatelského typu
gradeId	int	255	cizí	ID třídy

UserType – Třída obsahuje typy uživatelských účtů. Můžou jimi být: Student, Učitel a Administrator.

Tabulka 7 - Tabulka UserType

Atribut	Typ	Délka	Klíč	Popis
userTypeId	int	255	primární	ID typu uživatele
typeDesc	varchar	20		Popis typu uživatele
hasGrade	tinyint	1		Určuje, zda má mít uživatel třídu

Grade – Zde jsou uchovávány informace o třídách ve škole.

Tabulka 8 - Tabulka Grade

Atribut	Typ	Délka	Klíč	Popis
gradeId	int	255	primární	ID třídy
year	int	1		Ročník
startYear	int	4		Rok zahájení
endYear	int	4		Rok ukončení
maxPeople	int	5		Maximální počet studentů

Subject – Informace o vedených předmětech ve škole. Tabulka obsahuje cizí klíč na tabulku *User*, jelikož každý předmět musí mít svého učitele.

Tabulka 9 - Tabulka Subject

Atribut	Typ	Délka	Klíč	Popis
subjectId	int	255	primární	ID předmětu
name	varchar	20		Název předmětu
userId	int	255	cizí	ID uživatele (učitele)

Exam – Tato třída obsahuje informace o zkouškách z předmětů. Tabulka je napojena na tabulky *Subject* a *Result*

Tabulka 10 - Tabulka Exam

Atribut	Typ	Délka	Klíč	Popis
examId	int	255	primární	ID zkoušky
name	varchar	50		Název zkoušky
dayOfExecution	date			Datum konání zkoušky
examTypeId	int	255	cizí	ID typu zkoušky
subjectId	int	255	cizí	ID předmětu

ExamType – Tato třída obsahuje informace o typech zkoušek.

Tabulka 11 - Tabulka ExamType

Atribut	Typ	Délka	Klíč	Popis
examTypeId	int	255	primární	ID typu zkoušky
name	varchar	20		Název zkoušky
value	tinyint	1		Váha zkoušky

Result – Tato třída obsahuje informace o udělených známkách ze zkoušek. Tabulka obsahuje cizí klíče na tabulky *Exam* a *User*, jelikož každý výsledek musí patřit nějakému studentovi a dále musí být spojen s nějakou zkouškou.

Tabulka 12 - Tabulka Result

Atribut	Typ	Délka	Klíč	Popis
resultId	int	255	primární	ID známky
dayOfInsert	date			Datum přidání známky
scoreId	int	255	cizí	ID známky
examId	int	255	cizí	ID zkoušky
userId	int	255	cizí	ID uživatele (studenta)

Score – Obsahuje pouze hodnoty známek pro tabulku Result

Tabulka 13 - Tabulka Score

Atribut	Typ	Délka	Klíč	Popis
scoreId	int	255	primární	ID hodnoty známky
value	varchar	10		Hodnota známky

Schedule – Zde jsou informace o rozvrhu třídy. Zde jsou cizí klíče tři a to na tabulky *Grade*, *Subject* a *Classroom*.

Tabulka 14 - Tabulka Schedule

Atribut	Typ	Délka	Klíč	Popis
scheduleId	int	255	primární	ID rozvrhu
day	int	1		Den v týdnu
timeFrom	time			Čas od
timeTo	time			Čas do
gradeId	int	255	cizí	ID třídy
subjectId	int	255	cizí	ID předmětu
classroomId	int	255	cizí	ID místnosti

Classroom – Tato třída obsahuje pouze data o místnostech ve škole.

Tabulka 15 - Tabulka Classroom

Atribut	Typ	Délka	Klíč	Popis
classroomId	int	255	primární	ID místnosti
name	varchar	20		Název místnosti
roomNumber	int	5		Číslo místnosti
floor	int	5		Podlaží

Attendance – Tato třída obsahuje informace o docházce studenta. V tabulce se nachází cizí klíče na tabulky *Schedule* a *User*.

Tabulka 16 - Tabulka Attendance

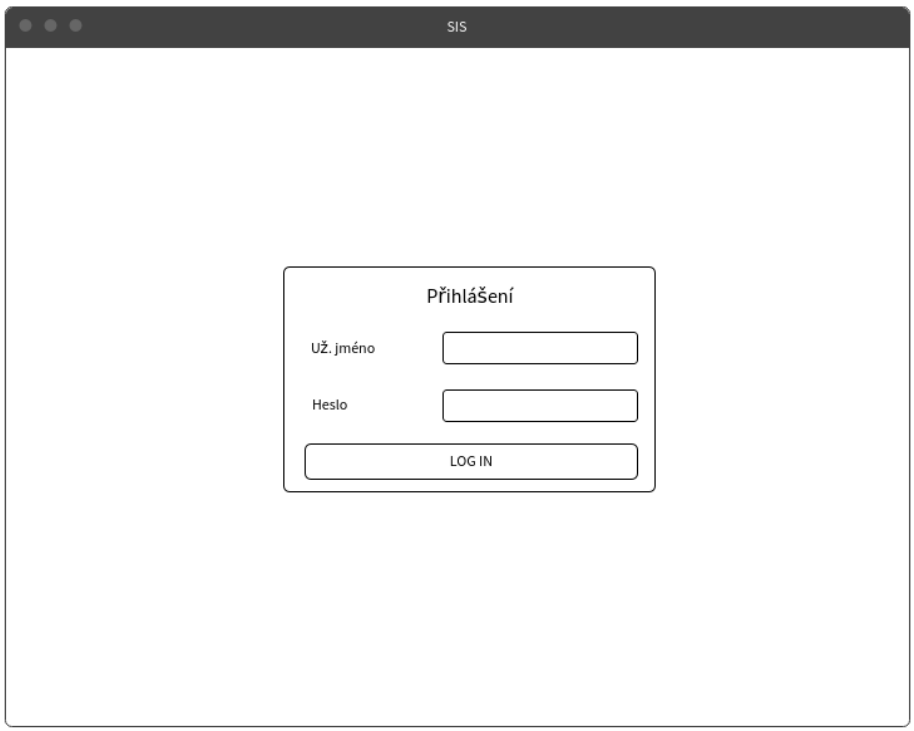
Atribut	Typ	Délka	Klíč	Popis
attendanceId	int	255	primární	ID docházky
status	tinyint	1		Přítomen / Nepřítomen
date	date			Datum zadání docházky
scheduleId	int	255	cizí	ID hodiny v rozvrhu hodin
userId	int	255	cizí	ID uživatele (studenta)

4.3.2 Wireframe

Wireframe je ukázka struktury a vzhledu webové stránky. Můžeme v něm najít přibližné místo, kde se bude jaký element nacházet a kde by mohl konečný uživatel najít potřebné informace. Jako ukázku pro tento informační systém byly vytvořeny tyto wireframy:

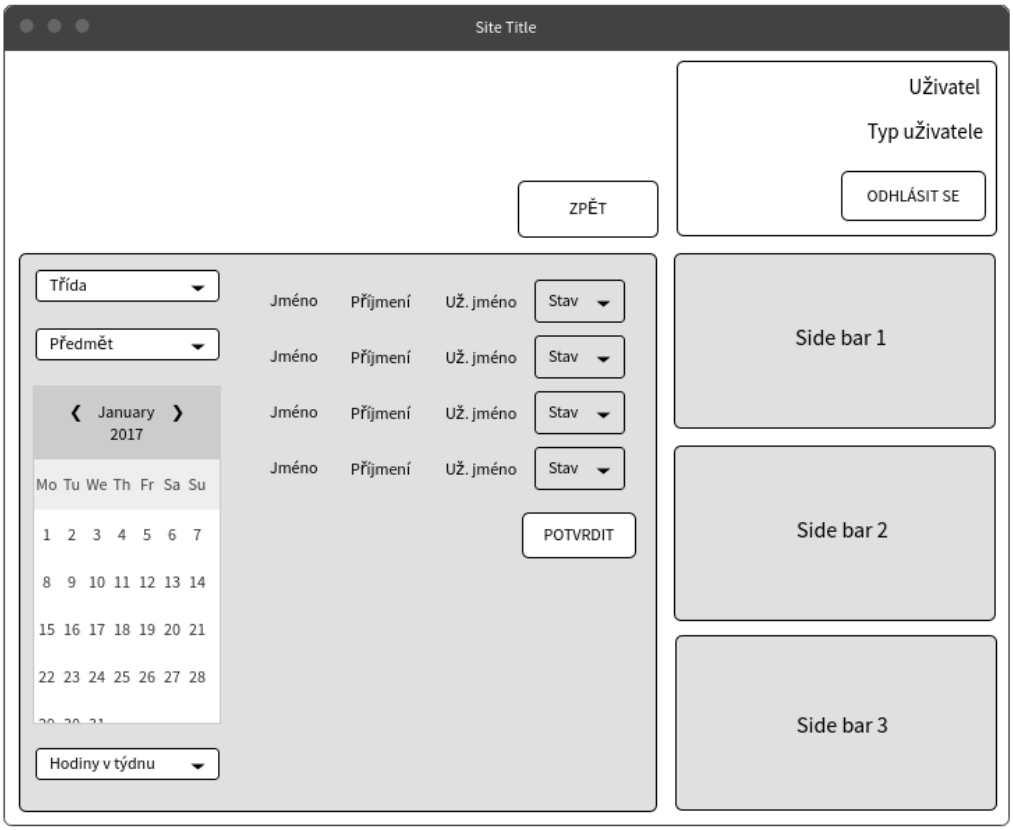
- Wireframe přihlašovací obrazovky při příchodu do aplikace
- Wireframe menu uživatele
- Wireframe formuláře pro přidání docházky do databáze
- Wireframe formuláře pro přidání nového uživatele

Obrázek 17 - Wireframe 1: přihlašovací obrazovka



Obrázek 18 - Wireframe 2: uživatelské menu

Obrázek 19 - Wireframe 3: přidání docházky



Obrázek 20 - Wireframe 4: registrace uživatele

4.4 Implementace

Struktura webové stránky byla zpracována pomocí značkovacího jazyka HTML, pro vzhled stránky byly použity kaskádové styly CSS. Front-end aplikace je tvořen jazykem JavaScript a knihovnou React. Pro ukládání dat byla použita databáze MySQL a pro komunikací s ní (back-end) byl použit jazyk PHP.

4.4.1 Vytvoření databáze

Podle návrhu tabulek byla vytvořena MySQL databáze.

Obrázek 21 - Tabulky v MySQL databázi

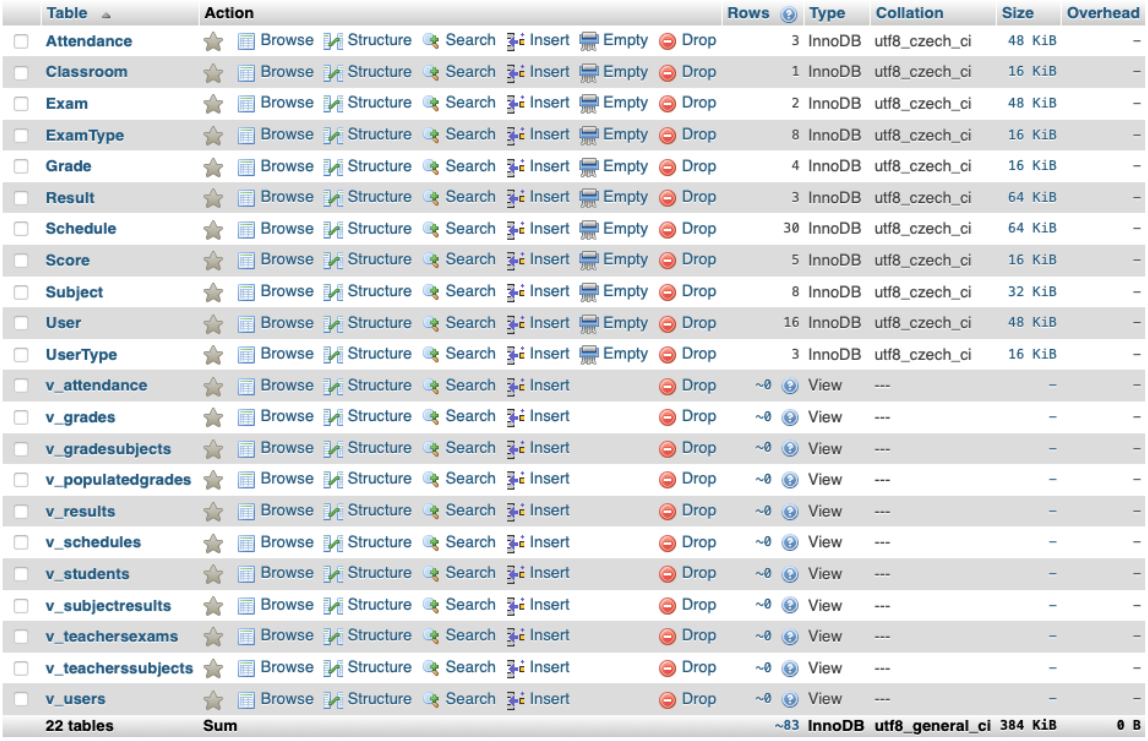


Table	Action	Rows	Type	Collation	Size	Overhead
Attendance	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_czech_ci	48 KiB	-
Classroom	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_czech_ci	16 KiB	-
Exam	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_czech_ci	48 KiB	-
ExamType	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8_czech_ci	16 KiB	-
Grade	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_czech_ci	16 KiB	-
Result	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_czech_ci	64 KiB	-
Schedule	Browse Structure Search Insert Empty Drop	30	InnoDB	utf8_czech_ci	64 KiB	-
Score	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8_czech_ci	16 KiB	-
Subject	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8_czech_ci	32 KiB	-
User	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8_czech_ci	48 KiB	-
UserType	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_czech_ci	16 KiB	-
v_attendance	Browse Structure Search Insert Drop	~0	View	---	-	-
v_grades	Browse Structure Search Insert Drop	~0	View	---	-	-
v_gradesubjects	Browse Structure Search Insert Drop	~0	View	---	-	-
v_populatedgrades	Browse Structure Search Insert Drop	~0	View	---	-	-
v_results	Browse Structure Search Insert Drop	~0	View	---	-	-
v_schedules	Browse Structure Search Insert Drop	~0	View	---	-	-
v_students	Browse Structure Search Insert Drop	~0	View	---	-	-
v_subjectresults	Browse Structure Search Insert Drop	~0	View	---	-	-
v_teachersexams	Browse Structure Search Insert Drop	~0	View	---	-	-
v_teacherssubjects	Browse Structure Search Insert Drop	~0	View	---	-	-
v_users	Browse Structure Search Insert Drop	~0	View	---	-	-
22 tables	Sum	~83	InnoDB	utf8_general_ci	384 KiB	0 B

Jako příklad je zde uveden SQL příkaz pro vytvoření tabulky User.

```
CREATE TABLE `User` (
  `userId` int(255) NOT NULL,
  `firstName` varchar(25) COLLATE utf8_czech_ci NOT NULL,
  `lastName` varchar(25) COLLATE utf8_czech_ci NOT NULL,
  `dayOfBirth` date NOT NULL,
  `gender` char(1) COLLATE utf8_czech_ci NOT NULL,
  `phone` varchar(13) COLLATE utf8_czech_ci NOT NULL,
  `email` varchar(50) COLLATE utf8_czech_ci NOT NULL,
  `username` varchar(20) COLLATE utf8_czech_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_czech_ci NOT NULL,
  `userTypeId` int(255) NOT NULL,
  `gradeId` int(255) DEFAULT NULL,
  isActive` tinyint(1) NOT NULL DEFAULT '1'
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

Následně byl do tabulky přidán primární klíč a poté nastavení tohoto klíče jako AUTOINCREMENT, aby se při každém novém záznamu do tabulky klíč generoval sám.

```
ALTER TABLE `User`
  ADD PRIMARY KEY (`userId`), ADD KEY `gradeId` (`gradeId`),
  ADD KEY `userTypeId` (`userTypeId`);
ALTER TABLE `User` MODIFY `userId` int(255) NOT NULL AUTO_INCREMENT;
```

Pro lepší manipulaci s daty bylo vytvořeno i několik VIEW tabulek. Následující SQL kód ukazuje vytvoření v_teacherssubjects VIEW, které zobrazuje učitelovi předměty.

```
CREATE VIEW v_teacherssubjects AS
SELECT s.subjectId AS 'subjectId',
    s.name AS 'subjectName',
    u.userId AS 'teacherId',
    u.firstName AS 'teacherFirstName',
    u.lastName AS 'teacherLastName',
    u.username AS 'teacherUsername'
FROM Subject s, User u
WHERE s.userId = u.userId
ORDER BY u.lastName ASC,
     u.firstName ASC,
     s.name ASC;
```

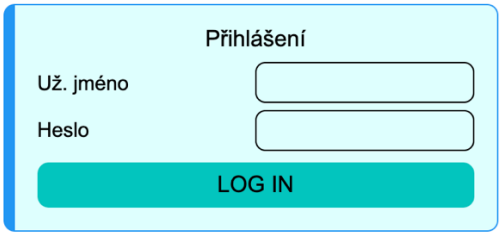
Následující kód ukazuje vytvoření VIEW v_gradesubjects, které zobrazuje předměty, které se učí ve třídách.

```
CREATE VIEW v_gradeSubjects AS
SELECT DISTINCT sch.subjectId AS 'subjectId',
            su.name AS 'subjectName',
            su.userId AS 'teacherId',
            sch.gradeId AS 'gradeId'
FROM Schedule sch, Subject su
WHERE sch.subjectId = su.subjectId
```

4.4.2 Přihlášení uživatele

Po příchodu na stránku se uživateli zobrazí formulář pro přihlášení.

Obrázek 22 - Přihlašovací formulář



Po zadání přihlašovacích údajů a kliknutí na tlačítko „LOG IN“ se spustí React komponenta AXIOS, která zavolá script v soubor logIn.php, který provede skript vůči databázi a zjistí, zda se v ní uživatel nachází a vrátí jeho typ.

Níže uvedený kód zobrazuje jednotnou funkci pro volání komponenty AXIOS. Jako parametry volání funkce jsou *jsonData*, která obsahují data, která chceme předat php scriptu. Dalším parametrem je *phpFile*, který specifikuje soubor, ve kterém se daný script nachází.

```
export async function axiosCall(jsonData, phpFile) {
  let return_value = null;

  await axios({
    method: 'POST',
    url: apiPath + 'api/' + phpFile + '.php',
    headers: { 'content-type': 'application/json' },
    data: JSON.stringify(jsonData)
  }).then(result => {
    return_value = result.data;
  }).catch(error => {
    console.log(jsonData.requestType + ": " + error.message);
  });

  return return_value;
}
```

Zde je uvedená PHP funkce pro přihlášení uživatele. Funkce provede SQL script vůči databázi, kde se snaží dostat data o klientovi se zadanými přihlašovacími údaji. Následně předá data zpět Reactu, který je následně zpracuje.

```
function logIn($mysqli, $username, $password) {
```

```
    $sql = "SELECT u.userId AS 'userId', u.password AS 'password', u.firstName AS 'firstName',
u.lastName AS 'lastName', ut.typeDesc AS 'typeDesc' FROM User u, UserType ut WHERE
u.userTypeId = ut.userTypeId AND u.username = '" . $username . "' AND u.isActive = 1";
    $result = $mysqli->query($sql);

    $userExists = false;
    $correctPassword = false;
    $userType = "";
    $userId = 0;
    $userFullName = "";

    if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();

        $userExists = true;
        $correctPassword = password_verify($password, $row['password']);
        $userType = $row['typeDesc'];
        $userId = $row['userId'];
        $userFullName = $row['firstName'] . " " . $row['lastName'];
    }

    return [ 'userExists' => $userExists, 'correctPassword' => $correctPassword, 'userType' =>
$userType, 'userId' => $userId, 'userFullName' => $userFullName ];
}
```

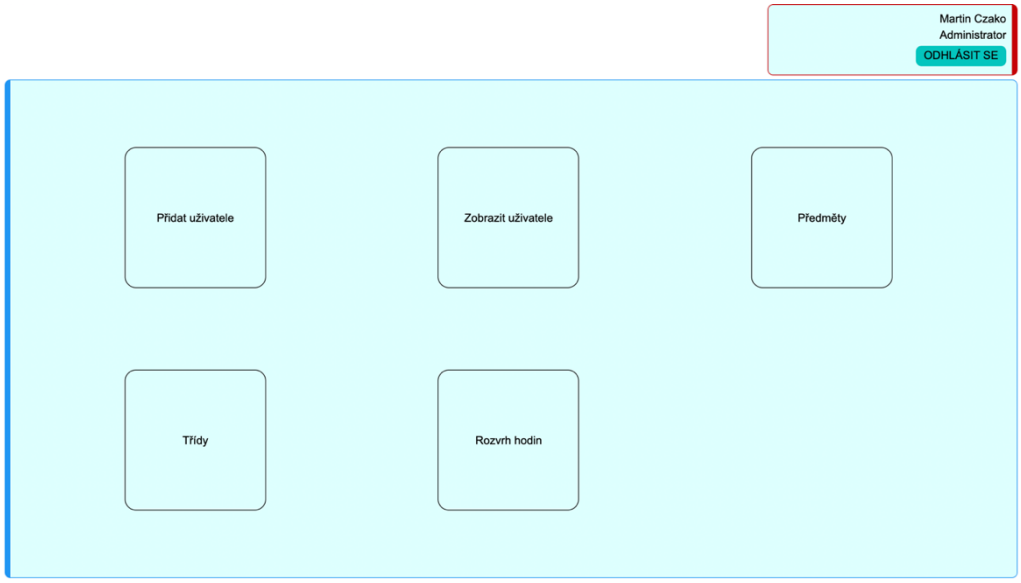
4.4.3 Funkce administrátora

Administrátor má na starost největší část databáze, a proto má také nejvíc funkcí.

4.4.3.1 Menu administrátora

V administrátorském menu je pět tlačítek přepínající zobrazení menu na konkrétní funkce. Po stisknutí jakéhokoliv tlačítka všechna tlačítka zmizí a zobrazí se vybraný menu funkce.

Obrázek 23 - Menu administrátora

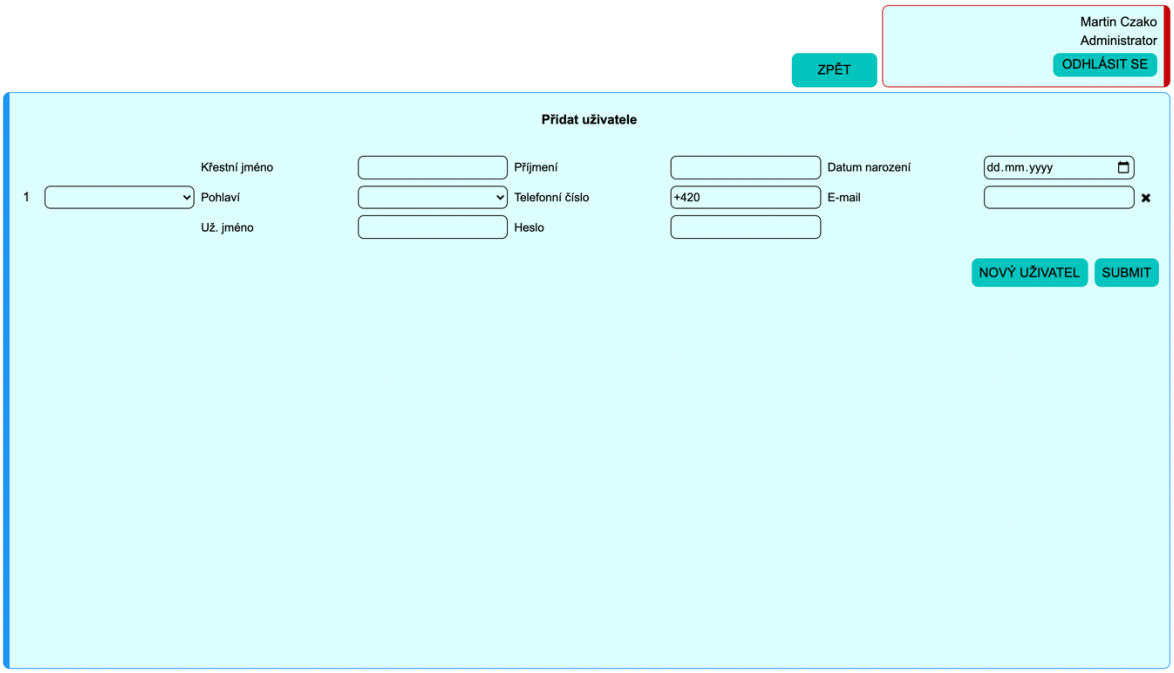


Administrátor má k dispozici funkce pro správu uživatelů, tříd, předmětů a také pro sestavení rozvrhu hodin pro třídy. Učitel může přidávat zkoušky, klasifikaci za dané zkoušky a upravovat docházku k hodinám v týdnu. Student žádné možnosti úpravy nemá a může si tedy data o jeho studiu pouze zobrazit.

4.4.3.2 Přidání nového uživatele

Přidání uživatele je ta základní funkce, bez kterého by celý systém neměl smysl. Pro zadání nového uživatele je potřeba zadat jeho jméno a příjmení, datum narození, pohlaví, telefonní číslo a email, uživatelské jméno a heslo. Dále se zvolí role nového uživatele. Pokud má být role uživatele Student, tak se zobrazí i další políčko Třída, které je také nutné vyplnit.

Obrázek 24 - Menu registrace uživatele



Uživatelů jde zadat více najednou. Po stisknutí tlačítka „Nový uživatel“ se pod současného uživatele přidá formulář pro dalšího uživatele. Počet formulářů není nijak omezen.

Po stisknutí tlačítka „Submit“ se zkontrolují všichni nový uživatelé a pokud existuje aspoň jeden se všemi správně vypsanými hodnotami, tak se zavolá php script pro zadání uživatele do databáze (níže uveden).

```
function addUser($mysqli, $first_name, $last_name, $dayOfBirth, $gender, $phone, $email, $username, $password, $gradeId, $userTypeId) {
    $sqlCheck = "SELECT userId FROM User WHERE phone = $phone OR email = $email OR username = $username";
    $resultCheck = $mysqli->query($sqlCheck);

    $sqlOk = false;

    if ($resultCheck->num_rows == 0) {
        $hashed = password_hash($password, PASSWORD_DEFAULT);
        $hashed = trim($hashed);
        $sql = "INSERT INTO User (firstName, lastName, dayOfBirth, gender, phone, email, username, password, gradeId, userTypeId)
                VALUES ('" . $first_name . "', '" . $last_name . "', '" . $dayOfBirth . "', '" . $gender . "', '" . $phone . "', '" . $email . "', '" . $username . "', '" . $hashed . "', '" . (int)$gradeId . "', '" . (int)$userTypeId . "')";
        $result = $mysqli->query($sql);

        if ($result) $sqlOk = true;
    } else {
```

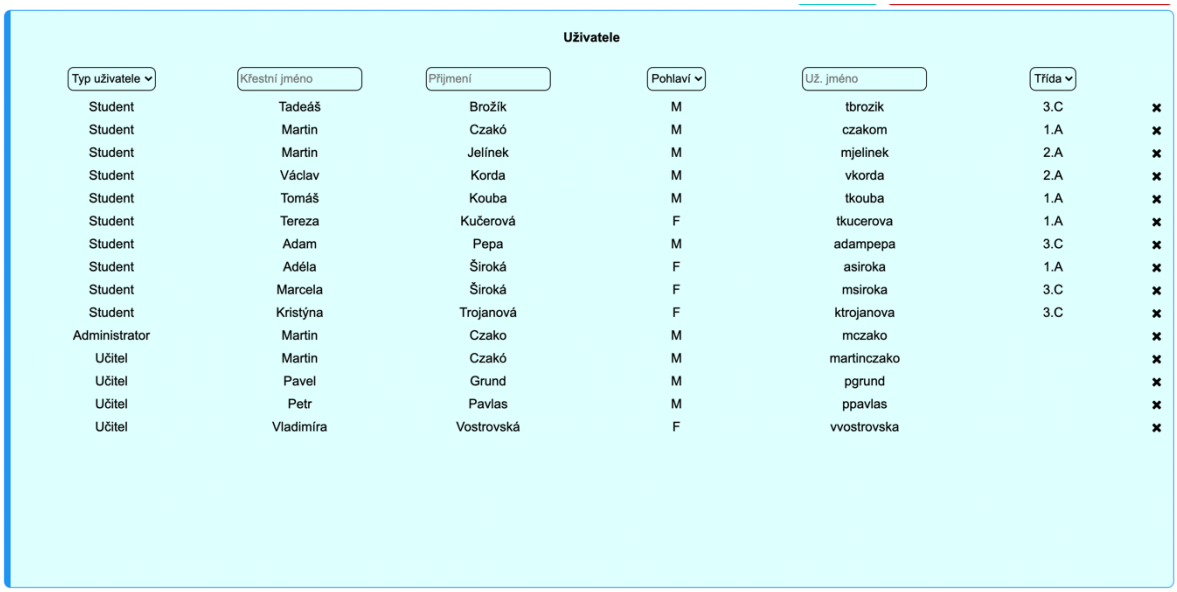
```
        $sqlOk = true;
    }

    return $sqlOk;
}
```

Nejdříve script zkontroluje, zda se již v databázi nenachází uživatel s uvedeným uživatelským jménem, emailem nebo telefonním číslem. Pokud se uživatel v databázi nenachází, přestoupí skript k jeho přidání. Heslo se do databáze přidává zašifrované pomocí hash. K tomu byla použita PHP funkce *password_hash()*.

4.4.3.3 Odstranění uživatele

Další funkcí určenou pouze pro administrátora je smazání uživatele.

Obrázek 25 - Meni zobrazení uživatelů

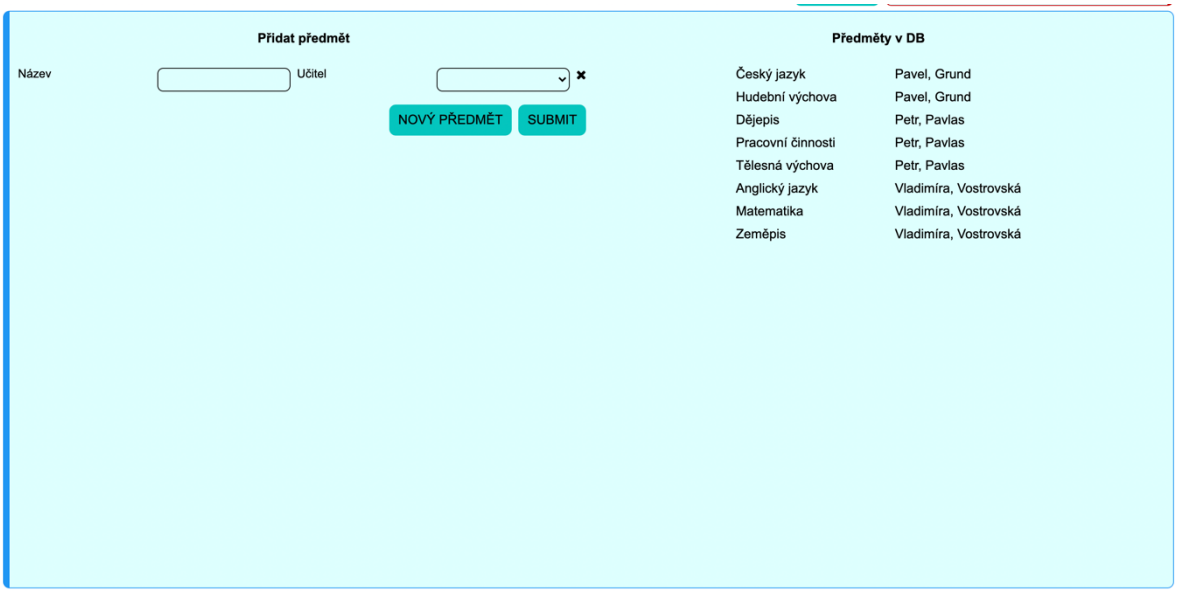


Uživatelé si je možno vyfiltrovat podle jeho typu, jména, příjmení, pohlaví, už. jména a třídy. Následně po stisknutí ikonky křížku na konci řádku se zavolá php funkce, která přes sql nastaví uživateli jeho atribut *isActive* na nula a tím se stane pro systém neaktivní. Smazání uživatele jaké takové v tomto systému neprobíhá.

4.4.3.4 Přidání předmětu

V sekci předmětů může administrátor přidat nový předmět. Zároveň je zde i výpisy předmětů aktuálně v databázi.

Obrázek 26 - Menu přidání předmětu



Administrátor musí pro přidání zadat pouze název předmětu a vybrat nějakého učitele. Pokud není v databázi veden žádný učitel, zobrazí se o této skutečnosti hláška.

4.4.3.5 Přidání třídy

V sekci tříd administrátor přidává nové třídy do databáze. Je zde také výpis všech tříd v systému.

Obrázek 27 - Menu přidání třídy

Třída musí obsahovat ročník (číslo od 1 do 9) a skupinu (písmeno A až C). Dále su musí specifikovat maximální počet studentů, který může třída mít a nakonec se musí

specifikovat rok, od kterého třída platí. Pokud v systému už existuje třídy, jako ta zadaná administrátor, tak je o tom administrátor obeznámen a systém mu nedovolí třídu přidat.

4.4.3.6 Sestavení rozvrhu hodin

Velmi důležitou součástí je stavba rozvrhu tříd. Bez toho by nebylo možné učitelem zadat žádné známky a docházku pro studenta.

Obrázek 28 - Menu správy rozvrhu hodin

Rozvrh hodin je rozdělen do pěti řádků, podle dní v týdnu. Systém při zvolení třídy načte ze systému její již nastavené hodiny. Každá hodina jde upravovat přes a to tak, že administrátor klikne na ikonku ozubených koleček uvnitř buňky. Buňka se potom změni na možnost úpravy, jak je ukázáno ve druhém řádku. Po zvolení nových hodnot a kliknutí na ikonku vpravo se buď do databáze hodina přidá, nebo se upraví stávající. Ikonka vlevo zavře možnost úpravy.

4.4.3.7 Přidání zkoušky

Učitel musí přidat do systému zkoušku, aby z ní mohl přidat známku studentům.

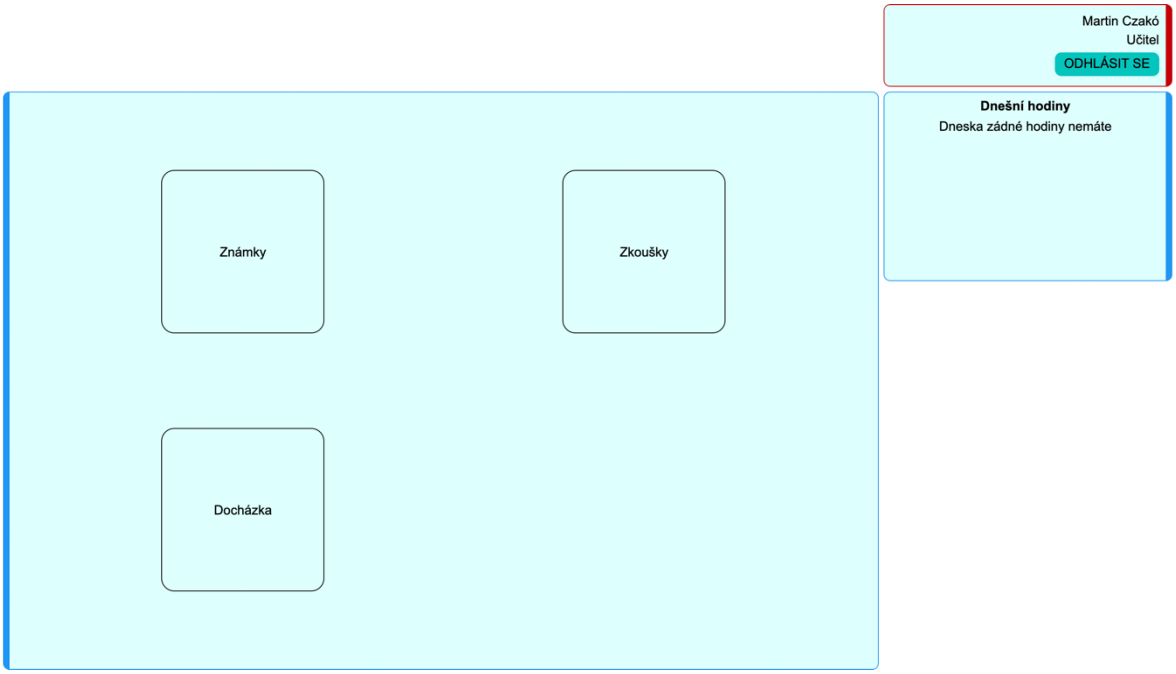
Pro přidání je nutné zadat její název, datum uskuteční, předmět ze kterého zkouška má být a potom typ zkoušky. Typ zkoušky se při načtení formuláře stáhnou z databáze (tabulka *ExamType*). Každý typ zkoušky má nějakou svoji váhu, podle které se následně dělá vážený průměr studentových známek.

Obrázek 29 - Menu přidání zkoušky

4.4.4 Funkce učitele

4.4.4.1 Menu učitele

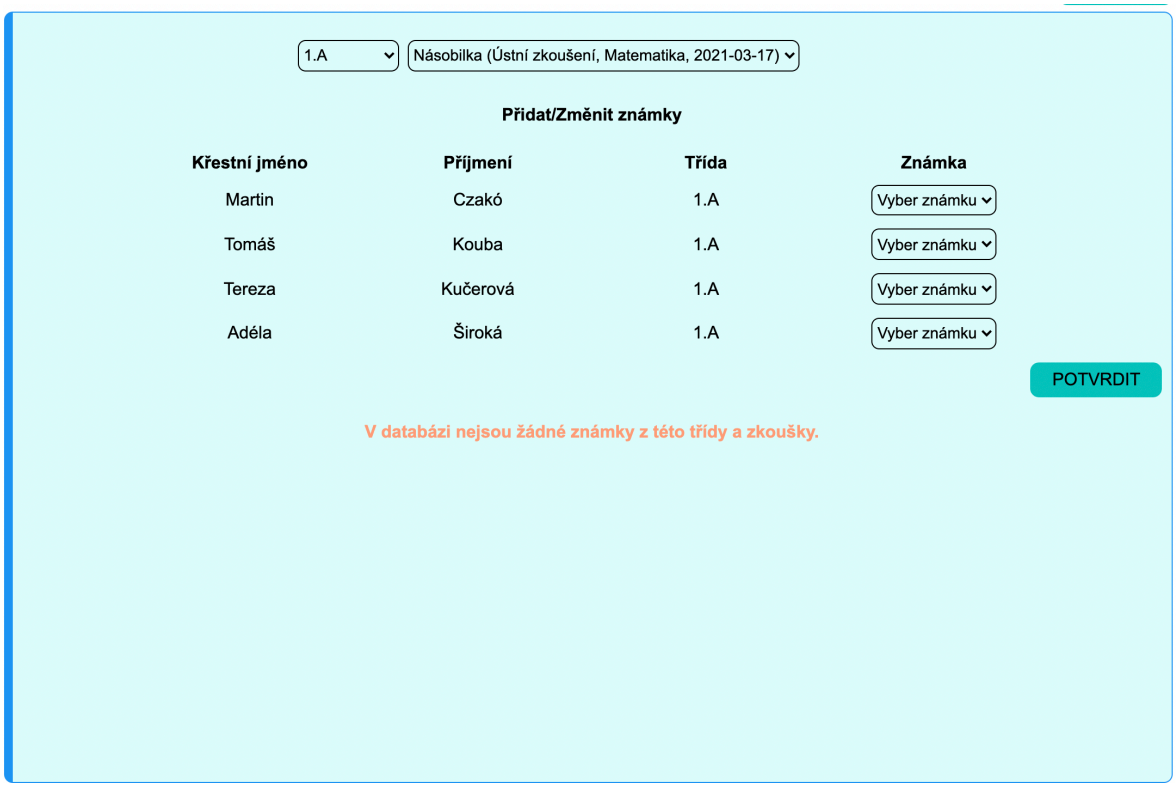
Obrázek 30 - Menu učitele



4.4.4.2 Klasifikace

Tato funkce umožňuje učitelovi zadat klasifikaci k dané zkoušce. Po načtení menu se z databáze stáhnou třídy, které mají nějaké studenty, dále učitelovi zkoušky. Dále si také systém načte typy známek vedené v databázi (tabulky *Score*). Po vybraní třídy a zkoušky se načtou studenti příslušné třídy a také známky, které jim již byly přiděleny. Již přidělené známky se dají v tomto menu upravovat.

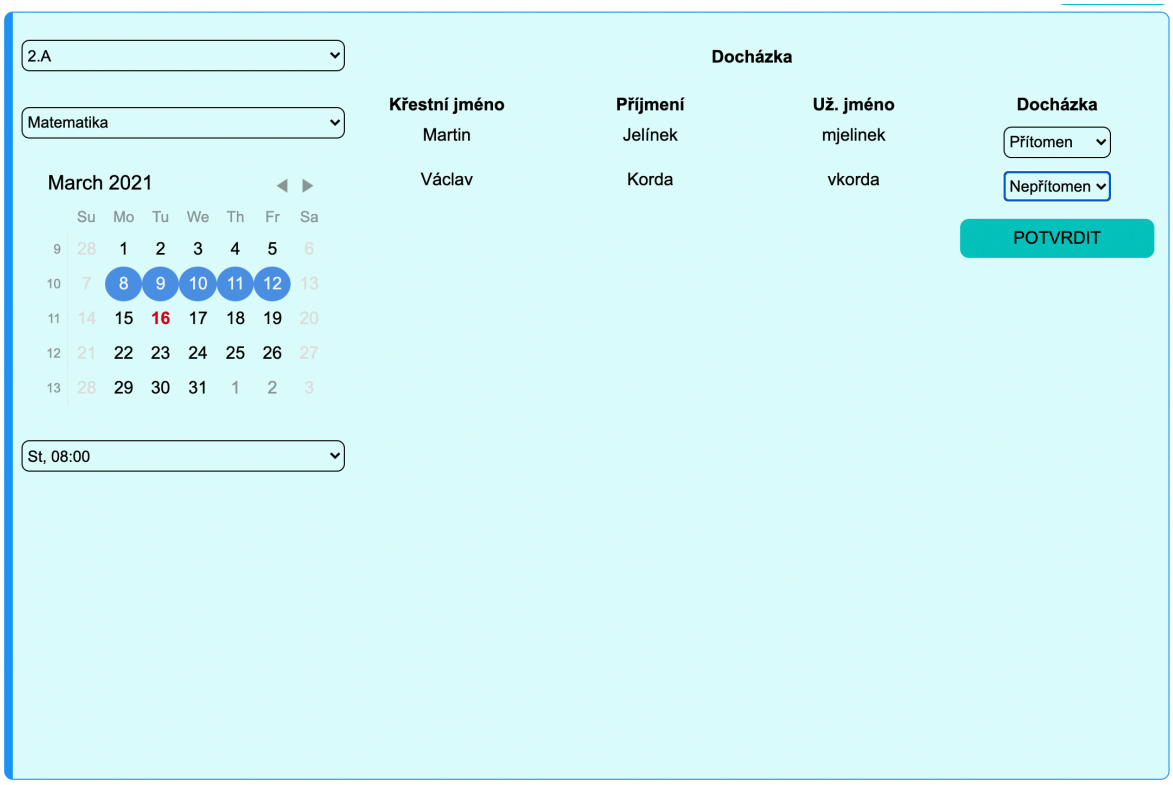
Obrázek 31 - Menu přidání známky



4.4.4.3 Zadání docházky

Další funkcí systému je zadání docházky žáku na učitelově hodině. Po zvolení tlačítka pro docházku se zobrazí příslušné menu. Při jeho načtení se z databáze stáhnou třídy, dále předměty, které učitel učí. Učitel tyto hodnoty zvolí v levém sloupci, poté zvolí kalendářní týden, ve kterém se hodina nachází a následně systém podle načte příslušný rozvrh a dá ho učiteli k vybrání. Po vybraní všech parametrů se zobrazí studenti a už stačí jenom vyplnit onu docházku.

Obrázek 32 - Menu přidání docházky

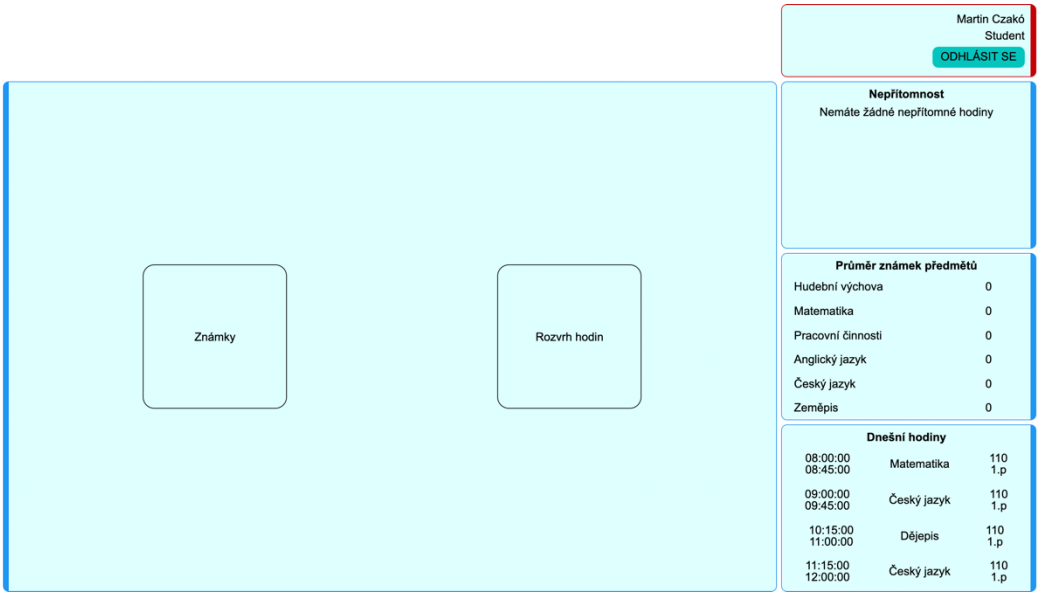


4.4.5 Funkce studenta

Student nemá žádné funkce, které by upravovaly data v databázi, má je možnost pouze zobrazit.

4.4.5.1 Menu studenta

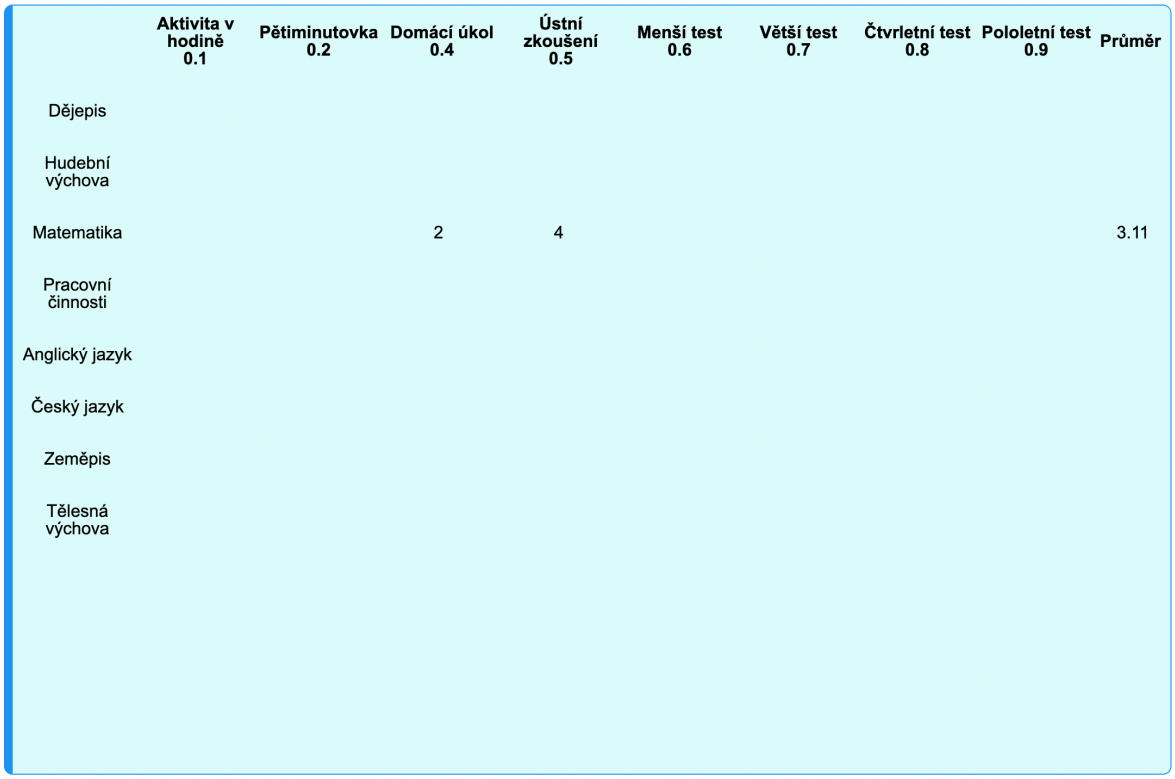
Obrázek 33 - Menu studenta



4.4.5.2 Studentovi známky

Studentovi se po vybraní této funkce zobrazí v menu jakási tabulka, kde je seznam všech zkoušek a předmětů, které se načtou při zobrazení menu z databáze. Na konci seznamu je zobrazen vážený průměr všech známek předmětu.

Obrázek 34 - Menu zobrazení známek

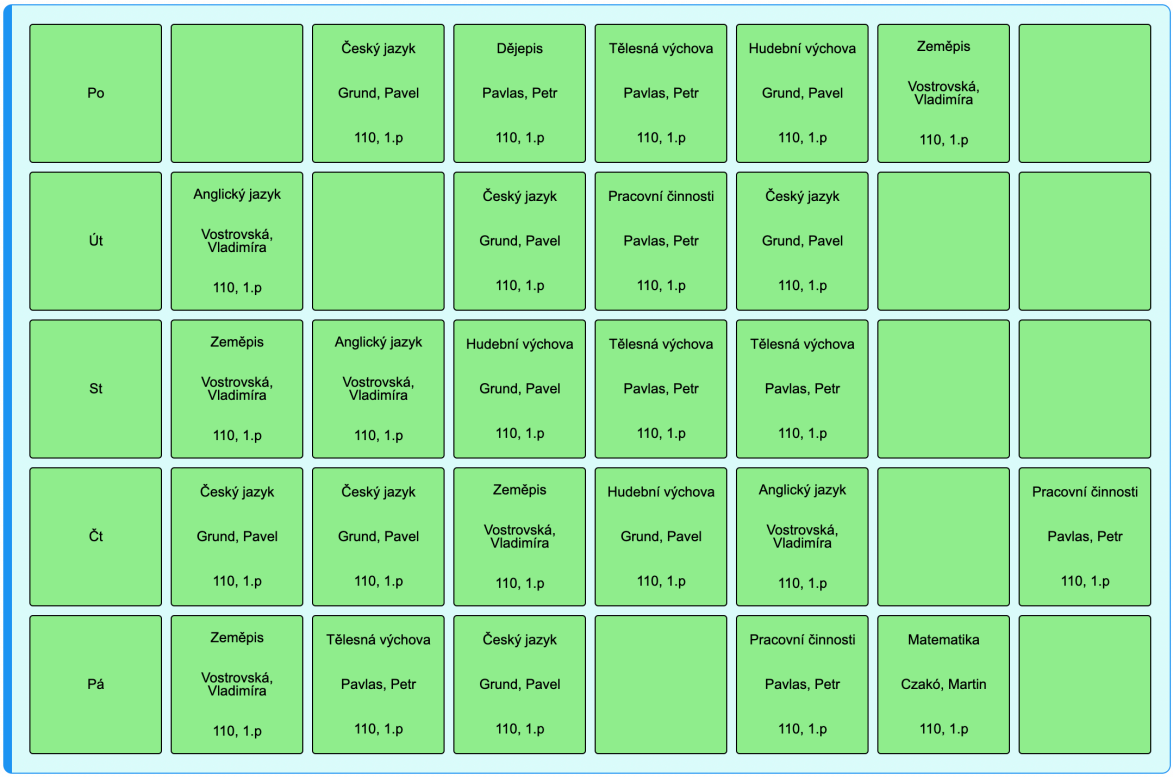


	Aktivita v hodině 0.1	Pětiminutovka 0.2	Domácí úkol 0.4	Ústní zkoušení 0.5	Menší test 0.6	Větší test 0.7	Čtvrletní test 0.8	Pololetní test 0.9	Průměr
Dějepis									
Hudební výchova									
Matematika			2	4					3.11
Pracovní činnosti									
Anglický jazyk									
Český jazyk									
Zeměpis									
Tělesná výchova									

4.4.6 Studentův rozvrh hodin

Druhá a poslední funkce pro studenta je možnost zobrazit si jeho rozvrh hodin. Systém vykreslení je úplně stejný jako ten pro administrátora, ovšem bez možnosti úpravy rozvrhu.

Obrázek 35 - Menu zobrazení rozvrhu hodin



Po		Český jazyk Grund, Pavel 110, 1.p	Dějepis Pavlas, Petr 110, 1.p	Tělesná výchova Pavlas, Petr 110, 1.p	Hudební výchova Grund, Pavel 110, 1.p	Zeměpis Vostrovská, Vladimíra 110, 1.p	
Út	Anglický jazyk Vostrovská, Vladimíra 110, 1.p		Český jazyk Grund, Pavel 110, 1.p	Pracovní činnosti Pavlas, Petr 110, 1.p	Český jazyk Grund, Pavel 110, 1.p		
St	Zeměpis Vostrovská, Vladimíra 110, 1.p	Anglický jazyk Vostrovská, Vladimíra 110, 1.p	Hudební výchova Grund, Pavel 110, 1.p	Tělesná výchova Pavlas, Petr 110, 1.p	Tělesná výchova Pavlas, Petr 110, 1.p		
Čt	Český jazyk Grund, Pavel 110, 1.p	Český jazyk Grund, Pavel 110, 1.p	Zeměpis Vostrovská, Vladimíra 110, 1.p	Hudební výchova Grund, Pavel 110, 1.p	Anglický jazyk Vostrovská, Vladimíra 110, 1.p		Pracovní činnosti Pavlas, Petr 110, 1.p
Pá	Zeměpis Vostrovská, Vladimíra 110, 1.p	Tělesná výchova Pavlas, Petr 110, 1.p	Český jazyk Grund, Pavel 110, 1.p		Pracovní činnosti Pavlas, Petr 110, 1.p	Matematika Czakó, Martin 110, 1.p	

5 Závěr

V teoretické části bakalářské práce byl čtenář seznámen s podstatou informačního systému, k čemu se využívá a jaké jsou jeho typy. Dále se zde dočte o cyklu života informačního cyklu a o nejpoužívanějších metodách při vývoji. Další součástí teoretické části bylo seznámení čtenáře s modelovým jazykem UML, s jeho využitím a s diagramy, které jsou jeho součástí. Dále byly čtenáři představeny použité technologie při vývoji informačního systému, jejich historie a základní syntaxe.

Druhou část práce tvoří část praktická. V této části proběhla nejdříve analýza požadavků na systém. Čtenáři je zde představen Use Case diagram systému, kde je vidět, jaké role uživatelů nakonec v systému budou a jaké funkce každá z těchto rolí bude moci vykonávat. Jsou zde definovány i scénáře užití systému, které říkají, jak se systém chová při interakci s uživatelem.

Dále proběhl návrh systému, kde byl čtenáři představen návrh výsledné databáze, se kterou bude informační systém pracovat. Je zde také k nahlédnutí náčrt, jak by měla stránka po dokončení vývoje vypadat.

Nakonec proběhla implementace, kdy podle stanovených návrhů a analýzy byl naprogramován výsledný informační systém. Pro čtenáře zde byla připravena ukázka několika funkcí a vysvětlení, jak tyto funkce fungují. Systém byl naprogramován za pomoci HTML a CSS, logická část front-end části aplikace je tvořena jazykem JavaScript a jeho knihovnou React, jako databáze byla zvolena databáze MySQL a ke komunikaci s ní byl použit programovací jazyk PHP.

Výsledný systém byl nahrán na web hosting a je dostupný na adrese www.sis-mczako.cz. V databázi byl vytvořen účet administrátora, aby se mohla aplikace vyzkoušet.

6 Seznam použitých zdrojů

1. **Zwass, Vladimir.** Information system. *Britannica.com.* [Online] [Citace: 13. 03 2021.] https://www.britannica.com/topic/information-system/Information-systems-in-the-economy-and-society.

2. **Koďousková, Barbora.** INFORMAČNÍ SYSTÉMY V KOSTCE: ERP, CRM, IMPLEMENTACE. *Rascasone.com.* [Online] [Citace: 03. 13 2021.] https://www.rascasone.com/cs/blog/informacni-systemy-erp-crm-implemetace.

3. **Kocan, Marek.** Co vlastně je informační systém a jak souvisí s řízením? *Živě.cz.* [Online] [Citace: 13. 03 2021.] https://www.zive.cz/clanky/co-vlastne-je-informacni-%20system-a-jak-souvisi-s-rizenim/sc-3-a-4436/default.aspx.

4. **Šmíd, Vladimír.** Životní cyklus informačního systému. *FI MUNI.* [Online] [Citace: 14. 03 2021.] https://www.fi.muni.cz/~smid/mis-zivcyk.htm.

5. **Čápka, David.** Lekce 1 - Úvod do UML. *ITnetwork.cz.* [Online] [Citace: 14. 03 2021.] https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy.

6. **Škutová, Jolana.** http://projekty.fs.vsb.cz/147/ucebniopory/978-80-248-2766-7.pdf. [Online] [Citace: 14. 03 2021.] http://projekty.fs.vsb.cz/147/ucebniopory/978-80-248-2766-7.pdf.

7. **Čápka, David.** Lekce 2 - UML - Use Case Diagram. *ITnetwork.cz.* [Online] [Citace: 14. 03 2021.] https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram.

8. —. Lekce 5 - UML - Class diagram. *ITnetwork.cz.* [Online] [Citace: 14. 03 2021.] https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model.

9. **Rejnková, Petra.** Diagram aktivit. *uml.czweb.org.* [Online] [Citace: 14. 03 2021.] http://uml.czweb.org/diagram_aktivit.htm.

10. **GeeksforGeeks.** HTML | Introduction. *GeeksforGeeks.org.* [Online] [Citace: 14. 03 2021.] https://www.geeksforgeeks.org/html-introduction/.

11. **Kosek, Jiří.** Historie a vývoj HTML. *htmlguru.cz.* [Online] [Citace: 14. 03 2021.] http://htmlguru.cz/uvod-historie.html.

12. **tutorialspoint.** What is CSS? *tutorialspoint.com.* [Online] [Citace: 14. 03 2021.] https://www.tutorialspoint.com/css/what_is_css.htm.

13. **Guru99.** What is JavaScript? Complete Introduction with Hello World! Example. *guru99.com.* [Online] [Citace: 14. 03 2021.] https://www.guru99.com/introduction-to-javascript.html.

14. **Morris, Scott.** TECH 101: WHAT IS REACT JS? *skillcrush.com.* [Online] https://skillcrush.com/blog/what-is-react-js/.

15. **Sufiyan, Taha.** What is ReactJS: Introduction To React and Its Features. *simplilearn.com.* [Online] [Citace: 14. 03 2021.] https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs.

16. **Agarwal, Harsh.** PHP | Introduction. *geeksforgeeks.com.* [Online] [Citace: 15. 03 2021.] https://www.geeksforgeeks.org/php-introduction/.

17. **Best-hosting.cz.** CO JE TO DATABÁZE MYSQL? *Best-hosting.cz.* [Online] [Citace: 14. 03 2021.] https://best-hosting.cz/cs/napoveda/co-je-to-databaze-mysql.