Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

****

**Informační systém pro správu výukových kurzů programování**BAKALÁŘSKÁ PRÁCE

Studijní program: Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Autor: Petr Svoboda

Vedoucí bakalářské práce: Ing. Viet Bach Nguyen

Praha, prosinec 2023

Poděkování

Touto cestou bych rád poděkoval panu Ing. Viet Bach Ngyuen za trpělivé vedení mé práce, konzultace a cenné rady během jeho vedení. Zároveň i za rychlou komunikaci a odezvu k mé práci.

Abstrakt

Informační systémy pro správu výukových kurzů se zabývají správou kurzů a jejich nedílnými součástmi. Bakalářská práce pojednává o postupu vytváření informačního systému pro správu výukových kurzů programování. Cílem této práce je navrhnout a implementovat systém pro správu výukových kurzů do společnosti poskytující kurzy programování s ohledem na možnost použití systému i v jiných společností. S klientem je nejprve sestaven seznam požadavků a funkcionalit, které očekává od systému. Poté je provedena analýza podnikových procesů, systémů dostupných na trhu a krátce je věnováno analýze koncového uživatele. Dále se práce zaměřuje na popis použitých technologií a návrh aplikace, který je schválen klientem. Dále je tento systém implementován a spuštěn v testovacím režimu pro provedení testování lektory a klientem. V poslední řadě se práce zaměřuje na sestavení možného plánu nasazení systému a možnosti budoucího rozšíření. Hlavním výsledkem celé práce je systém, který je naimplementován a odsouhlasen klientem.

Klíčová slova

Informační systém, správa kurzů, webová aplikace, MERN stack, JavaScript

JEL Klasifikace

C88, L86

Abstract

This bachelor's thesis discusses the process of creating an information system for the management of programming training courses. The goal of this work is to design and implement a system for the management of training courses in a company providing programming courses, taking in to account the possibility of using the system in other companies as well. First, a list of requirements and functionalities expected from the system is drawn up with the client. After that, an analysis of business processes and systems available on the market is carried out and a brief analysis of the end user is given. Furthermore, the work focuses on the description of the technologies used and the design of the application, which is approved by the client. Furthermore, this system is implemented and launched in test mode for testing by lecturers and clients. Finally, the work focuses on drawing up a possible system deployment plan and the possibility of future expansion. The main result of the entire work is a system that is implemented and approved by the client.

Keywords

Information system, course management, web application, MERN stack, JavaScript

JEL Classification

C88, L86

Obsah

[Úvod 11](#_Toc151662705)

[1 Informační systémy pro správu a administraci výukových kurzů 13](#_Toc151662706)

[1.1 Sběr a analýza dat 13](#_Toc151662707)

[1.2 Personalizované učení 13](#_Toc151662708)

[1.3 Komunikace a zapojení 14](#_Toc151662709)

[2 Analýza potřeb poskytovatele kurzů programování 15](#_Toc151662710)

[2.1 O společnosti 15](#_Toc151662711)

[2.1.1 Poskytování kurzů 15](#_Toc151662712)

[2.1.2 Vývoj společnosti 15](#_Toc151662713)

[2.1.3 Současný stav systému pro správu v dané společnosti 17](#_Toc151662714)

[2.1.4 Požadavky na nový systém 17](#_Toc151662715)

[2.2 Postup vývoje systému 17](#_Toc151662716)

[2.3 Klíčové procesy společnosti 18](#_Toc151662717)

[2.3.1 Správa kurzů a lekcí 18](#_Toc151662718)

[2.3.2 Správa uživatelů 19](#_Toc151662719)

[2.3.3 Správa docházky a rozvrhů 19](#_Toc151662720)

[2.3.4 Komunikace s uživateli 19](#_Toc151662721)

[2.4 Podnikové procesy v systému 19](#_Toc151662722)

[2.4.1 Proces tvorby uživatele 21](#_Toc151662723)

[2.4.2 Proces tvorby kurzu 22](#_Toc151662724)

[2.4.3 Proces tvorby lekce 23](#_Toc151662725)

[2.4.4 Proces tvorby třídy 24](#_Toc151662726)

[2.5 Požadavky 25](#_Toc151662727)

[2.5.1 Funkční požadavky 25](#_Toc151662728)

[2.5.2 Nefunkční požadavky 30](#_Toc151662729)

[2.6 Typy uživatelů systému 30](#_Toc151662730)

[3 Rešerše dostupných řešení na trhu 31](#_Toc151662731)

[3.1 Sawyer 31](#_Toc151662732)

[3.2 Teach‘n go 33](#_Toc151662733)

[3.3 Tabulka porovnání 37](#_Toc151662734)

[3.4 Shrnutí dostupných řešení 38](#_Toc151662735)

[4 Životní cyklus projektu 39](#_Toc151662736)

[4.1 Fáze životního cyklu projektu 39](#_Toc151662737)

[4.2 Role projektu 39](#_Toc151662738)

[4.3 Fáze zahájení 40](#_Toc151662739)

[4.3.1 Vize projektu 40](#_Toc151662740)

[4.3.2 Požadavky 40](#_Toc151662741)

[4.3.3 Plán projektu 42](#_Toc151662742)

[4.4 Fáze rozpracování 42](#_Toc151662743)

[4.4.1 Tvorba správy uživatelů 43](#_Toc151662744)

[4.4.2 Autentizace uživatele 43](#_Toc151662745)

[4.4.3 Správa firmy 43](#_Toc151662746)

[4.4.4 Tvorba částí výuky 43](#_Toc151662747)

[4.4.5 Docházkový systém 44](#_Toc151662748)

[4.4.6 Hlavní obrazovka 44](#_Toc151662749)

[4.4.7 Iterace testování 44](#_Toc151662750)

[4.5 Fáze konstrukce 44](#_Toc151662751)

[4.6 Fáze zavedení 45](#_Toc151662752)

[5 Návrh aplikačního systému 46](#_Toc151662753)

[5.1 Návrhový vzor MVC 46](#_Toc151662754)

[5.2 ExpressJS server a Rest API 47](#_Toc151662755)

[5.3 Databáze 49](#_Toc151662756)

[5.4 Uživatelské rozhraní 50](#_Toc151662757)

[5.5 Použité technologie 52](#_Toc151662758)

[6 Implementace aplikace 55](#_Toc151662759)

[6.1 Správa uživatelů a profil 55](#_Toc151662760)

[6.2 Kurzy a lekce 55](#_Toc151662761)

[6.3 Třídy 56](#_Toc151662762)

[6.4 Docházka a rozvrhy 56](#_Toc151662763)

[6.5 Admin správy 56](#_Toc151662764)

[6.6 Databáze 56](#_Toc151662765)

[6.7 Rozdělení kódu 57](#_Toc151662766)

[6.7.1 Backend 58](#_Toc151662767)

[6.7.2 Frontend 62](#_Toc151662768)

[7 Plán testování a nasazení aplikace 66](#_Toc151662769)

[7.1 Uživatelské testování 66](#_Toc151662770)

[7.2 Plán nasazení 66](#_Toc151662771)

[7.3 Možnosti pokračování 67](#_Toc151662772)

[Závěr 69](#_Toc151662773)

[Použitá literatura 71](#_Toc151662774)

[Přílohy I](#_Toc151662775)

[Příloha A: Kód a demo aplikace I](#_Toc151662776)

[Příloha B: Interní materiály I](#_Toc151662777)

[Příloha C: Testovací scénáře I](#_Toc151662778)

[Příloha D: Ukázky výsledné aplikace V](#_Toc151662779)

Seznam obrázků

[Obrázek 1 - Počty kurzů (zdroj: (CODE Rebels, 2023)) 16](#_Toc151662780)

[Obrázek 2- Počty dětí a lektorů (zdroj: (CODE Rebels, 2023)) 16](#_Toc151662781)

[Obrázek 3 - Proces vytváření nového uživatele (zdroj: autor) 21](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662782)

[Obrázek 4 - Proces vytváření nového kurzu (zdroj: autor) 22](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662783)

[Obrázek 5 - Proces vytváření lekce (zdroj: autor) 23](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662784)

[Obrázek 6 - Proces tvorby třídy (zdroj: autor) 24](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662785)

[Obrázek 7 - Diagram případu užití, první část (zdroj: autor) 28](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662786)

[Obrázek 8 - Diagram případů užití, druhá část (zdroj: autor) 29](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662787)

[Obrázek 9 - Sawyer příklad rozvrhu (zdroj: (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023)) 31](#_Toc151662788)

[Obrázek 10 - Sawyer detail kurzu seznam studentů (zdroj: (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023)) 32](#_Toc151662789)

[Obrázek 11 - Sawyer seznam kurzů (zdroj: (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023)) 33](#_Toc151662790)

[Obrázek 12 - Tvorba systému (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023)) 34](#_Toc151662791)

[Obrázek 13 - Hlavní obrazovka (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023)) 34](#_Toc151662792)

[Obrázek 14 - Teach'n go příklad rozvrhu (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023)) 35](#_Toc151662793)

[Obrázek 15 - Teach'n go příklad kurzu/třídy (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023)) 35](#_Toc151662794)

[Obrázek 16 - Teach'n go příklad tvorby kurzu/lekce (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023)) 36](#_Toc151662795)

[Obrázek 17 - Model databáze (zdroj: autor) 49](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662796)

[Obrázek 18 - Wireframe úvodní obrazovky (zdroj: autor) 50](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662797)

[Obrázek 20 - Wireframe formuláře pro tvorbu lekce (zdroj: autor) 51](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662798)

[Obrázek 19 - Wireframe detailu lekce (zdroj: autor) 51](file:///C:\Users\petrp\OneDrive\Plocha\Petr%20Svoboda%20-%20BP%20-%20Informační%20systém%20pro%20správu%20výukových%20kurzů%20programování.docx#_Toc151662799)

[Obrázek 21 - Seznam tříd (zdroj: autor) V](#_Toc151662800)

[Obrázek 22 - Detail lekce (zdroj: autor) V](#_Toc151662801)

[Obrázek 23 - Rozvrh třídy (zdroj: autor) VI](#_Toc151662802)

Seznam výpisu programového kódu

[Výpis 1 - Ukázka části controlleru lekcí (zdroj: autor) 47](#_Toc151662803)

[Výpis 2 - Ukázka rout lekcí (zdroj: autor) 48](#_Toc151662804)

[Výpis 3 - Ukázka MongoDB schématu (zdroj: autor) 52](#_Toc151662805)

[Výpis 4 - Struktura kódu (zdroj: autor) 57](#_Toc151662806)

[Výpis 5 - Konfigurace express serveru (zdroj: autor) 59](#_Toc151662807)

[Výpis 6 - Controller lekcí (zdroj: autor) 60](#_Toc151662808)

[Výpis 7 - Ukázka části lessonSlice s Reducerem lekcí (zdroj: autor) 62](#_Toc151662809)

[Výpis 8 - Ukázka části lessonService (zdroj: autor) 64](#_Toc151662810)

[Výpis 9 - Ukázka komponenty Input (zdroj: autor) 65](#_Toc151662811)

Seznam zkratek

IS Informační systém

CRM Customer Relationship Management

JS Javascript

MERN MongoDB, ExpressJS, ReactJS, NodeJS

MEAN MongoDB, ExpressJS, Angular, NodeJS

UI User Interface

UX User Experience

API Application Programming Interface

JSON Javascript Object Notation

BSON Binary JSON

NoSQL Non-SQL

EJS Embedded JavaScript

HTML Hypertext Markup Language

CSS Cascading style sheets

SASS Syntactically Awesome Style Sheets

SMTP Simple Mail Transfer Protocol

POP3 Post Office Protocol version 3

IMAP Internet Message Access Protocol

MVC Model View Controller

JSX JavaScript Syntax Extension

Úvod

Nejen pandemie COVID-19 změnila směr a trendy dnešní doby a nejen to. Lidé si zvykli pracovat z domova, kde vše potřebné řešili online. Proto se většina společností přizpůsobila a vynakládají velkou snahu, aby všechny schůzky, porady a zasedání konaly online pomocí nejrůznějších řešení. Školy, školící a vzdělávací společnosti musely přesunout nejrůznější lekce a kurzy do online prostoru z důvodu bezpečnosti studentů. (Tudor, 2022)

S odstupem pandemie zůstala vysoká poptávka po online kurzech, a proto se společnost[[1]](#footnote-1) rozhodla pořídit informační systém pro správu kurzů jak pro online lekce, tak i pro lekce ve třídách. Druhým důvodem pro pořízení systému je neustálé rozšiřování nabídky nejrůznějšími výukovými kurzy programování. Nejdůležitějším důvodem pro zavedení systému je velké množství přibývajících studentů.

Požadavky pro dosažení cíle této práce je popsat současný stav společnosti, analyzovat podnikové procesy a identifikovat klíčové procesy, prozkoumat existující řešení na trhu. Cílem této práce je navrhnout a implementovat nový informační systém pro správu výukových kurzů programování. Vlastní systém přináší spoustu zajímavých příležitostí, lze ho přizpůsobit co nejblíže parametrům zadavatele, rychlá řešení, opravy problémů a implementace nových funkcionalit a mnoho dalších. Důležitou součástí této práce bude porovnání existujících řešení na trhu a jejich různých funkcionalit. Z daného porovnání poté bude možné využít funkcionality, které lze naimplementovat do nově implementovaného systému, pokud je jednatel bude také požadovat, nebo které usnadní správu kurzů nebo práci se systémem.

Tato bakalářská práce je organizována následujícím způsobem. V první kapitole bude představena problematika systému pro správu výukových kurzů a také společnost, pro kterou je tento systém vyvíjen, avšak vždy s ohledem na možnosti implementace systému pro jiné společnosti. Ve druhé kapitole proběhne analýza potřeb poskytovatele kurzů programování a požadavků na systém. Analýze se rozdělí na analýzu podnikových procesů společnosti, určení klíčových procesů a možnosti jejich automatizace a požadavků klienta na systém. Třetí kapitola se zaměří na rešerši dostupných řešení na trhu. Čtvrtá kapitola obsahuje návrh aplikačního systému a jeho součástí. Pátá kapitola se zaměří na popis implementace aplikace a použitých technologií, rozdělení kódu a ukázky z již naprogramovaného systému. V šesté kapitole je popis nasazení a testování aplikace a možnosti jednotlivých technologií a jejich plánů. Poslední kapitola obsahuje další možnosti rozšíření a obohacení systému. V závěru je shrnuto splnění cílů aplikace a obsah celé práce.

Tato práce může sloužit jako vzor pro vybudování podobného systému ať pro správu kurzů nebo pro správu obdobných objektů. Práce obsahuje odbornou terminologii a zaměřuje se na složité téma a tím by měl být uživatel znalých základů programování v jazyku Javascript a znalých základů vývoje webových aplikací a jejich základních technologií.

# Informační systémy pro správu a administraci výukových kurzů

Jedná se o softwarové aplikace, které se používají pro správu dat studentů, kurzů a administrativních úkonů. Tyto systémy jsou navrženy za účelem zefektivnění a automatizace provozu poskytovatelů kurzů a umožnily správcům, lektorům a zaměstnancům efektivněji spravovat záznamy studentů, rozvrhy hodin, docházku, třídy, lekce a další úkoly související s poskytováním kurzů kurzy. Řadí se mezi CRM systémy, přesněji CRM systémy ve vzdělávání. (Daradoumis *et al.*, 2010)

CRM systémy, česky Systémy pro řízení vztahů se zákazníky, se zabývají udržování dlouhodobých vztahů se zákazníky. V případě CRM systémů ve vzdělávání lze hovořit o zvyšování zapojení a udržení studentů ve školách a školících zařízení. CRM ve vzdělávání také zahrnuje použití technologických řešení a postupů pro řízení a zlepšování interakcí mezi školami a studenty. Hlavním cílem těchto systémů je vytvoření personalizované vzdělávací zkušenosti, která splňuje potřeby studentů, zlepšuje komunikaci mezi studenty a školami a zlepšuje spokojenost studentů a míru udržení. Ke splnění cíle CRM systému ve vzdělávání je potřeba daný systém rozdělit na tři hlavní složky viz jednotlivé sekce (Daradoumis *et al.*, 2010).

## Sběr a analýza dat

Školy shromažďují data o počtu studentů, jejich studijních výsledcích, chování a angažovanosti studentů. Tato data se analyzují, aby se identifikovali studenti, kteří mohou mít problémy, a aby se vyvinula řešení těchto problémů, a aby byla možnost porovnat výsledky školy a studentů a jak se škola a studenti vyvíjí. (Daradoumis *et al.*, 2010)

Sběr a analýza dat probíhá několika metodami. První metody jsou tzv. metody observační, jedná se o pozorování jednotlivců a jejich výsledků. Další metody jsou rozhovory a dotazníky, které shromažďující data prostřednictvím záměrně cílených otázek. V případě edukačních systému se jedná o zpětné vazby. Poslední metodou je dokumentace a staré analýzy, které slouží za účelem získání údajů z minulosti. (Daradoumis *et al.*, 2010)

## Personalizované učení

Školy využívají analýzu dat k vytváření přizpůsobených vzdělávacích zkušeností pro studenty. To zahrnuje přizpůsobení výuky jedinečným vzdělávacím potřebám každého studenta a poskytování podpůrných služeb, jako je doučování, poradenství a akademické koučování. (Daradoumis *et al.*, 2010)

Každý student má své silné stránky, potřeby, dovednosti a zájmy. Personalizované učení se následně snaží, co nejvíce zaměřit na jednotlivé vlastnosti a sestavit ideální učební plán. Učební plán je sestaven dle znalostí jednotlivce a dle metody učení, která mu nejvíce vyhovuje. Personalizované učení nenahrazuje intervenční a preventivní školní programy. (Morin, 2019)

## Komunikace a zapojení

Školy používají technologická řešení, jako je e-mail, textové zprávy, sociální média, vlastní webové stránky atd., aby komunikovaly se studenty a udržely je v zapojení do učení co nejdéle. To zahrnuje poskytování pravidelných aktualizací o hodnocení, úkolech a nadcházejících událostech ať už vzdělávacích anebo kulturních apod. (Daradoumis *et al.*, 2010)

Zavedením systému CRM do vzdělávání mohou školy a školící zařízení zlepšit zapojení studentů, udržení a akademický úspěch. Mohou také zlepšit provozní efektivitu pomocí automatizace rutinních úloh a snížením administrativní zátěže. Úspěšná implementace CRM systému ve vzdělávání však vyžaduje pečlivé plánování, spolupráci a průběžné hodnocení jak od lektorů a zaměstnanců tak od studentů, aby bylo zajištěno, že řešení daného systému bude přizpůsobeno jedinečným potřebám každého studenta a školy. (Hrnjic, 2016)

Celkově může použití CRM ve vzdělávání poskytnout řadu výhod, včetně lepšího zapojení studentů, personalizované podpory a efektivnější komunikace. Implementace CRM systémů však vyžaduje pečlivé plánování, školení a trvalou podporu, aby bylo zajištěno, že systém bude efektivně využíván a poskytuje zamýšlené výhody. (Hrnjic, 2016) V dalším části bude pojednáváno o daném klientovi, pro kterého je systém vytvářen.

# Analýza potřeb poskytovatele kurzů programování

Tato kapitola se zaměřuje na popis společnosti, analýzu klíčových procesů, funkčních a nefunkčních požadavků společnosti. Klíčových procesů je hned několik, jedná se o správu kurzů, komunikaci se studenty a jejich rodiči a také o správě docházky. Požadavky jsou rozděleny na několik částí, a to funkční a nefunkční. Další podkapitola se zabývá několika alternativními řešeními na trhu, jednotlivé řešení budou porovnána dle několika společných kritérií. V poslední časti budou krátce shrnuty typy koncových uživatelů.

## O společnosti

Společnost CODE Rebels poskytuje kurzy programování dětem i dospělým od roku 2018 a od té doby poskytla více než 2000 hodin a odučila přes 1000 studentů. Společnost řídí a vlastní pouze jedna jednatelka. Důvodem založení společnosti je nedostatečná výuka programování na základních školách, a proto se jednatelka rozhodla založit společnost a poskytovat kurzy. Kurzy se poskytují převážně online, ale jsou i lekce ve školách, a dokonce i ve firmách. Každý kurz se skládá z několika lekcí a většinou se vyučují dvakrát týdně.

### Poskytování kurzů

Kurzy společnost poskytuje online i prezenčně, ale online kurzy jsou mnohem více preferované. Společnost také nabízí kurzy školám a firmám. Nyní společnost nabízí kurzy programování v jazyce JavaScript a Python poté poskytuje kurzy ve vývojovém prostředí Scratch, spojování barevných bloků, které představují části kódů do bloků kódů a následné naprogramování hry. V poslední řadě také nabízí kurz 3D modelování v aplikaci TinkerCAD.

Kurzy se skládají z 15 lekcí, vždy se pracuje na několika projektech, které studenti mohou dále rozvíjet. Lekce se vyučují jednou týdně, pokud student nemůže na danou lekci, lze si vybrat jinou nebo lze poskytnout lekci náhradní. Většina kurzů se vyučuje alespoň dvakrát týdně.

### Vývoj společnosti

První rok na trhu byl pro společnost náročný, jakožto nováček na trhu se musel překonat všechny překážky jako každý, ale už tak se podařilo odučit 45 studentů a poskytnou 4 kurzy a z toho byl 1 kurz prezenčně ve třídě společnosti, jedna lekce na základní škole a 2 lekce ve jedné firmě. V roce 2018 ještě online kurzy nebyly poskytovány. První rok byly kurzy odučeny čtyřmi lektory. Následující rok byl úspěšnější. Přibyl jeden lektor a celkem 5 lektorů odučilo 9 kurzů a celkem 75 dětí. Každý další rok děti a kurzy přibývaly a bylo i více lektorů, podrobnější informace viz Obrázek 1 a 2. Oba obrázky zobrazují postup vývoje ve školních letech. Obrázek 1 znázorňuje vývoj počtu kurzů od vzniku společnosti po 1. pololetí roku 2023. Rozděleno podle typu kurzu, jak je z obrázku patrné celkem se nejvíce odučilo kurzů online a to 88.

Obrázek - Počty kurzů (zdroj: (CODE Rebels, 2023))

Obrázek 2 interpretuje vývoj počtu dětí a lektorů od vzniku společnosti po 1. pololetí roku 2023. Z obrázku lze vypozorovat, že děti neustále přibývají a celkem lektoři odučili 891 dětí. Data za rok 2023 nejsou kompletní, jedná se pouze o první školní pololetí.

Obrázek - Počty dětí a lektorů (zdroj: (CODE Rebels, 2023))

Z čísel je jasné, že studenti přibývají každým rokem a tím rostou i počty kurzů, a proto se společnost rozhodla zavést vlastní systém pro snazší správu kurzů.

### Současný stav systému pro správu v dané společnosti

Současný stav začíná být velmi nevyhovující z několika důvodů. Používá se několik různých systémů a s přibývajícím množstvím dětí a kurzů začíná být obsluha těchto systému složitější, z důvodu potřeby správy dat na několika místech, nepřehlednosti systémů a přeplnění systémů. Další problém je přibývající množství materiálů na kurzy. Všechny materiály jsou uloženy na sdíleném cloudovém úložišti Google Drive, jsou rozdělené ve složkách, ale začíná jich být více a více a za nedlouho bude úložiště přeplněné a nepřehledné. Docházkový systém je poměrně zastaralý a zbytečně komplikovaný. Tento systém není příliš uživatelsky přívětivý, nabízí příliš mnoho funkcionalit, které nejsou využívány. Důvodem přechodu na nový systém je i posílání emailů dětem a lektorům z kurzů. Nynější systém je komplikovaný, musí se vybírat po jednom studentovi, což je velmi neefektivní.

### Požadavky na nový systém

Ve společnosti je jediný jednatel a stejný je i vlastník, proto všechny rozhovory probíhaly osobně nebo online. První schůzka byla o představení současného stavu správy kurzů, viz předchozí kapitola. Zároveň klient představil své požadavky na funkcionalitu, které vyžaduje od nového systému. Následně mi byly předány potřebné materiály z předchozích kurzů.

Z dalších schůzek vyplynulo, že nový systém prozatím budou využívat pouze lektoři a administrátoři, ale jakmile se odzkouší lektory a administrátory, tak se systém nejdříve zpřístupní pro starší studenty, kteří lépe dokážou popsat případné nedostatky a chyby v systému a poté pro mladší. Takto se rozhodl klient a bude se od toho vyvíjet i další práce.

Předmětem dalších schůzek bylo upřesnění funkčních požadavků a jakmile byl systém v pokročilejším stavu, proběhly ukázky systému a jeho následné zdokonalování a potřebné úpravy dle domluvy s klientem. Z těchto požadavků vyplynulo, že systém lze rozdělit na několik částí, a to správa uživatelů, správa kurzů a lekcí, docházkový systém a posílání upozornění o změnách na email. Dále bude systém obsahovat i zpracování zpětné vazby.

Důležitým požadavkem je obecnost systému, nepsat systém přímo na míru dané společnosti, ale vytvořit systém, tak aby mohl být použit jakoukoliv další společností, jako cloudové řešení. Systém bude složen z několika podsystémů a funkcionalit, které bude jednoduché přidávat, upravovat a odebírat. Více a podrobněji v další kapitole.

## Postup vývoje systému

Vývoj systému bude prováděn pomocí agilní metodiky. Agilní metodika pojednává o nestálých cyklech, v případě tohoto systému bude vypadat cyklus následovně. Postup cyklu vývoje systému se rozdělí na několik hlavních částí, jako jsou například plánování a analýza, návrh, vývoj a údržba ve fázi plánování a analýzy je velmi důležitá komunikace s klientkou a vyjasnění si všech nedorozumění, určení, kdo bude koncový uživatel a jaké účely by měl systém splňovat, a hlavně co klient chce postavit. Další důležitou součástí je analýza společnosti a stanovaní všech činností a procesů a určení které činnosti jsou pro společnost klíčové, a které lze zautomatizovat. Tento cyklus se bude opakovat do té doby, dokud bude systém využívaný. (Software Testing Help, 2023)

V průběhu fáze návrhu se věnuje pozornost rozdělení systému na části, popřípadě podsystémy. Vytváří se schéma databáze, modely obrazovek, určení rolí apod. Následně se určí technologie, které budou použity. V další části tvorby systému, vývoj, se odehrává nejdůležitější činnost, a to programování a testování. Během programování se klade velký důraz kvalitní uživatelskou zkušenost (UX/UI). Při testování je důležité zapojit alespoň pár budoucí uživatelů, aby systém prošel tzv UAT. (Software Testing Help, 2023) Pokud se vše splní a klientka je spokojená, lze systém nasadit.

Po nasazení nás čeká provoz a údržba systému, což znamená neustálé zdokonalování systému, různé vylepšení, zavádění nových funkcí a opravování nedostatků. Poté může přijít na řadu audit a pokud se systém stane nevyhovující nebo zastaralý, nastane jeho vyřazení. (Sanie, 2022). V případě dané společnosti audit nenastane.

## Klíčové procesy společnosti

Tato část se zabývá procesní analýzou, jedná se tedy o soupis sad kroků nebo postupů k dosažení prováděné činnosti, neboli popis způsobu, jakým je určitá činnost prováděna ve společnosti (IBM Cloud Education, 2021). Jednotlivé klíčové procesy představují jednotlivé podkapitoly. Každá podkapitola je zaměřena na popis postupů a také možnost automatizace částí procesu. Bude pojednáváno o těchto procesech: Správa kurzů a lekcí, Správa uživatelů, Správa docházky a rozvrhů, Správa tříd a Komunikace s uživateli. Jedná se popis současného způsobu fungování systému ve společnosti.

### Správa kurzů a lekcí

Základní proces firmy je správa a poskytování kurzů a lekcí. Pro vytvoření kurzu je potřeba ho zapsat do tabulky na Google Disk a vytvořit docházky v docházkovém systému. Dále musí lektor kurzu vytvořit třídu v Google Classroom a zapsat odkaz do stejné tabulky. Poté se musí ručně poslat email studentům, popřípadě jejich rodičům s informacemi o kurzu a s odkazem na online lekci nebo adresou. Po každé lekci lektor zapíše, co se dělalo na hodině a popřípadě nasdílí kódy do Google Classroomu.

Co se týká správy lekcí, jsou to pouze dokumenty ve Wordu, které jsou taktéž uloženy na Google Disku. Jednotlivé dokumenty se skládají ze dvou částí, osnov a přednášek. Údržba těchto dokumentů není snadná, protože se každý soubor musí spravovat zvlášť. Jednotlivé soubory jsou rozděleny do příslušných složek dle kurzů, přestože je to přehledné tak to práci spíše komplikuje. Už nyní lze říct, že tyto procesy nejsou optimální a některé části potřebují zautomatizovat.

### Správa uživatelů

Druhým důležitým procesem společnosti je správa uživatelů. Vyskytuje se na dvou místech, a to v tabulkách na Google Disku, kde je seznam lektorů a počty studentů v jednotlivých kurzech. Druhá část se nachází v docházkovém systému, kde jsou sepsáni všichni studenti a lektoři a kontakty na ně. Tento způsob není příliš vhodný z hlediska duplicity správ, protože musí se kontrolovat, zda se data v tabulkách a v docházkovém systému shodují.

### Správa docházky a rozvrhů

Ke správě docházky společnost používala externí docházkový systém, ale nově se docházka zapisuje už jen do třídních knih, které si spravuje každý lektor sám. Docházkový systém je zastaralý a zbytečně komplikovaný. Obsahuje přespříliš různých funkcí, které nelze využít, avšak přesto se pořád používá na emailovou komunikaci. Třídní knihy v tabulkách vypadají jen jako tabulka studentů a datumů lekcí, kam se zapíše, zda jednotliví studenti byli přítomni.

Správa rozvrhů je k dispozici jen pro lektory a administrátory, ale je to opět jen tabulka v excelu na Google Disku. V dané tabulce se nachází odkdy, dokdy a v kolik kurz probíhá a odkaz na Google Classroom a online lekci a také lektor a počty přihlášených studentů. Opět tento systém není vhodný, často nastává problém ve sdílení třídních knih mezi lektory a náhradními lektory. Bohužel někdy nemají přístup k potřebným knihám. Další problémem, který naštěstí již nepřetrvává, je nekonzistence jednotlivých knih.

### Komunikace s uživateli

Jak již bylo zmíněno v předchozích kapitolách, společnost používá Google Classroom a emailovou komunikaci. Google Classroom používají převážně lektoři a studenti a lektoři tam zapisují co proběhlo na dané lekci. Studenti se mohou ptát lektora nebo ostatních studentů na nejrůznější dotazy. Také se používá ke sdílení naprogramovaných kódů a dalších materiálů z lekcí.

Emailová komunikace probíhá pomocí docházkového systému, kam nejdříve musí administrátor zapsat jednotlivé studenty a kontakty na ně a na jejich rodiče. Poté přiřadí studenty do kurzů a následně může administrátor posílat emaily jednotlivým studentům anebo celým skupinám a kurzům. Emailová komunikace převážně slouží, jako rychlá komunikace na nejdůležitější upozornění na zrušení lekce nebo posun začátku lekce apod. Emailová komunikace se také používá také jako informativní a marketingové oznámení.

## Podnikové procesy v systému

Nový systém se rozdělí na několik funkčních částí a jednotlivé části budou podobné jako v předchozích řešení. Konktrétně se jedná o tyto části: správa kurzů a lekcí, správa uživatelů, správa tříd, správa docházky a rozvrhů a komunikace s uživateli. Nejdůležitější části jsou vymodelovány jako procesy a jsou zobrazeny na Obrázku 3, 4, 5 a 6. Tyto diagramy byly vymodelovány pomocí metodiky EPC. EPC metodika znamená událostmi řízený procesní řetězec a pojednává tedy o tvorbě procesů za pomoci grafů, které popisují pracovní postupy, podnikové a obchodní procesy. (Microsoft, 2023)

### Obsah obrázku diagram Popis byl vytvořen automatickyProces tvorby uživatele

Obrázek - Proces vytváření nového uživatele (zdroj: autor)

### Obsah obrázku diagram Popis byl vytvořen automatickyProces tvorby kurzu

Obrázek - Proces vytváření nového kurzu (zdroj: autor)

### Proces tvorby lekce

Obrázek - Proces vytváření lekce (zdroj: autor)

### Obsah obrázku diagram Popis byl vytvořen automatickyProces tvorby třídy

Obrázek - Proces tvorby třídy (zdroj: autor)

Základem metodiky EPC je navázání událostí na funkce, které spouští další události, a tak to probíhá až do konečné události. Každý proces vymodelovaný pomocí EPC metodiky se skládá ze dvou základních událostí a spouštěcí události a konečné události. V případě vymodelovaných procesů je vždy spouštěcí událost přihlášení uživatele s rolí admin nebo lektor do systému. Každý proces končí úspěšném vytvoření daného objektu tj vytvoření uživatele, kurzu, lekce nebo třídy s docházkami a rozvrhem. Jednotlivé procesy se skládají z několika událostí a funkcí.

První je spouštěcí událost přihlášení uživatele, jak již bylo řečeno. Následuje funkce zobrazení seznamu daných objektů uživatelem. Po zobrazení seznamu následuje zvolení akce tvorby objektu uživatelem a zobrazení formuláře s poli. Následně musí uživatel vyplnit daný formulář, vybrat hodnoty a vyplnit pole. Poté stačí přiřadit podružnými objekty, nebo potvrdit registraci a objekt je úspěšně vytvořen. V případě tvorby třídy se proces ukončuje i vygenerováním docházek a rozvrhů. V případě lekce je nutné uvážit, zda se jedná o admina nebo lektora. Pokud uživatel má přiřazenou roli lektor je nutné danou lekci schválit před jejím vytvořením.

## Požadavky

Tato sekce se bude zabývat již zmíněnými požadavky v kapitole 1.4, ale budou rozebrány více do detailu. Rozdělení požadavků zůstane i nadále na funkční a nefunkční. Funkční požadavky budou popsány a poté budou vyjádřeny pomocí diagramu případů užití, které budou navázány na jednotlivé typy uživatelů. Nefunkční požadavky jsou v podstatě kvalitativní omezení, která musí systém splňovat podle požadavků a potřeb klienta. Mohou se zabývat požadavky na bezpečnost, udržitelnost, spolehlivost, škálovatelnost atd. V této části jsou i požadavky na obecnost systému. (Singla, 2020)

### Funkční požadavky

Funkční požadavky se týkají následujících případů užití, které se věnují autentizaci uživatele:

* Přihlásit se
* Obnovit zapomenuté heslo pomocí odkazu zaslanému na email
* Odhlásit se

Každý uživatel musí mít možnost:

* Editace vlastních osobních údajů
* Spravovat telefonní čísla (u studentů osobní a číslo na rodiče)
* Změnit si heslo
* Zobrazit zapsané třídy
* Zobrazit vlastní rozvrh
* Zobrazit svou docházku

V seznamu kurzů musí být možnost:

* Procházet dostupné kurzy
* Zobrazit stránku konkrétního kurzu
* Filtrovat kurzy

V detailu kurzu tedy musí být:

* Zobrazit podrobnější informace o kurzu
* Zobrazit seznam lekcí, ze kterých se kurz skládá

V lekci půjde o tyto případy užití:

* Zobrazit popis lekce
* Pro lektory zobrazit detail lekce (materiály a osnovy)
* Pro lektory udělat změnu v detailu lekce
* Pro lektory vytvořit novou lekci

V seznamu tříd:

* Zobrazit třídy ve kterých je zapsán
* Zobrazit stránku konkrétní třídy
* Vyhledávat třídy

V detailu třídy:

* Zobrazit příslušné lekce z kurzu
* Zobrazit rozvrh
* Zobrazit docházku
* Zobrazit spolužáky
* Zobrazit lektory
* Pro lektory zobrazit zapsané studenty v jeho třídě
* Pro lektory zobrazit docházku studentů v jeho třídě
* Zobrazit online hovor
* Pro lektory zapsat docházku zapsaným studentům
* Pro lektory upravit docházku zapsaným studentům

Administrátor musí mít možnost:

* Přidělovat studentům náhradní lekce, pokud chyběli na poslední lekci
* Přidělovat náhrady, když chybí lektor
* Spravovat kurzy (vytvořit, zobrazit, editovat, mazat)
* Spravovat lekce (vytvořit, zobrazit, editovat, mazat, přidávat materiály a osnovy)
* Spravovat třídy
* Spravovat docházku všech uživatelů (vytvořit, zobrazit, editovat, mazat)
* Spravovat uživatele (registrovat nové, zobrazit, editovat, mazat, zakazovat/ povolovat)
* Zobrazit si seznam svých přidělených administrátorských oprávnění
* Přiřazovat práva uživatelům a rolím
* Posílat emaily studentům

Všichni uživatelé budou moci kdykoliv zaslat zpětnou vazbu a také budou moci v případě problémů kontaktovat podporu.

Uživatelem typu administrátor je myšleno takový uživatel, který má přidělené nejméně jedno administrátorské oprávnění. Z toho lze odvodit, každý administrátor může provádět takové administrátorské funkcionality, ke kterým má přidělená daná administrátorská oprávnění viz Obrázek 7 a 8. Na obrázku 7 a 8 lze vidět jednotlivá oprávnění uživatelů, podle kategorie.

Obsah obrázku diagram, řada/pruh, skica, černobílá

Popis byl vytvořen automaticky

Obrázek - Diagram případu užití, první část (zdroj: autor)

Obsah obrázku diagram, text, skica, řada/pruh

Popis byl vytvořen automaticky

Obrázek - Diagram případů užití, druhá část (zdroj: autor)

### Nefunkční požadavky

S tvorbou systému pro správu kurzů se pojí řada nefunkčních požadavků. Z hlediska legislativy musí systém splňovat následující:

* Dbát na ochranu osobních údajů dle GDPR

Mezi hlavní nefunkční požadavky řadíme (Singla, 2020):

* Bezpečnost
* Udržitelnost
* Spolehlivost
* Škálovatelnost
* Znovupoužitelnost
* Responzivita
* Dobře zpracované uživatelské rozhraní (UI)
* Kvalitní uživatelský zážitek (UX)

## Typy uživatelů systému

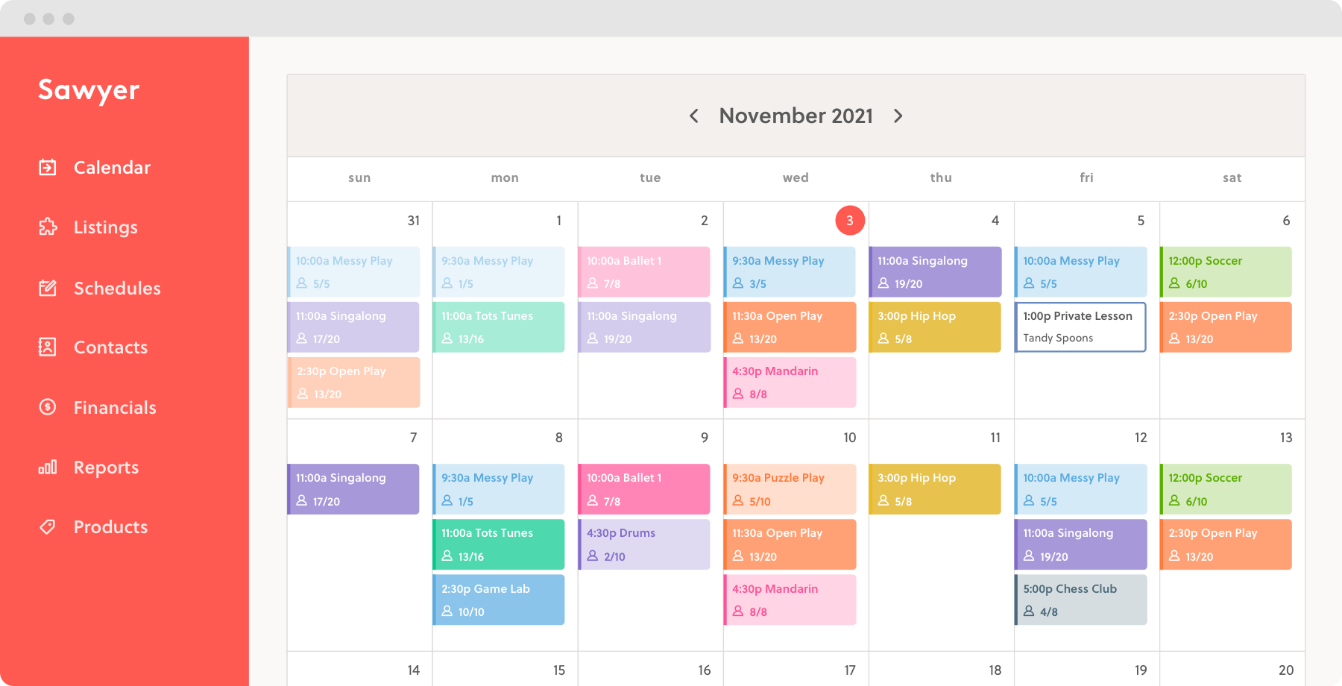
Primární cílové skupiny budou tři, a to studenti, lektoři a manažeři/administrátoři. Studenti budou uživatelé, kteří zatím systém využívat nebudou, ale do budoucna bude systém připraven i pro ně. Lektoři systém budou využívat pro zapisování docházky a k získání materiálů a osnov pro lekce. Dále také budou moci navrhovat změny v lekcích nebo i nové lekce. Administrátoři budou systém spravovat, vytvářet kurzy, lekce, uživatele apod. Cílové skupiny pokrývají většinu věkových kategorií, od studentů až po dospělé ve firmách.

# Rešerše dostupných řešení na trhu

Cílem analýzy již dostupných řešení na trhu je popsat, jaké nejdůležitější funkcionality používají ostatní systémy, jaké funkcionality mají jednotlivé systémy navíc a jaké funkcionality lze zavést i do zamýšleného systému. Různých řešení systému na správu kurzů na trhu lze najít spousty, ale mnohdy jsou příliš komplikované, drahé nebo nepraktické pro budoucí použití. Z již dostupných řešení byly vybrány dvě řešení, které splňují všechny požadavky klienta. Jedná se o systémy Sawyer a Teach ‘n go. Tyto systémy budou porovnány pomocí několika kritérií a také budou porovnány se současným způsobem provozu a výsledek porovnání bude tabulka funkcionalit. Jednotlivá kritéria jsou splnění požadavků, UI, UX, postup tvorby kurzu a jaké další funkcionality jednotlivé systémy nabízí. Systémy jsem zvolil po konzultaci s klientem a jeho průzkumem trhu. Tyto systémy byly nejblíže jeho požadavkům, avšak nenaplnily jeho očekávání a možnosti.

## Sawyer

Systém Sawyer systém pro správu škol, kroužků a dalších mimoškolních aktivit jako jsou sportovní, umělecké, vědní, jazykové kurzy a mnoho dalších. Tento systém nabízí spoustu funkcionalit, od základních jako jsou správa kurzů, lekcí, uživatelů, docházkový systém, správa rozvrhů, ale i pokročilejší funkcionality jako jsou kupování kurzů přímo v aplikaci, základní ale i detailní reporting, systém pro marketing a mnoho dalších. Tento systém nabízí několik variant od základního až po kompletní balíček. Jedná se o americký systém, kde některé funkcionality jsou dostupné pouze v USA nebo v amerických dolarech. (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023)



Obrázek - Sawyer příklad rozvrhu (zdroj: (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023))

Na obrázku výše vidíte podobu rozvrhů. Pro studenty a lektory rozvrhy vytváří správce systému. Taktéž se jedná o hlavní stránku po přihlášení uživatele. Zobrazení rozvrhu si lze také přizpůsobit na denní, týdenní nebo měsíční časové omezení. Lze si také povšimnout bočního panelu s různými kategoriemi, které se týkají jednotlivých funkcionalit. K vytvoření rozvrhů je nutné mít vytvořen alespoň jeden kurz a jednoho lektora a pak můžeme vytvořit semestr (pololetí), ve kterém se kurz bude poskytovat a určíme čas a den daného kurzu v určeném semestru.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek - Sawyer detail kurzu seznam studentů (zdroj: (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023))

Tento systém splňuje všechny požadavky, avšak jak si lze povšimnout na obrázcích výše, jedná se o příliš komplexní systém viz Obrázek výše. Ovšem tento nedostatek lze přehlédnout díky skvělým funkcionalitám navíc. Velká výhoda tohoto systému je jednoduchý reporting systém, který je obsažen již v základním balíčku. Další nespornou předností je správa nákupů uživatelů (studentů a rodičů) přímo v systému nebo přes Marketplace, avšak prozatím jen v amerických dolarech. Tato funkcionalita zatím využívána nebude. (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023)

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek - Sawyer seznam kurzů (zdroj: (*Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023))

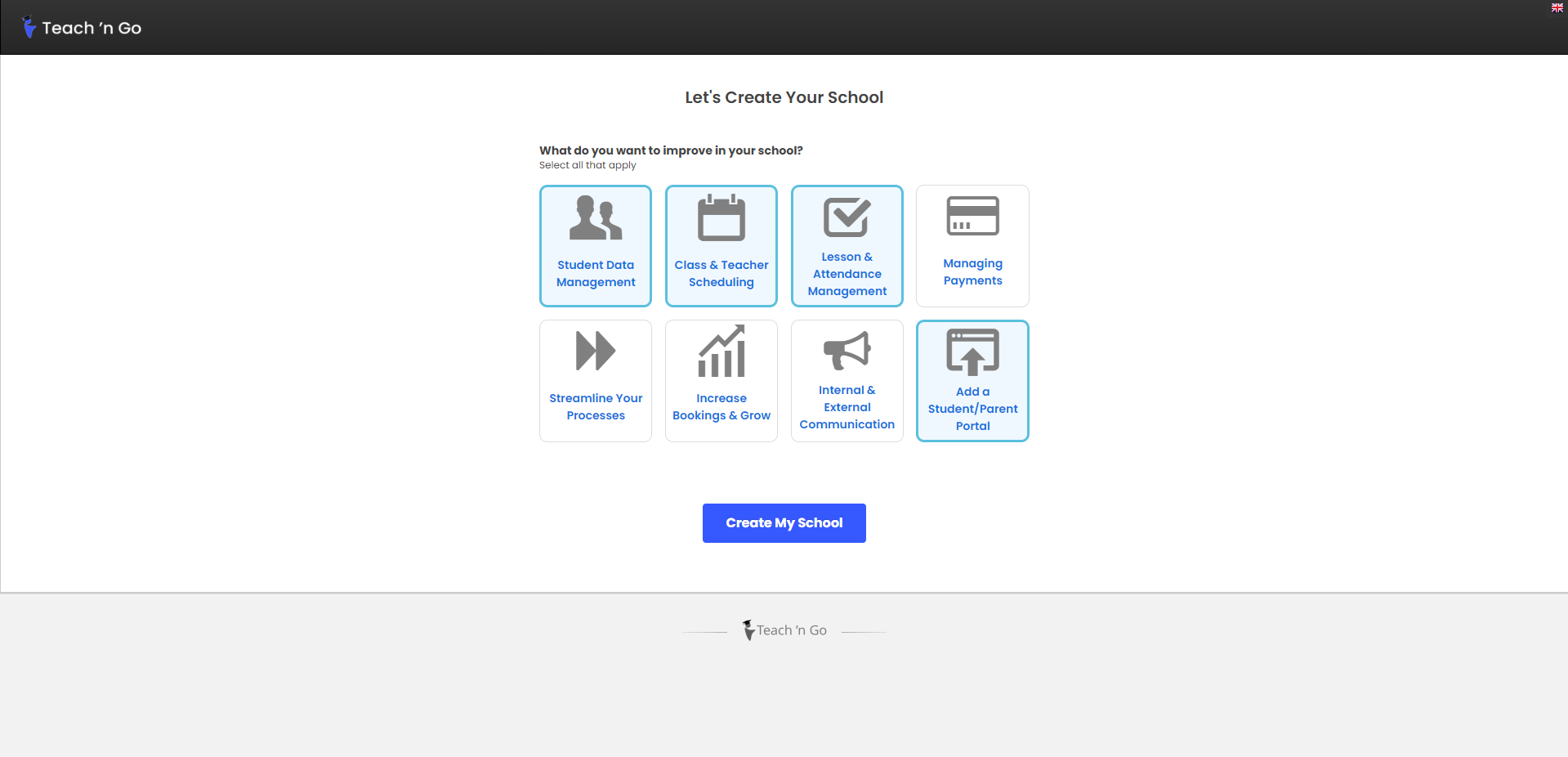
Z pohledu UI a UX je systém jednoduchý, přehledný a moderní, i responzivní. Systém je rozdělen na několik částí, každá část odpovídá dané funkcionalitě. I nezkušený uživatel je schopen porozumět systému, ale ne kompletně. Jediná nevýhoda je jazyk systému, lze ho používat jen v angličtině, ale jednotlivé kurzy apod. lze psát česky a v jakémkoliv dalším jazyce. Systém je plně responzivní.

Během tvorby kurzu nebyly objeveny žádné potíže. Postup byl poměrně jednoznačný. Tvorba kurzu probíhá pomocí několika stránkového formuláře. První krok je zadání jména kurzu, kategorií, určení barvy, vybrání, zda se bude jednat o online či prezenční kurz. Druhý krok je vyplnění volitelných detailů kurzu, jako například: popis, požadavky, dovednosti atd. Ve třetí fázi si určujeme, pro koho kurz je, věkové omezení, dětský nebo dospělácký kurz nebo oboje. V posledním kroku můžeme poskytnou ještě více detailů jako je známkování, zda budou domácí úkoly atd.

Celkový dojem z tohoto systému je velmi spokojený, splňuje všechny požadavky, které prozatím společnost vyžaduje. Avšak pokud by systém společnost chtěla zpřístupnit pro studenty je nutné myslet, že je k dispozici pouze anglicky. Pro interní účely je ovšem dostačující.

## Teach‘n go

Systém Teach’n go je mezinárodní systém pro správu kurzů, mimoškolních aktivit, ale i škol a univerzit. Tento systém nabízí správu uživatelů, rolí, kurzů, lekcí, docházky a další potřebné systémy, ale oproti předešlému řešení nabízí mobilní aplikaci jak na Android, tak i na iOS. Systém nabízí i několik jazyků od angličtiny po španělštinu, čínštinu, japonštinu a třeba i srbštinu. Celkem je dostupný v 16 jazykách. Tento systém je již používán ve více než 40 zemích po celém světě. (*Teach ’n Go - Modern School Management Software*, 2023)



Obrázek - Tvorba systému (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023))

Prvním krokem po registraci jako administrátor v tomto systému je výběr funkcionalit, které budeme chtít používat. Jak si lze povšimnout, je jich na výběr 8, avšak na interní účely společnost bohatě stačí 4 modře zvýrazněné funkcionality. Jedná se o správu uživatelů, kurzů, lekcí a docházky a také studentský/rodičovský portál.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek - Hlavní obrazovka (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023))

Na obrázku výše lze vidět podobu hlavní obrazovky po přihlášení uživatele. Vlevo se nachází postranní panel s důležitými kategoriemi jako jsou správa uživatelů, tříd atd. Nejdůležitější část je přehled lekcí, které se konají dnes, popřípadě uživatel může vybrat datum. Vpravo se nacházejí jednotlivé sekce, studentova statistika. Student může vidět kolik lekcí zameškal, zda dostal nějaké pochvaly nebo důtky, upozornění, narozeniny a odškrtávací seznam pro nejrůznější úkoly a poznámky.

Obsah obrázku stůl

Popis byl vytvořen automaticky

Obrázek - Teach'n go příklad rozvrhu (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023))

Na obrázku 14 lze vidět podobu rozvrhů, která je rovněž hlavní stránkou po přihlášení uživatele. Opět se dají barvy jednotlivých kurzů v rozvrhu upravit. Lze si znovu vybrat denní, týdenní nebo měsíční zobrazení. Rozvrhy jsou vytvářeny správcem systému. Rozvrhy lze filtrovat dle několika kategorií, jako i u předchozího systému.

Obsah obrázku text, snímek obrazovky, přenosný počítač

Popis byl vytvořen automaticky

Obrázek - Teach'n go příklad kurzu/třídy (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023))

Detail kurzu (třídy) je jednoznačný, nejdříve lze nalézt detaily třídy jako například kde se koná, v kolik se odehrává, kdo učí daný kurz, počet lekcí atd. Dále je možné si povšimnout seznamu lekcí, studentů poznámek a příloh atd. Kdykoli je možné lekce přidat, ale po vytvoření kurzu se lekce automaticky vygenerují. Když se lekce vytváří, tak je možné vyplnit jen pár informací, a to datum a čas konání, místo a vybrat učitele, což není optimální z hlediska materiálů pro danou lekci. Každá lekce se zabývá jiným tématem, a proto by bylo vhodné mít možnost lekci nazvat a napsat alespoň krátký popis.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek - Teach'n go příklad tvorby kurzu/lekce (zdroj: (*Teach ’n Go - Modern School Management Software*, 2023))

Pro vytvoření kurzu (třídy) je potřeba vyplnit formulář, který lze vidět na Obrázku 16. Kurz musí mít nějaký název, musí ho někdo učit, musí mít určenou místnost nebo online a taky musí být určen den v týdnu a hodina konání lekcí. Dále se mohou určit jednotlivé poplatky a výplata učitele. Daný formulář je přehledný, rozdělen do několika částí a celkově lze kurz vytvořit rychle a snadno.

Z pohledu UI a UX je systém přehledný a poměrně jednoduchý, avšak má zastaralejší design oproti předchozímu systému. Systém je plně responzivní až na pár detailů, jako třeba u navigační lišty, jakmile se jednotlivé prvky nevejdou tzv. odskočí na další řádek. Tento problém by mohl vyřešit například hamburger menu.

Celkově je systém přehledný a jednoduchý. Splňuje všechny základní požadavky a nabízí i zajímavé funkce navíc. Výhoda tohoto systému je mobilní aplikace a 10letá praxe na trhu. Další velkou výhodou je výběr jazyků, do budoucna budou i jazyky přibývat. Skvělý systém jen pro interní účely s možností rozšíření i pro účely externí. Jediná výtka tohoto systému je zastaralý design.

## Tabulka porovnání

V této části je zaměřeno na několik klíčových vlastností daných systémů i jakožto ceny. Klíčové vlastnosti jsou splnění všech požadavků, UI a UX, extra funkcionality, cena, dostupnost. Předchozí řešení jsou porovnána i se současným způsobem provozu společnosti a jsou určeny funkcionality navíc, které by bylo vhodné naimplementovat i do nového systému.

Tabulka - Porovnání systémů (zdroj: autor na základě (CODE Rebels, 2023; *Best Kids Activity Registration & Management Software | Sawyer Tools*, 2023; *Teach ’n Go - Modern School Management Software*, 2023))

|  |  |  |  |
| --- | --- | --- | --- |
| Vlastnosti | Sawyer | Teach’n go | Současný způsob provozu |
| **Splnění požadavků** | Ano | Ano | Ne |
| **UI/UX** | Jednoduchý, přehledný, responzivní | Jednoduchý, přehledný, téměř responzivní, zastaralý vzhled | Několik systémů, nepraktické |
| **Cena** | 30 000 CZK/rok\* | 21 000 CZK/rok\* | Cca 1 000 CZK/rok |
| **Dostupnost** | USA | 44 zemí | - |
| **Navíc** | MarketPlace, kupování přímo v aplikaci, marketing systém, reporting systém, support | Kupování přímo v aplikaci, marketing systém, reporting systém | - |
|  |  |  |  |

\*Po zaokrouhlení a převedení z měny USD u Sawyer systému a EUR u Teach’n go systému

Z výše uvedeného srovnání je poznatek takový, že řešení dostupná na trhu jsou poměrně drahá, nepodporují češtinu, a rozhodnutí takové, že je skutečně nejlepší pro společnost zvolit možnost vlastního řešení, které nebude tak drahé a splní všechny požadavky.

## Shrnutí dostupných řešení

V přechozí kapitole bylo zhodnoceno pár řešení dostupných na trhu. Byly následně zhodnoceny pomocí několika kritérií, a to, zda splňují všechny požadavky, z pohledu UI a UX, jak snadno lze vytvořit kurz, cena, jazyky a dostupnost a také jaké další funkcionality nabízejí. Z této kapitoly lze vybrat funkce, které jsou požadované. Poté bude vybráno několik funkcí, které tyto systémy nabízejí navíc a které lze použít i ve vlastním řešením a následně bude snaha o jejich implementaci i do nového systému.

# Životní cyklus projektu

Pro usnadnění a přehlednost vývoje systému byla zvolena metodika MMSP-AV neboli Metodika pro menší softwarové projekty – agilní vývoj. Tato metodika je rozšířením MMSP agilními metodami a přístupy. Původní metodika MMSP byla vytvořena lokalizací a customizací metodiky OpenUP (*Metodika MMSP-AV*, 2023). Životní cyklus může být definován jako postup vývoje systému, díky kterému dochází na základě předem definovaných požadavků k vytvoření systému jako takového (Cohn, 2006). Metodika MMSP-AV byla zvolena z důvodu detailního propracování, přehlednosti a snadné použitelnosti.

## Fáze životního cyklu projektu

Životní cyklus projektu se bude skládat ze 4 hlavních fázích. První fáze se nazývá Fáze Zahájení, která bude pojednávat o zahájení projektu, určení požadavků na daný systém, určení klíčových rolí, návrhu systému a jeho nejdůležitějších součástí a příprava testů. Druhá fáze, Fáze Rozpracování, bude pojednávat o prvotní tvorbě systému a jeho nejdůležitější součásti, plánování testů a manuální testování. Další fáze, Fáze Konstrukce, pojednává o samotné tvorbě systému a jeho součástí, provedení, tvorbě unit testů a implementaci řešení. Poslední fáze, Fáze Zavedení, se zaměřuje na implementaci a nasazení systému, provedení testů a celkové zhodnocení. Během každé fáze bude probíhat několik iterací, počet těchto iterací bude záležet na vývoji. (Cohn, 2006)

Fáze Zahájení se skládá z vize projektu, plánování projektu, požadavky na nový systém, rozdělení systému na části dle požadavků, role – autor, lektoři a klient budou zastupovat roli testerů pomocí testovacích případů, ostatní role zastupuje autor. (Arlow and Neustadt, 2007)

Fáze Rozpracování se skládá z návrhu architektury, začátku tvorby systému a daných součástí, tvorby testů a testování vytvořených částí. (Arlow and Neustadt, 2007)

Fáze Konstrukce se skládá z kompletní tvorby systému, implementace systému, tvorby unit testů, manuálního testování. (Arlow and Neustadt, 2007)

Fáze Zavedení se skládá z implementace a nasazení systému, provedení všech testů, nalezení nových požadavků a celkové shrnutí postupu. (Arlow and Neustadt, 2007)

## Role projektu

Všechny role pokryje autor, a navíc během testování klient a lektoři pokryjí role testerů. Klient a pan vedoucí také zaujme roli zainteresované strany ve všech fázích, což znamená, že budou pomáhat při identifikaci a detailní vymezení požadavků a také budou zasahovat do návrhu uživatelských obrazovek.

## Fáze zahájení

Fáze zahájení je první fáze životního cyklu vývoje projekt, jejíchž hlavním úkolem je úspěšně zahájit projekt porozumět a identifikovat požadavky na vyvíjený systém, identifikovat klíčové procesy a funkce, které by měl systém obsahovat a určit rizika, která by mohla mít negativní dopad na vývoj.

### Vize projektu

Projekt si klade za cíl vytvoření systému pro poskytování kurzů a tím zlepšit komunikaci a práci učitelů a studentů a zjednodušit správu kurzů administrátory. Pokud by se nový systém neimplementoval, tak s přibývajícím množstvím studentů a kurzů by se současný systém stal velmi nepřehledným až téměř nepoužitelným. Nový systém umožní přehlednější správu všech potřebných částí systému pro poskytování kurzů a snazší komunikaci mezi uživateli. Jedná se o CRM systém pro studenty a lektory, z důvodu řízení komunikace a interakcí mezi společností a klienty (studenti).

### Požadavky

Tato část se zabývá požadavky z pohledu programového. V předchozí kapitole byly požadavky rozebrány z pohledu uživatelského dle požadavků klienta. Tyto požadavky se zaměřují na pokrytí všech důležitých částí systému a funkcionalit. Stejně jako v přechozí kapitole jsou rozděleny na funkční a nefunkční.

#### Funkční požadavky:

Autentizace uživatele:

* Přihlásit se
* Obnovit zapomenuté heslo
* Odhlásit se
* Správa profilu

Správa uživatelů:

* Správa uživatelů
* Správa rolí
* Správa pravomocí
* Registrovat uživatele

Firma:

* Vlastní doména, správa domény
* Možnost více škol/poskytovatelů
* Správa škol/poskytovatelů

Výuka:

* Správa tříd
* Správa kurzů
* Správa lekcí
* Správa zápisů
* Komunikace mezi uživateli
* Správa rozvrhů

Docházkový systém:

* Správa docházek
* Možnost zapisovat
* Správa náhrad

Hlavní obrazovka:

* Týdenní rozvrh
* Úkoly, poznámky
* Zapsané třídy/kurzy
* Komunikace s uživateli
* Materiály – skeny učebnic, naprogramované kódy…
* Další kontakty
* Marketplace
* Upozornění – novinky, úkoly, zprávy ve třídách…
* Nastavení

#### Nefunkční požadavky:

Z pohledu použitelnosti bude nutné mít systém v českém jazyce a popřípadě anglicky, ale do budoucna rozhodně ve více jazykách. Z pohledu bezpečnosti systému každý uživatel může vykonat takové činnosti, které má povolené a každý uživatel musí být autorizován a autentifikován. Systém musí být spolehlivý a v případě výpadku musí být co nejrychleji obnoven (předchozí verze, odstranění funkcionality…). Systém by měl být rychlý a měl by zvládnout i několik desítek aktivních uživatelů. Také se u systému předpokládá rozšiřování o potřebné funkcionality pro firmy, studenty a lektory.

Z pohledu UI a UX musí být systém responsivní, přenositelný, dále musí mít dobře zpracované uživatelské rozhraní a vzhled a musí poskytnout kvalitní uživatelský zážitek a zkušenost (přehledný, snadno použitelný…).

Systém musí splňovat GDPR zákon. Z pohledu dokumentace je vhodné zdokumentovat jednotlivé kroky při správě daných entit, jako např.: tvorba kurzů, zapisování docházky, tvorba a zápis tříd atd.

Prozatím žádná obchodní pravidla klient nevyžaduje, ale do budoucna lze vymyslet spoustu různých pravidel.

### Plán projektu

Projekt bude vytvářet autor sám za pomocí připomínek klienta a pana vedoucího. Klient bude pouze systém testovat. Každý cyklus se bude skládat ze čtyř částí. První částí bude plánování tvorby dané části systému či funkcionality. V druhé části půjde o implementaci a tvoření části systému. Ve třetí části půjde o otestování části systému a souvisejících funkcionalit, kde testovat bude autor a poté klient. V poslední části proběhne celkové zhodnocení testování a implementace a shrnutí všech poznatků a případné zařazení nových požadavků do plánu další iterace dle důležitosti. Jednotlivé cykly jsou vypsány níže, odpovídají kategoriím požadavků:

1. Tvorba správy uživatelů
2. Autentizace uživatele
3. Správa firmy
4. Tvorba částí výuky
5. Docházkový systém
6. Hlavní obrazovka

## Fáze rozpracování

V této fázi nejprve proběhne analýza požadavků, která pomůže k upřesnění všech nejasností. Dále se zaměřuje na identifikaci objektů a jejich vztahů. A to jsou tedy:

* Uživatel, který má roli a role má práva, popřípadě uživatel může mít sám práva.
* Poskytovatel kurzů, který má na starosti své kurzy.
* Lekce, ze kterých se skládá daný kurz.
* Třída, kde jsou zapsáni studenti (Zápis) a vyučuje je lektor a používá osnovy dle daného kurzu.
* Docházka, kam zapisuje lektor z dané třídy a zapisuje daného studenta.
* Rozvrh, který bude generován automaticky a bude určovat, která třída kdy bude, jakou lekci a daného lektora (i náhradního – nutně ručně) a popřípadě budou určeny náhradní lekce pro chybějící studenty

Dále proběhne detailní popis komponent, vývoj částí a systému jako takového a jeho testování, budou také probíhat iterace dle kategorií požadavků a na závěr výběr prostředí a implementace.

Nejdříve proběhne v každé iteraci návrh a tvorba endpointu na backendu pro správu dat a poté návrh a tvorba frontendu a uživatelských obrazovek. Následně bude nutné dané části propojit dle potřeby. Na konci každé iterace proběhne otestování dané části a celkové shrnutí dané iterace a v případě nových požadavků, proběhne jejich zařazení mezi požadavky dané iterace, nebo vytvoření iterace nové. Viz další kapitola.

Na závěr proběhne celkové shrnutí fáze a zapsaní případných nově vzniklých funkcionalit a požadavků do další fáze, pokud budou vhodné pro další fázi. Níže jsou vypsány jednotlivé iterace.

### Tvorba správy uživatelů

Tato iterace je zaměřena na:

* Správa uživatelů
* Správa rolí
* Správa pravomocí
* Registraci uživatelů

Pro každou část je nutné vytvořit vlastní endpoint a poté navrhnout a vytvořit frontend. Uživatele bude registrovat administrátor.

### Autentizace uživatele

Tato iterace je zaměřena na:

* Přihlásit se
* Obnovit zapomenuté heslo
* Odhlásit se
* Správa profilu

Všechny dané součásti spolu souvisí a souvisí s předchozí iterací. Bude potřeba několik endpointů. Dále proběhne návrh uživatelských obrazovek a jejich tvorba.

### Správa firmy

Tato iterace je zaměřena na:

* Vlastní doména, správa domény
* Možnost více škol/poskytovatelů
* Správa škol/poskytovatelů

Pro každou část je nutné vytvořit vlastní endpoint a poté navrhnout a vytvořit frontend. V případě požadavku na vlastní doménu je důležité nezapomenout na možnost subdomén od autora (hlavního systému) anebo subdomén poskytovatele.

### Tvorba částí výuky

Tato iterace je zaměřena na:

* Správa tříd
* Správa kurzů
* Správa lekcí
* Správa zápisů
* Komunikace mezi uživateli
* Správa rozvrhů

Pro každou část je nutné vytvořit vlastní endpoint a poté navrhnout a vytvořit frontend.

### Docházkový systém

Tato iterace je zaměřena na:

* Správa docházek
* Možnost zapisovat
* Správa náhrad

Pro obě části bude sloužit jeden endpoint. Následně se navrhne a vytvoří uživatelské obrazovky a rozšíří se uživatelské obrazovky třídy pro snadný zápis.

### Hlavní obrazovka

Tato iterace je zaměřena na:

* Týdenní rozvrh
* Úkoly, poznámky
* Zapsané třídy/kurzy
* Komunikace s uživateli
* Materiály – skeny učebnic, naprogramované kódy…
* Další kontakty
* Marketplace
* Upozornění – novinky, úkoly, zprávy ve třídách…
* Nastavení

Dané části budou potřebovat několik endpointů a obrazovek. Tato iterace se také bude zaměřovat na propojení všech součástí, nechť všechny dané požadavky souvisí s předchozími částmi.

### Iterace testování

Poslední iterace se zaměří na otestování jednotlivých částí a funkcionalit a celkové shrnutí. V případě nových požadavků proběhne jejich zařazení do nové iterace nebo do další fáze, dle důležitosti a závažnosti požadavku.

## Fáze konstrukce

Po navrhnutí, vytvoření, propojení a implementaci všech důležitých částí systémů bude potřeba navrhnout a vytvořit nově vzniklé funkcionalit, které můžou vzniknout během tvorby systému nebo díky nedokonalostem mezi požadavky. Následně je potřeby vytvořit unit testy a následně systém otestovat. Nedílnou součástí této fáze je vytvoření funkční verze systému, která respektuje dané návrhy z předchozích fází. (Arlow and Neustadt, 2007)

Díky složitosti systému a jeho součástí a pro spojení jednotlivých součástí a funkcionalit daných součástí vznikají nové funkcionality, pro zajištění plynulosti systému.

Dále implementace systému na dané prostředí – Render, MongoDB, Brevo (Sendinblue)

## Fáze zavedení

Provedení všech testů, celkové shrnutí a hodnocení, vývoj a návrh nově vzniklých funkcionalit, build, reporting, údržba. Zavedení subdomén.

# Návrh aplikačního systému

Tato kapitola se zaměřuje na návrh aplikace a představení použitých technologií MERN stacku. Návrh aplikace se bude skládat z několika částí to jak z pohledu API a databáze, tak i z pohledu uživatelského rozhraní. Návrh databáze bude pojednávat o základní představě databáze a jejích kolekcí a relací mezi nimi. Také bude zahrnovat prvotní modely kolekcí. Návrh uživatelského rozhraní je podstatnou fází návrhu webové aplikace, protože zahrnuje vytvoření vizuálního designu a uživatelské zkušenosti aplikace. Jedná se tedy o webovou aplikaci v architektuře klient-server. Tato architektura představuje výpočetní model, ve kterém server hostí, dodává a spravuje většinu zdrojů a služeb požadovaných klientem. Klienti poskytují rozhraní, které umožňuje uživateli počítače požadovat služby serveru a zobrazovat výsledky, které server vrací. Servery čekají na požadavky od klientů a poté na ně reagují. (Terra, 2023)

Proces návrhu uživatelského rozhraní bude modelován pomocí wireframingu. Wireframing je proces vytváření základního rozvržení aplikace, včetně umístění obsahu, funkcí a navigačních prvků. Je to model, který obsahuje pouze základní rozložení stránky pomocí „drátování“.

Pro vytvoření aplikace byl zvolen MERN stack, a to z několika důvodů. Za prvé MERN Stack je JavaScriptový stack, programuje se jen v jednom programovacím jazyku (je možnost použití Typescriptu, pro datovou typovost), což je velká výhoda. Další důvod je jednoduchost a rychlost vývoje fullstackových webových aplikacích. Velkou výhodou jsou známé technologie v MERN stacku. MERN stack je založen na MVC architektuře, což znamená je velmi snadné a rychlé naučit se a vytvořit danou aplikaci. Nelze opomenout flexibilitu MERN stacku a jeho součástí, jelikož jsou všechny open source, to zabraňuje problémům s licencemi (Jha, 2021). MERN stack má spoustu dalších výhod, ale teď je čas se zaměřit na návrhový vzor MVC.

## Návrhový vzor MVC

Tato podkapitola se zabývá vysvětlením návrhového vzoru MVC, jeho principy a použití. MVC návrhový vzor se dělí na tři části a to jsou:

* M jako Model – spravuje data, odpovídá na požadavky na stav dat, odpovídá na instrukce z controlleru.
* V jako View – vykresluje model a jeho data na vhodnou formu pro snadné zobrazení, práci a použitelnost pro uživatele, může být několik view pro jeden model.
* C jako Controller – dostává vstup od uživatele a reaguje na něj tak že posílá požadavky na model, řídí komunikaci mezi view a modelem. (Jha, 2021)

Rozděluje aplikaci na byznys logiku a jednotlivá zobrazení. Toto dělení usnadňuje snazší práci a lepší údržbu a přehlednost. Většina dnešních frameworků je prezentována s tím, že je založena na principu MVC. MERN stack nevyužívá MVC typickým způsobem, protože zahrnuje jak server, tak prohlížeč. Je možné docílit klasického MVC pomocí server-side šablon a posílat do prohlížeče jen HTML kód a pak prohlížeč bude představovat View vrstvu. (Jha, 2021)

V MERN stacku je možné jednotlivé části MVC přirovnat k jednotlivým technologiím. Každá technologie potom odpovídá za dané funkce v MVC a má potom i vlastnosti dané části. Pro komunikaci s MongoDB se používá knihovna mongoose, více v sekci 5.3 Databáze. Jednotlivé části a odpovídající technologie jsou (Jha, 2021):

* Knihovna mongoose poté odpovídá Modelu, a tedy datové části aplikace a přistupuje k MongoDB, kde jsou všechna data uložená. Vyřizuje požadavky na stav data a odpovídá controllerům z ExpressJS serveru.
* ReactJS pokrývá část View a obsahuje jednotlivé stránky a komponenty, potřebné ke správnému a srozumitelnému zobrazení dat pro uživatele.
* ExpressJS server a Node.js odpovídají Controlleru a zařizují komunikaci mezi View a Modelem, tedy mongoose a Reactem. Také překládají data mezi mongoose a Reactem, tak aby vzájemně byly data správně zpracovaná, zobrazená a uložená. (Jha, 2021)

Lze si povšimnout další nesporné výhody MERN stacku, jednotlivé technologie odpovídají daným částem architektury. Záleží na každém vývojáři, do jaké míry se bude vzorem MVC řídit, je možné, že ne všechny části vždy budou součástí každé aplikace, obzvláště, jedná-li se o jednoduché aplikace.

## ExpressJS server a Rest API

ExpressJS server se skládá ze dvou částí, a to controllerů a rout. Controllery obsahují funkce pro komunikaci s databází. Každý controller obsahuje čtyři základní funkce, a to na vytváření dat, získávání dat, úpravu dat a mazání dat. Jednotlivé funkce se následovně upravují dle potřeb aplikace. Může být několik funkcí na získávání dat atd. Každá část aplikace (třídy, kurzy, lekce, uživatelé atd.) má vlastní controller, který řídí komunikaci s danou částí aplikace. Každý název funkce v controlleru bude začínat příslušeným typem, pokud se jedná o získáváni dat tříd tak například:

Výpis - Ukázka části controlleru lekcí (zdroj: autor)

1. const asyncHandler = require('express-async-handler')
2. const Lesson = require('../models/lessonModel');
3. //...
4. const updateLesson = asyncHandler(async (req, res) => {
5. const lesson = await Lesson.findById(req.params.id)
6. if(!lesson) {
7. res.status(400)
8. throw new Error("Lesson not find")
9. }
10. const updatedLesson = await Lesson.findByIdAndUpdate(req.params.id, req.body, {
11. new: true,
12. })
13. res.status(200).json(updatedLesson)
14. })
15. //...
16. module.exports = {
17. getLessons,
18. setLesson,
19. updateLesson,
20. deleteLesson
21. }

ExpressJS také obsahuje routy, které odpovídají cestám pro React. Každá routa je značená /api/... a poté s názvem dané části aplikace, např.: /api/lessons. Název dané routy odpovídá controlleru, který řídí komunikaci mezi danou částí aplikace a databáze. Každý controller má svoje routy a ty slouží k rozlišení typu požadavku z front endu.

Výpis - Ukázka rout lekcí (zdroj: autor)

1. const express = require('express')
2. const router = express.Router()
3. const { getLessons, setLesson, updateLesson, deleteLesson } = require('../controllers/lessonController')
4. …
5. const { authenticate, authorize } = require("../middleware/authMiddleware")
6. …
7. router.use(authenticate)
8. …
9. router.route("/").get(authorize("lessonGet"), getLessons)
10. .post(authorize("lessonCreate"), setLesson)
11. router.route("/:id").delete(authorize("lessonDelete"), deleteLesson)
12. .put(authorize("lessonUpdate"), updateLesson)
13. …
14. module.exports = router

Jak lze vidět routy jsou jednoduché konstrukty, které směrují komunikaci na danou část controlleru, které odpovídá požadavek. Server určuje klíčové routy (např.: /api/lessons), na které předává jednotlivé detailní routy.

## Databáze

Aplikace bude ukládat data do NoSQL databáze zvané MongoDB. Rozdílů mezi NoSQL databázemi a SQL databázemi je hned několik. Klasické databáze používají tabulky a jednotlivé řádky pak představují data kdežto u MongoDB se data ukládají do kolekcí a jednotlivé dokumenty představují jednotlivé záznamy dat. Druhým rozdílem jsou vztahy mezi daty. U SQL databází se řeší, jestli data obsahují odkaz na jiné data již v databázi, pokud ano je mezi nimi vztah. U NoSQL databází se toto nepoužívá, databáze se navrhuje na základě dotazů, které na ní směřují. Modely NoSQL databází mají několik způsobů, jak vymodelovat databázi, například pomocí síťového grafu (MongoDB, 2023b). Níže lze viděl model databáze aplikace.

Obsah obrázku diagram, Plán, text, řada/pruh

Popis byl vytvořen automaticky

Obrázek - Model databáze (zdroj: autor)

U NoSQL databází se neřeší, jestli jeden dokument odkazuje na mnoho jiných nebo zda mnoho jiných odkazuje na daný dokument. Neexistují vztahy 1:1, 1:N nebo M:N, pouze se ukládají odkazy na jiné dokumenty do pole hodnot k danému dokumentu. Z toho vyplývá, že NoSQL databáze poskytují řadu výhod včetně flexibilních datových modelů, horizontálního škálování, velmi rychlých dotazů a snadného použití pro vývojáře. Existuje několik typů NoSQL databází a tato webová aplikace bude používat dokumentové databáze. (MongoDB, 2023b)

Pro správnou komunikaci mezi controllerem a MongoDB používám knihovnu mongoose. Jedná se o ODM knihovnu neboli Object Data Modeling knihovnu. Spravuje vztahy mezi daty, poskytuje ověřování schémat a používá se k překladu mezi objekty v kódu a reprezentaci těchto objektů v MongoDB. Pomocí mongoose se vytváří schémata, která slouží jako vzor, proto jak data budou vypadat. Zastupují část modelu v MVC. (mongoose, 2023)

## Uživatelské rozhraní

Pro návrh uživatelského rozhraní byl použit takzvaný drátěný model neboli wireframe, který představuje nejzákladnější kostru webu a naznačuje jeho strukturu, funkcionalitu, rozložení prvků a charakterizuje jednotlivé sekce stránky. Je možné se dívat na drátěný model jako na jakýsi nákres toho, jak bude web vypadat. Dále je zobrazeno rozložení následujících stránek a to jsou: úvodní stránka, detail lekce a formulář pro tvorbu nové lekce.

Obsah obrázku text, zástrčka, snímek obrazovky, diagram

Popis byl vytvořen automaticky

Obrázek - Wireframe úvodní obrazovky (zdroj: autor)

Úvodní stránka bude začínat uvítacím textem a nad ním se bude nacházet navigační panel s nejdůležitějšími odkazy. Pod touto sekcí bude mít každý uživatel na úvodní obrazovce seznam tříd, ve kterých je zapsaný a následně, jestliže má nějaká třída lekci, která se koná ve stejný den jako se uživatel nachází v aplikaci, zobrazí se tlačítko „Dnešní lekce“. Uživatel také bude mít možnost překliknout se do detailu třídy, kde najde spoustu dalších užitečných informací.

Obsah obrázku text, snímek obrazovky, číslo, Písmo

Popis byl vytvořen automatickyDetail lekce bude rozdělen na dvě části a to popis, osnovy a docházka a druhá část bude videohovor. Takto detail lekce uvidí lektor a admin, pokud se jedná o studenta, ten neuvidí docházky a odkaz na materiály a místo osnov uvidí jen krátký popis lekce.

Obrázek - Wireframe formuláře pro tvorbu lekce (zdroj: autor)

Obrázek - Wireframe detailu lekce (zdroj: autor)

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automatickyVýše lze vidět podobu formuláře pro tvorbu lekce. Jedná se o jednoduchý a přehledný formulář, který se skládá z několika polí. Návrh aplikace a aplikace samotná se mění během vývoje neustále, je možné najít odlišnosti mezi návrhem a skutečností. Je to z důvodu časté komunikace s klientem a jeho měnicím se vkusem a představami pro aplikaci.

## Použité technologie

MERN stack je populární sada technologií používaných k vývoji tzv. full-stack webových aplikací. Zkratka MERN je akronymem pro kolekci technologií a to jsou: MongoDB, Express.js, React a Node.js. Každá z těchto technologií slouží k jedinečnému účelu v procesu vývoje webu a společně poskytují výkonnou sadu nástrojů pro vývojáře k vytváření škálovatelných a efektivních webových aplikací. Jednou z hlavních výhod MERN stacku je, že umožňuje vývojářům používat jediný jazyk a to JavaScript, během celého vývojového procesu, od front-endu až po back-end. To snižuje složitost procesu vývoje a usnadňuje vytváření a údržbu webových aplikací. (MongoDB, 2023c)

Další výhodou je dostupnost široké škály knihoven a frameworků třetích stran, které lze použít s MERN stackem. Například Redux lze použít s Reactem pro správu stavu a Passport.js lze použít s Express.js pro ověřování a autorizaci. Lze i velmi snadno nějakou technologii nahradit jinou, například React lze nahradit technologií Angular a přejít na MEAN stack. (MongoDB, 2023c)

#### MongoDB

MongoDB je dokumentově orientovaná databáze NoSQL, která ukládá data do dokumentů podobných JSON. Dané dokumentu jsou pak ukládány do jednotlivých kolekcí. Na rozdíl od tradičních relačních databází MongoDB nevyžaduje pevné schéma, díky čemuž je flexibilnější a přizpůsobivější měnícím se požadavkům na data v porovnání s klasickými relačními databázemi. Poskytuje také vysokou škálovatelnost a dostupnost, což aplikacím umožňuje zpracovávat velké objemy dat.

Jak již bylo psáno MongoDB ukládá data do BSON dokumentů, který můžou být určeny schémata. Schémata v MongoDB definují strukturu dokumentů v kolekcích. Schémata můžou obsahovat informace, jako jsou názvy polí, datové typy, ověřovací pravidla a indexy. Schéma může být vynuceno MongoDB, aby bylo zajištěno, že data jsou konzistentní a platná. Nicméně se nejedná o tzv. integritní omezení v relačních databázích. (MongoDB, 2023a). Pro práci s MongoDB jsem zvolil knihovnu mongoose.

Výpis - Ukázka MongoDB schématu (zdroj: autor)

1. const courseSchema = mongoose.Schema(
2. {
3. title: {
4. type: String,
5. required: [true, "Course should have some title"],
6. },
7. description: {
8. type: String,
9. },
10. users: [{
11. type: mongoose.Schema.Types.ObjectId,
12. ref: "User",
13. }],
14. lessons: [{
15. type: mongoose.Schema.Types.ObjectId,
16. ref: "Lesson",
17. }]
18. }, {
19. timestamps: true,
20. }
21. )

Více je možné se dozvědět v následující kapitole, např. jak aplikovat a použít MongoDB a knihovnu mongoose. Také lze zjistit, jak vytvářet schémata, datové typy MongoDB, základní požadavky pro práci s MongoDB a podobně.

#### ExpressJS

Express.js je back-endový webový framework, která běží na běhovém prostředí Node.js. Poskytuje sadu nástrojů a funkcí pro vytváření webových aplikací a rozhraní API (v případě nového systému REST API, více viz kapitola 5), jako je směrování, middleware a zpracování chyb. Express.js je známý svou jednoduchostí a flexibilitou, která umožňuje vývojářům přizpůsobit chování a strukturu aplikací podle jejich potřeb. Express.js poskytuje širokou škálu funkcí a funkcionalit, které umožňují vývojářům vytvářet robustní a škálovatelné webové aplikace (Express, 2023). Některé z klíčových funkcí Express.js zahrnují:

* Routing – Express.js poskytuje jednoduchý a flexibilní způsob, jak definovat trasy a mapovat je na konkrétní funkce nebo obslužné rutiny. To umožňuje vývojářům vytvářet čistý a organizovaný kód, který se snadno udržuje a upravuje.
* Middleware – obsahuje middleware framework, který umožňuje vývojářům přidávat funkce do jejich aplikací. Middlewarové funkce lze použít k úpravě požadavků a odpovědí, ověření vstupu, ověření uživatelů a další.
* Error handling – nabízí robustní framework pro zpracování chyb, který umožňuje vývojářům zpracovávat chyby konzistentním a předvídatelným způsobem. To pomáhá zlepšit spolehlivost a stabilitu webových aplikací.
* Statické soubory – poskytuje vestavěný middleware pro poskytování statických souborů, jako jsou obrázky, CSS soubory a JavaScript skripty. To zjednodušuje proces poskytování statických aktiv a zlepšuje výkon webových aplikací.
* Šablony – podporuje řadu šablon, jako jsou EJS, Handlebars a Pug. To umožňuje vývojářům snadno vytvářet dynamická zobrazení pro jejich webové aplikace. (Express, 2023)

#### ReactJS

React je front-endová knihovna napsaná v jazyce JavaScript vyvinutá společností Facebook (Meta). Umožňuje vytvářet dynamická a citlivá uživatelská rozhraní rozdělením aplikace na opakovaně použitelné komponenty. Virtuální DOM (Document Object Model) v knihovně React zajišťuje vysoký výkon tím, že minimalizuje počet skutečných manipulací DOM potřebných pro vykreslování, což z něj činí ideální volbu pro vytváření složitých a interaktivních uživatelských rozhraní. Proto tato technologie bude použita pro tvorbu uživatelského rozhraní aplikace (Meta, 2023)

#### Node.js

Node.js je běhové prostředí JavaScriptu. Umožňuje vývojářům psát kód na straně serveru v JavaScriptu, což usnadňuje vytváření kompletních webových aplikací pomocí jediného jazyka. Node.js poskytuje vysoký výkon a škálovatelnost, díky čemuž je vhodný pro vytváření rozsáhlých real-time webových aplikací.

Společně tyto technologie poskytují kompletní sadu technologií pro vytváření full-stack webových aplikací. MongoDB poskytuje flexibilní a škálovatelné databázové řešení, zatímco Express.js a Node.js umožňují skriptování na straně serveru a back-endový vývoj. React poskytuje dynamické a interaktivní uživatelské rozhraní a zároveň poskytuje výkon a škálovatelnost na straně front-endu. (MongoDB, 2023c)

#### SASS

Jedná se o výkonný preprocesorový skriptovací jazyk, který rozšiřuje CSS. Zkratka SASS znamená Syntactically Awesome Style Sheets neboli Syntakticky úžasné šablony stylů. SASS vám umožňuje psát kód CSS efektivněji tím, že nabízí spoustu dalších funkcí, jako jsou proměnné, vnoření, funkce, mixiny, a další. Skvělou funkcí SASSu je spojování několika šablon do jedné, což nám umožňuje psát šablony modulárnějším a udržitelnějším způsobem Veškerý kód v SASSu se zkompiluje do CSS, které lze ihned použít na webových stránkách. (Sass team, 2023)

#### Nodemailer

Nodemailer je oblíbená open-source knihovna pro Node.js, kterou vývojáři používají pro odesílání e-mailů z jejich aplikací. Poskytuje jednoduché a snadno použitelné rozhraní pro posílání e-mailů a podporuje širokou škálu emailových služeb a protokolů, včetně nejdůležitějších jako jsou SMTP, POP3 a IMAP. Pomocí Nodemaileru lze posílat emaily textové nebo ve formátu HTML, lze posílat e-maily s přílohami, nastavovat hlavičky a šablony a reagovat na emailové události, jako jsou např.: oznámení o doručení nebo chyby. Slouží k odesílání transakčních emailů, marketingových emailů, newsletterů a mnoho dalšího (Reinman, 2023). Tato technologie pokrývá část komunikace s uživateli.

# Implementace aplikace

Tato fáze se zaměřuje na podrobnou implementaci aplikace a jednotlivých součástí, které již byly zmíněny v předchozích kapitolách. Jednotlivé součásti aplikace budou rozděleny do individuálních kapitol a podkapitol. Aplikace bude sloužit ke správě kurzů programování a emailové komunikaci se studenty. Správu kurzů lze rozdělit na několik částí a správa kurzů jako takových, správa lekcí, ze kterých se jednotlivé kurzy skládají, správa docházky a rozvrhů a správa tříd, do kterých jsou studenti přiřazováni.

Aplikace bude fungovat podobně jako školní systém, vytvoří se uživatelé a přiřadí se jim daná role, a to student nebo lektor, popřípadě admin. Následuje další krok, a to tvorba kurzu, který představuje předmět a skládá se z lekcí, poté se vytvoří třída, do které se přiřadí studenti, učitelé a kurz. Po vytvoření třídy se vygeneruje docházka a rozvrhy. Jakmile se vše vygeneruje následuje již jen učení a popřípadě emailová komunikace.

Hlavní cílem aplikace prozatím je správa docházky a uživatelů a komunikace s nimi. Ostatní části jsou řešeny a plánovány po zavedení a odzkoušení hlavních cílů lektory a adminy, přesto však probíhá práce i na ostatních částí, aby byl poté vývoj systému rychlejší a snazší.

## Správa uživatelů a profil

Základní část této aplikace je správa uživatelů. Nejdříve admin musí uživatelé vytvořit a přiřadit jim role. Možnosti rolí jsou student, lektor a admin. Avšak je možnost jakoukoli roli vytvořit nebo upravit a přiřadit jim potřebné vlastnosti. Každý student se může posléze přihlásit a zobrazit si svůj profil a upravit a doplnit potřebné informace jako například telefonní číslo, další jména apod.

Admin má možnost zobrazit si seznam všech uživatelů, upravit profil, kteréhokoliv z nich. Admin také může mazat nepotřebné uživatele a přiřazovat je do tříd a zapisovat jim docházku nebo zapsat náhradní lekce pro studenty a náhrady za lektory. Admin také registruje uživatele. Každý uživatel musí mít unikátní email.

## Kurzy a lekce

Další částí aplikace jsou kurzy a lekce. Každý uživatel má možnost zobrazit si kurzy a lekce, ze kterých se kurzy skládají a jejich podrobnosti. Kurz obsahuje důležité informace a seznam lekcí, které se pak učí ve třídě. Lekce obsahují osnovy a materiály a pořadí, tak aby učitelé měli všechny potřebné informace a prostředky k úspěšnému odučení dané hodiny. Kurzy déle obsahují informaci o daném semestru, ve kterém se vyučuje. Kurzy lze vidět všechny, ale nejdříve se vypisují kurzy z aktuálního semestru. Kurzy spadají pod daného poskytovatele.

## Třídy

Pro všechny uživatele se zobrazuje seznam tříd, ve kterých jsou zapsány. Následně si uživatelé mohou zobrazit detail dané třídy a zobrazit si seznam lekcí, svou docházku a rozvrh. V detailu třídy mohou studenti nalézt kontakt na lektora v případě dotazů mimo lekci. Pokud student chyběl na nějaké lekci a potřebuje náhradní, v seznamu tříd se mu objeví daná lekce objeví, ale bude zvýrazněna jako náhradní.

O správu tříd se stará admin. Třídy admin vytváří, upravuje, maže a přiřazuje studenty přes zápis, lektory a kurzy a také zařizuje náhradní lekce pro studenty a náhrady za lektory. Kurz obsahuje seznam lekcí a jejich pořadí v kurzu a pokud admin vytvoří třídu s přiřazeným kurzem, tak se docházka a rozvrh vytvoří automaticky s názvem třídy a lekce. Pokud ovšem admin kurz nepřiřadí, tak se rozvrh a docházka vytvoří pouze s názvem třídy.

## Docházka a rozvrhy

Každý uživatel, který je zapsán v nějaké třídě si může zobrazit vlastní docházky a rozvrhy. Docházky a rozvrhy se generují automaticky po vytvoření třídy, jak již bylo zmíněno v předchozích kapitolách. Rozvrhy obsahují název dané třídy a lekce a také čas a datum, kdy se konají dané lekce. Na rozdíl od rozvrhů docházky rozšiřují dané záznamy o seznamy studentů, kteří se lekce zúčastnili a lektory, kteří danou lekci odučili a zapsali. Docházky a rozvrhy budou ukládány do databáze jako dva dokumenty a jednotlivé hodnoty budou získávány dle potřebných parametrů.

## Admin správy

Oproti běžným uživatelům a již zmíněným možnostem, má admin ještě několik správ na starosti. Jedná se o správu rolí, pravomocí a emailové komunikace. Správa rolí je možnost vytvářet role nové pomocí přiřazení potřebných práv a názvu role. Poté stačí vybrat uživatele a přiřadit jim novou roli. Admin může taky role mazat a upravovat.

Další důležitou součástí je komunikace se studenty a ta se provádí pomocí emailů. Lze posílat email individuálně anebo skupinám studentů dle tříd. Celou správu emailů má na starosti admin. Každý admin musí spadat pod určitého poskytovatele.

## Databáze

Každá z částí aplikace má svoji vlastní kolekci a schéma. V každé kolekci jeden dokument představuje jeden záznam dat, jako například jedna lekce a její data představují jeden dokument. Takto to platí pro všechny části aplikace.

## Rozdělení kódu

Tato kapitola popisuje konkrétní implementaci kódu a jeho rozdělení na části. Jako první se kód rozdělí na dvě základní části, a to backend a frontend. Část backendu se bude skládat z controllerů, rout, modelů a konfigurace serveru a připojení k databázi. Také se bude na backendu nacházet konfigurace emailu a potřebné routy a controller. Část frontendu se bude dělit na komponenty, stránky a část zaměřující se na komunikaci se serverem, a to jsou features, které se dělí na services a slices.

Výpis - Struktura kódu (zdroj: autor)

+---backend

| +---config

| +---controllers

| +---emailConfig

| +---middleware

| +---models

| \---routes

+---frontend

| +---build

| | \---static

| | +---css

| | \---js

| +---public

| \---src

| +---app

| +---components

| | +---form

| | +---table

| | \---users

| +---features

| | +---attendances

| | +---auth

| | +---classes

| | +---courses

| | +---email

| | +---lessons

| | +---permissions

| | +---providers

| | +---roles

| | \---users

| +---pages

| | +---auth

| | +---classes

| | +---courses

| | +---lessons

| | \---users

| \---stylesheets

### Backend

Ve složce backend nalezneme ihned několik složek. Ve složce Config nalezneme nastavení připojení k MongoDB databázi. Jedná se o jednoduchý try catch kód, kde se pomocí knihovny mongoose snažíme připojit k databázi. Dále lze vidět složku middleware, ve které se nachází dva soubory, a to jsou ErrorMiddleware a AuthMiddleware.

ErrorMiddleware se stará o správné zpracování a zobrazení chybových hlášek, kdekoli na backendu. AuthMiddleware má na starosti ověření uživatele a určení jeho pravomocí. AuthMiddleware obsahuje dvě klíčové funkce, a to authenticate a authorize. Authenticate slouží pro ověření totožnosti uživatele, používá u všech rout, kromě routy login, která slouží pro přihlášení uživatele. Uživatel se ověřuje pomocí JWT (JSON Web Tokenu), který se generuje při přihlášení uživatele po zadání správné kombinace emailu a hesla. Druhá klíčová funkce je authorize, která slouží pro ověření pravomocí uživatele. Používá se u rout, které potřebují důležitější pravomoci. Authorize obsahuje parametr pravomocí a může jednat o jednu pravomoci jako text anebo pole pravomocí.

Dále lze vidět složku EmailConfig, která slouží pro nastavení posílání emailů. Emaily se posílají pomocí knihovny Nodemailer přes emailového klienta společnosti Sendinblue, která umožňuje posílat emaily pomocí jejich personifikovaného API klíče. Více o emailu v následujících sekcích. Ve složce s názvem backend se nachází další tři složky a to models, routes a controllers, o nich také více v následujících sekcích.

#### Modely

Ve složce Modely nalezneme modely neboli schémata jednotlivých souborů v kolekci. Schémata určují, jak budou vypadat ukládaná data a jednotlivé položky v souboru a jejich typ. Může se jednat například o položku title s typem string, neboli text. Jednotlivé části aplikace mají svůj vlastní model. Každý model je pojmenovaný dle dané části aplikace. Jednotlivé schémata se dělí na položky trv. fieldy, které představují jednotlivé hodnoty dat. Takže kolekce se dělí na soubory a soubory se dělí na fieldy. Pomocí modelů se vytváří, získávají, upravují a mažou dané soubory v kolekcích. Lze pracovat i s několika soubory najednou.

#### Konfigurace serveru

Složka backend také obsahuje soubor s konfigurací express serveru, který obsahuje nastavení express serveru a klíčové routy. Nastavení serveru obsahuje číslo portu, na kterém server běží, vyhodnocení prostředí, ve kterém se server nachází (produkce nebo vývoj) a také nastavení použití JSON komunikace a errorMiddlewaru. Klíčové routy mají prefix api a poté název dané části aplikace a následně se na každou cestu váže soubor s detailními routy a controllery pro zajištění přehlednosti kódu.

Výpis - Konfigurace express serveru (zdroj: autor)

1. //...
2. connectDB()
3. const app = express()
4. app.use(express.json())
5. app.use(express.urlencoded({extended: false}))
6. …
7. app.use("/api/courses", require("./routes/courseRoutes"))
8. app.use("/api/users", require("./routes/userRoutes"))
9. app.use("/api/classes", require("./routes/classRoutes"))
10. app.use("/api/lessons", require("./routes/lessonRoutes"))
11. app.use("/api/roles", require("./routes/roleRoutes"))
12. app.use("/api/attendances", require("./routes/attendanceRoutes"))
13. app.use("/api/permissions", require("./routes/permissionRoutes"))
14. app.use("/api/sendemail", require("./routes/emailRoutes"))
15. if(process.env.NODE\_ENV == "production") {
16. app.use(express.static(path.join(\_\_dirname, "..", "frontend", "build")))
17. app.get("\*", (req, res) => res.sendFile(path.resolve(\_\_dirname, "..", "frontend", "build", "index.html")))
18. } else {
19. app.get("/", (req, res) => res.send("Please set to production"))
20. }
21. app.use(errorHandler)
22. app.listen(port, () => console.log(`Server started on port ${port}`))

První krok, než se server spustí, je připojení k databázi pro ověření, že je možné s databází komunikovat a předejít tím nechtěných chyb. Server běží na výchozím portu 5000 ale lze ho přizpůsobit dle potřeb dané aplikace.

#### Routes

Nedílnou součástí backendu aplikace je složka Routes, která obsahuje jednotlivé cesty pro každou část aplikace. Cesty pro části aplikace jsou si podobné, skládají se ze čtyř základních požadavků a to post, get, put a delete, kde požadavky get a post jsou na kořenové cestě a požadavky put a delete potřebují id souboru, který se bude upravovat nebo mazat. Každý požadavek obsahuje kontrolu pravomocí uživatele, zda může uživatel na danou routu přistoupit. Příklad cest lze vidět v podkapitole 5.2 u výpisu kódu číslo 3 Ukázka rout lekcí. Všechny routy dohromady poté vytvářejí API pro frontend aplikace.

#### Controllers

Controllery se dělí stejné jako routy dle jednotlivých částí aplikace. Slouží k vyřízení komunikace mezi databází a frontendem. Každý controller obsahuje čtyři základní funkce, kde každá funkce odpovídá danému požadavku z rout. Následovně v každé funkci se díky modelu získávají data z databáze pomocí metod na každém modelu z mongoose. Níže je možné vidět, jak takový controller vypadá, v tomto případě se jedná o controller lekcí.

Výpis - Controller lekcí (zdroj: autor)

1. const asyncHandler = require('express-async-handler')
2. const Lesson = require('../models/lessonModel');
3. const getLessons = asyncHandler(async (req, res) => {
4. const lessons = await Lesson.find({ user: req.user.id })
5. res.status(200).json(lessons)
6. })
7. const setLesson = asyncHandler(async (req, res) => {
8. if(!req.body.title){
9. res.status(400)
10. throw new Error("Please add lesson title")
11. }
12. const lesson = await Lesson.create(req.body)
13. res.status(200).json(lesson)
14. })
15. const updateLesson = asyncHandler(async (req, res) => {
16. const lesson = await Lesson.findById(req.params.id)
17. if(!lesson) {
18. res.status(400)
19. throw new Error("Lesson not find")
20. }
21. const updatedLesson = await Lesson.findByIdAndUpdate(req.params.id, req.body, {
22. new: true,
23. })
24. res.status(200).json(updatedLesson)
25. })
26. const deleteLesson = asyncHandler(async (req, res) => {
27. const lesson = await Lesson.findById(req.params.id)
28. if(!lesson) {
29. res.status(400)
30. throw new Error("Lesson not find")
31. }
32. await lesson.remove()
33. res.status(200).json({id: req.params.id})
34. })
35. /...

Nový soubor se vytvoří pomocí metody create, která obsahuje parametr nových dat. Pro získání dat se používá metoda find, s parametry na získaní jen potřebných dat. Úprava dat probíhá pomocí metody update, kde jsou zapotřebí vyplnit dva parametry a to, určit, který soubor chci aktualizovat a druhý parametr obsahuje aktualizovaná data. K mazání dat se používá metoda remove nebo deleteOne, ale než se soubor smaže je potřeba najít správný soubor ke smazání pomocí id.

#### Konfigurace emailu

Konfigurace a připojení do emailového klienta probíhá pomocí API klíče, získaného po přihlášení do Brevo (Sendinblue) služby. Při každém odeslání emailu se vytvoří transportní objekt z Nodemaileru a email se odešle na určenou emailovou adresu s daným obsahem. Lze posílat email i s přílohou ať už textový dokument nebo obrázek. Brevo (Sendinblue) nabízí posílat celou řadu emailu, od osobních přes marketingové a tak dále. Pro úspěšné odeslání emailu je tedy zapotřebí ověřit svou totožnost u poskytovatele emailového serveru, vyplnit obsah, odesílatele a příjemce emailu. Lze vyplnit i další pole jako například: kopie, skryté kopie a všechny ostatní. Nodemailer nabízí několik způsobu ověření totožnosti, avšak ověření pomocí API klíče v základu nenabízí, proto používám rozšíření vytvořené na míru Brevo (Sendinblue), které se nazývá nodemailer-sendinblue-transport. Toto rozšíření automaticky ověří totožnost pomocí API klíče a pak stačí vytvořit transport z Nodemaileru a odeslat email pomocí metody sendMail.

### Frontend

Složka frontend se dělí na tři podsložky, a to jsou build, public a src. Ve složce build se nachází zkompilovaný kód ze složky src. Složka public obsahuje výchozí html soubor a všechny potřebné soubory pro správné fungování html souboru, jak jsou ikonky, obrázky a další soubory pro meta tagy. Nejdůležitější složka je src, která obsahuje všechen kód, pro správnou funkci aplikace.

Složka src se dále dělí na: pages, components, features, app a stylesheets. Ve složce frontend lze také najít soubor app.js a index.js. Soubor app.js se skládá ze tří částí, které jsou importy – slouží pro použití kódu, který byl definován jinde, app komponenta – je to funkce, která vrací JSX výraz, který nakonec prohlížeč vykreslí, a na konci souboru se nachází příkaz export – zajišťuje, aby daná komponent mohla být použita i v jiných souborech. Takto vypadají všechny komponenty a stránky typu JSX. (Mozilla Corporation, 2023)

App komponent slouží k vytvoření cest k jednotlivým stránkám, obsahuje navigační panel v záhlaví a další komponenty, které jsou potřeba na všech stránkách. V souboru index.js lze nalézt vstupní bod webové aplikace, nachází se zde App.js komponenta. (Mozilla Corporation, 2023) Ve složce stylesheets se nachází všechny sass stylesheety, potřebné pro kompilaci do výchozího stylesheetu index.css.

Složka app obsahuje dva soubory, a to jsou auth-verify a store. Auth-verify je komponenta, sloužící k ověření tokenu uživatele a jeho platnosti. Pokud tokenu vypršela platnost, komponenta uživatele odhlásí. Soubor store vytváří tzv. store, který obsahuje všechny stavy a propojuje je s reducerema. Reducery jsou v podstatě obyčejné JS funkce, které očekávají dva parametry, předchozí stav a akci a poté vrací nový aktualizovaný stav. Jak napovídá název Reducer, česky „Reduktor“, spojí předchozí stav a akci do jedné entity, nové aktualizované instance stavu. (Abramov and Redux documentation authors, 2023)

Výpis - Ukázka části lessonSlice s Reducerem lekcí (zdroj: autor)

1. import {createSlice, createAsyncThunk} from "@reduxjs/toolkit"
2. import lessonService from "./lessonService"
3. const initialState = {
4. lessons: [],
5. isError: false,
6. isSuccess: false,
7. isLoading: false,
8. message: "",
9. }
10. export const createLesson = createAsyncThunk("lessons/create", async (lessonData, thunkAPI) => {
11. try {
12. const token = thunkAPI.getState().auth.user.token
13. return await lessonService.createLesson(lessonData, token)
14. } catch (error) {
15. const message = (error.response && error.response.data &&
16. error.response.data.message) || error.message || error.toString()
17. return thunkAPI.rejectWithValue(message)
18. }
19. })
20. /...
21. export const lessonSlice = createSlice({
22. name: "lesson",
23. initialState,
24. reducers: {
25. reset: (state) => initialState
26. },
27. extraReducers: (builder) => {
28. builder
29. .addCase(createLesson.pending, (state) => {
30. state.isLoading = true;
31. })
32. .addCase(createLesson.fulfilled, (state, action) => {
33. state.isLoading = false
34. state.isSuccess = true
35. state.lessons.push(action.payload)
36. })
37. .addCase(createLesson.rejected, (state, action) => {
38. state.isLoading = false
39. state.isError = true
40. state.message = action.payload
41. })
42. /...
43. }
44. })
45. export const {reset} = lessonSlice.actions
46. export default lessonSlice.reducer

Jednotlivé reducery se skládají z case, které představují požadavky a jejich stav. Daný reducer se poté vyexportuje jako lessonSlice.reducer. Je důležité podoktnout, že se jedná o nový moderní způsob tvorby reducerů. Tento způsob zajišťuje přehlednost a snadnou čitelnost kódu oproti starému způsobu, kde reducery rychle nabíraly na velikosti kódu, a tak se stávaly rychle složitými a velmi obsáhlými. (Abramov and Redux documentation authors, 2023)

#### Features

Tato část obsahuje podsložky dle všech částí aplikace a složku auth. Ve všech složkách se nachází dva soubory se sufixem v názvu Service a Slice.V service souboru se nachází funkce, které díky axiosu posílají požadavky s daty na backend a přijímají odpovědi z backendu. Dále jednotlivé funkce v Service předají data souboru Slice. Axios je knihovna určená pro komunikaci s API, jedná se o http klienta jak pro prohlížeč, tak i pro node.js, který je založený na Promise. (Zabriskie and Sarjeant, 2023)

Výpis - Ukázka části lessonService (zdroj: autor)

1. import axios from "axios"
2. const API\_URL = "/api/lessons/"
3. const createLesson = async (lessonData, token) => {
4. const config = {
5. headers: {
6. Authorization: `Bearer ${token}`
7. }
8. }
9. const response = await axios.post(API\_URL, lessonData, config)
10. return response.data
11. }
12. /...
13. const lessonService = {
14. createLesson,
15. updateLesson,
16. getLessons,
17. deleteLesson,
18. }
19. export default lessonService

Z kódu výše vyplívá, že axios používá klasické http metody, v ukázce výše metoda POST. Lze si povšimnout z čeho se axios požadavek skládá, a to z hlavičky, data a samozřejmě z webové adresy. Druhá část features jsou slices, ukázku. jak takový slice vypadá lze vidět ve výpisu kódu 7. Slices reagují na dané funkce ze services a podle stavu požadavku aktualizuje stav reduxu na úspěchu daného požadavku. Slice je tedy kolekce reducer logiky a akcí pro danou vlastnost aplikace. Typicky jsou definované v jednom souboru. (Abramov and Redux documentation authors, 2023)

#### Pages a Components

V poslední části se zaměřím na jednotlivé stránky a komponenty. Každá část aplikace má část stránek, jsou rozděleny do tří částí a to seznam, detail a akce. Stránka se seznamem vypisuje základní data z určité části aplikace, jedná se o název a popis a jednotlivá data zobrazená pomocí komponenty Card. Detail stránka obsahuje detailní data z jednoho souboru, jako např.: detail lekce obsahuje název lekce, popis, osnovy, videohovor, datum atd. kdežto v seznamu lekcí je vidět jen název a krátký popis. Poslední část je tvorba a editace daných dat, která je zobrazená pomocí formuláře na stránce akce.

Komponenty představují různé součásti stránek, které se dají znovu použít na několika místech. Jedná se o tabulky, inputy, karty, seznamy karet atd. Výhoda použití komponent je velká přehlednost v kódu, velmi snadné úpravy potřebných komponent a snadná údržba.

Výpis - Ukázka komponenty Input (zdroj: autor)

1. const Input = ({ id, value, label, type = "text", placeholder = "", onChange, required, min }) => {
2. return (
3. <div className="form-group ">
4. <label htmlFor={id}>{label}</label>
5. <input type={type} className="form-control" id={id} name={id} placeholder={placeholder}
6. defaultValue={value} onChange={onChange} required={required} min={min}/>
7. </div>
8. )
9. }
10. export default Input

Jak již bylo řečeno komponenty a stránky jsou v podstatě JavaScriptové funkce, které vracejí JSX výraz. JSX výraz je velmi podobný HTML kódu a lze s ním i tak pracovat. Opět je důležité danou komponentu vyexportovat, aby mohla být použita všude, kde je potřeba.

Celkově systém pokrývá všechny požadavky klienta a případy použití. Systém je tedy prozatím kompletní, avšak není dokonalý a musí se neustále zlepšovat. Práce na vývoji systému neustává a bude trvat ještě dlouho. Více v další kapitole.

# Plán testování a nasazení aplikace

Poslední fáze životního cyklu se zaměřuje na plán nasazení a testování aplikace. Jedná se o poslední dva kroky k úspěšnému spuštění systému ve společnosti.

## Uživatelské testování

Testování aplikace probíhá dvěma způsoby. Vývojář systému testuje aplikaci způsobem black-box, tedy zaměřuje se na testování kódu jako takového a probíhá neustále. Druhý způsob je prováděn klientem a lektory ze společnosti. Klient testuje všechny funkcionality administrátorské role a lektoři zkoušejí systém pomocí uživatelů s rolemi lektor a student. Výsledky testování poté jsou používány pro zdokonalení systému. Testování probíhá na každé schůzce s jednatelkou a nápady, návrhy a nedostatky jsou poté vyvíjeny a přidávány během vývoje systému. Testování bude vždy probíhat s klienty a následně bude snaha o co nejrychlejší implementaci užitečných funkcí pro daného klienta a nabídnutí dané funkce i dalším klientům.

Testování probíhalo pomocí testovacích scénářů přiložených v příloze Testovací scénáře. Testovat aplikaci vždy bude klient a několik lektorů, tak aby odpovídali popisu koncového uživatele popsaného v kapitole 2.6. Testovací scénáře se zaměřují na následující funkcionality:

* Zápis docházek
* Průběh online lekcí
* Přihlášení a odhlášení uživatele
* Zobrazení tříd a jejich detailu
* Zobrazení rozvrhu uživatele

Každý testovací scénář v příloze Testovací scénáře obsahuje základní scénář, a navíc téměř všechny testovací scénáře obsahují alternativní scénáře. Každý scénář je nutno otestovat všemi testery a po splnění scénáře je nutno vyhodnotit průběh a prodiskutovat zpětnou vazbu. Následně objevené nápady, návrhy a nedostatky budou začleněny do vývoje systému. Testování probíhalo a bude probíhat opakovaně. Vždy nějaké scénáře projdou a některé naopak selžou.

Popsat testování

## Plán nasazení

Plán nasazení popisuje kroku, které bude nutné provést před nasazením do produkce. Zbývá přejít na produkční databázi, vybrat potřebný plán hostování, zjistit, zda bude potřeba upgrade plán emailového klienta a zakoupit doménu.

MongoDB nabízí dvě možnosti pro produkční aplikace. První je plán „Serverless“, který je určený pro serverless aplikace, které se vyznačují malým nebo proměnlivým provozem. Výhoda je vyžadování minimální konfigurace, placení za spotřebované zdroje a flexibility zdrojů. Tento plán není vhodný pro tuto aplikaci, jelikož data se budou neustále upravovat a přidávat, což by vyžadovalo velký provoz. Druhý plán je „Dedicated“, který je sofistikovaným aplikacím a systémům v produkčním prostředí. Výhody tohoto plánu jsou dostupnost v potřebných regionech a oblastech, vlastní nastavení záloh, až 4 TB úložiště. Výběr plánu záleží na spotřebovaných zdrojích. (Mongo, 2023)

Hostování backend i frontend aplikace neboli Platform as a Service, což znamená, že se jedná o kompletní cloudové prostředí, které zahrnuje vše, co vývojáři potřebují k vytváření, spouštění, hostování a správě aplikací (Google, 2023). Hostování probíhá u společnosti[[2]](#footnote-2), která nabízí několik možností, jak si plán poskládat. První možnost je plán zdarma, který je omezen 100 GB přenosem dat, 500 build minutami a 750 instance-hodinami, kdy je systém dostupný, ať už je používán či nikoliv. Plán zdarma by mohla být vhodná varianta, pokud se nepřekročí limit 750 instance-hodinami, poté se musí vybrat placený plán. Placené plány jsou dva, a to „team“ a organizace. Plán organizace je pro velké organizace a rozdílů mezi team plánem není mnoho. Výhody obou plánu jsou neomezení instance-hodiny, mnohem větší limity pro build minuty a pro přenos dat, podpora pomocí chatu a další. Dále je možné u každého plánu připlatit za jednotlivé služby a rozšíření limitu služeb, dle potřeb dané aplikace. Výběr a skladba plánu se bude odvíjet od spotřebovaných zdrojů. (Render, 2023)

Dále bude nutné zjistit, zda bude potřeba vyšší plán u emailového klienta. Základní plán, který je zdarma, nabízí posílání až 300 emailů denně. Tento plán je rozhodně postačující do začátků, avšak časem a s šířením systému bude rozhodně potřeba plán navýšit, ať už z důvodů velkého počtu studentů anebo posílání marketingových emailů všem studentům, ať současným, nebo minulým. Poté by mohl stačit plán „Starter“, který nabízí až 20 000 emailů měsíčně. Výběr plánu je prozatím jednoznačný, ale do budoucna bude potřeba vyšší. (Sendinblue, 2023)

Před nasazením aplikace do produkce bude vše s klientem prodiskutováno, hlavně se schůzka zaměří na očekávanou návštěvnost a rozpočet. Podle získaných informacích od klienta se poté uváží správná volba plánů pro nasazení aplikace. Poté zbývá výběr domény a ten je vždy jen na daném klientovi.

## Možnosti pokračování

Tato část je zaměřena na možnosti rozšíření a pokračování aplikace. Je spoustu způsobů, jak aplikaci zdokonalit a rozšířit. První nápad a poměrně důležitý je vytvoření aplikace pro mobilní zařízení, a to hlavně pro Android a iOS. Na tento nápad bude zaměřeno ihned po nasazení a otestování aplikace. Jelikož server Express může zůstat téměř stejný bude se měnit hlavně část frontend. Díky ReactJS a jeho sesterské knihovny React Native, který se specializuje na mobilní zařízení, není přechod příliš obtížný, ale ani snadný.

Další možnost rozšíření aplikace je možnost kupování kurzů přímo v aplikaci. Bylo by nutné naimplementovat platební bránu a systém by se musel rozdělit na veřejný sektor a interní sektor, aby nedocházelo k bezpečnostním incidentům. Také by byla možnost generovat faktury ze systému pro výplaty lektorům, avšak zde se musí dávat pozor na lokální zákony, pokud by se aplikace začala používat v zahraničí.

Jedna z mnoha možností jak aplikaci obohatit, je implementace vlastních videohovorů a nevyužívat API meet.jitsi. Ovšem v současném nasazení aplikace přes render.com, kde aplikace používá tarif zdarma a je omezený množství posílání dat mezi klientem a serverem, musel by se zaplatit vyšší tarif, anebo naimplementovat řešení na vlastní server, pokud by byl k dispozici.

Lze nalézt spousty dalších řešení, způsobů a nápadů, jak aplikaci obohatit, vylepšit a zdokonalit. Odevzdáním této práce vývoj aplikace nekončí a bude vyvíjena a testována ještě spoustu měsíců poté.

Závěr

Cílem práce bylo navrhnout a implementovat systém pro správu kurzů. V úvodu byla popsána problematika systémové podpory ve vzdělávací oblasti. Problematika byla rozdělena do tří částí, kde bylo jasně ukázáno proč se systémy implementují. Problematika také sloužila k zařazení systému pro správu kurzů do kategorie edukačních systémů. Dále byl popsán samotný klient, jeho společnost a jeho požadavky na aplikaci.

Následně byla provedena analýza podnikových procesů, požadavků klienta, koncového klienta a rešerše již dostupných řešení na trhu. Analýza podnikových procesů se zaměřila na klíčové procesy v klientovi společnosti a možnost jejich optimalizace a automatizace. Poté byly požadavky rozděleny na funkcionální a nefunkcionální. Poté proběhla krátká analýza koncového uživatele. Následně byla provedena podrobná analýza již dostupných řešeních na trhu a jejich výhod a nevýhod. Z této části vzešlo několik poznatků. První jsou zajímavé funkcionality, které jsou vhodné naimplementovat i do nového systému. Druhý poznatek je takový, že již dostupná řešení jsou drahá a nejsou tak uživatelsky přívětivá. Proto se společnost rozhodla implementovat vlastní řešení, které bude za přijatelnou cenu a bude uživatelsky velmi přívětivé a systém bude vytvořen na míru.

Následně byl proveden návrh systému a určeny technologie, které bude systém používat. Byl vybrán MERN stack, z důvodů jednoduchosti a rychlosti použití, snadné implementace, známých technologií, programovaných v jedno jazyku a použití známých technologií. Technologie byly popsány a bylo určeno k čemu jednotlivé technologie slouží.

Návrh systému je rozdělen na několik částí. Základní částí systému je správa uživatelů. Tuto celou část má na starosti administrátor, jako vytváření, mazání a úprava uživatelů. Uživatelé si mohou zobrazit a upravit svůj profil. Administrátor má také na starost správu rolí, budou potřeba tři role a to student, lektor a administrátor. Správa rolí umožňuje vytvářet a upravovat role s danými vlastnostmi. Část lekcí a kurzů se zabývá všeobecně kurzy, které společnost poskytuje a lekcemi, které dávají dohromady celý kurz. Velmi důležitou součástí systému jsou třídy a jejich správa. Třída představuje daný kurz se studenty a lektory, který se odehrává a opakuje po určitou dobu. Lze si tyto části představit jako ve škole, kdy kurz představuje jednotlivé předměty s osnovami a třída poté představuje žáky, který se učí daný předmět od učitele. Po vytvoření třídy se automaticky generují docházky a rozvrhy daných tříd, studentů a lektorů. Docházky a rozvrhy se opakují v daný čas, který je určen třídou, ale vše lze pozměnit.

Systém nyní běží v testovacím režimu a je neustále zdokonalována, optimalizována a testována. Společnost aplikaci zatím plně nevyužívá v ostrém provozu. Nejdříve je potřeba opakovaně prodiskutovat plán nasazení, kde je několik možností. Možností a kombinací je spoustu, od nejlevnější se základními zdroj po dražší s více než dostačujícími zdroji. Vše záleží na klientovi a jeho možnostech.

V poslední částí je diskuse několika možností rozšíření daného systému. Ukázané možnost nejsou samozřejmě všechny. Způsobů, jak systém obohatit a zdokonalit je mnoho, vždy bude záležet na nápadů a návrhů od klientů, testerů a vývojářů. První rozšíření systému o aplikaci pro mobilní zařízení se systémem Android a iOS je poměrně důležité a bude na něj zaměřeno ihned po zavedení systému.

Primárním cílem této práce bylo navrhnout a implementovat systém, který je určen pro správu výukových kurzů programování. Cíle práce byly naplněny a všechny požadavky a případy užití byly pokryty. Jedním z hlavních přínosů této práce je systém pro správu kurzů s ohledem na možnost implementace systému i v jiných společností. Aplikace využívá MERN stacku a je napsaná v jazyce Javascript, backend systému je tvořen databází MongoDB a Express serverem s pomocí Node.js. Frontend část systému je vytvořena pomocí knihovny ReactJS.

Použitá literatura

Abramov, D. and Redux documentation authors (2023) *Getting Started with Redux | Redux*. Available at: https://redux.js.org/introduction/getting-started (Accessed: 28 April 2023).

Arlow, J. and Neustadt, I. (2007) *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* Brno: Computer Press.

*Best Kids Activity Registration & Management Software | Sawyer Tools* (2023). Available at: https://www.hisawyer.com/for-business/ (Accessed: 28 February 2023).

CODE Rebels (2023) ‘CODE Rebels interní materiály’.

Cohn, M. (2006) ‘Agile Estimating and Planning by Mike Cohn’, *Journal of Product Innovation Management*, 23(6), pp. 588–589.

Daradoumis, T. *et al.* (2010) ‘CRM and higher education: developing a monitoring system to improve relationships in e-learning environments’, *International Journal of Services Technology and Management*, 14, pp. 103–125. Available at: https://doi.org/10.1504/IJSTM.2010.032887.

Express (2023) *Express - Node.js web application framework*. Available at: https://expressjs.com/ (Accessed: 28 February 2023).

Google (2023) *What Is PaaS?*, *Google Cloud*. Available at: https://cloud.google.com/learn/what-is-paas (Accessed: 4 May 2023).

Hrnjic, A. (2016) ‘The transformation of higher education: evaluation of CRM concept application and its impact on student satisfaction’, *Eurasian Business Review*, 6(1), pp. 53–77. Available at: https://doi.org/10.1007/s40821-015-0037-x.

IBM Cloud Education (2021) *What is Business Process Analysis?* Available at: https://www.ibm.com/cloud/blog/business-process-analysis (Accessed: 18 March 2023).

Jha, S. (2021) ‘Understanding the MVC Architecture in the MERN Stack’, *Medium*, 13 March. Available at: https://shubhamjha25.medium.com/understanding-the-mvc-architecture-in-the-mern-stack-aff893abce50 (Accessed: 27 April 2023).

Meta (2023) *React – A JavaScript library for building user interfaces*. Available at: https://reactjs.org/ (Accessed: 27 February 2023).

*Metodika MMSP-AV* (2023). Available at: http://mmsp-av.wz.cz/ (Accessed: 21 September 2023).

Microsoft (2023) *Create an Event-driven Process Chain (EPC) diagram - Microsoft Support*. Available at: https://support.microsoft.com/en-us/office/create-an-event-driven-process-chain-epc-diagram-c821fae3-8621-46dd-b885-2239405554be (Accessed: 3 May 2023).

Mongo (2023) *Pricing | MongoDB*. Available at: https://www.mongodb.com/pricing (Accessed: 29 April 2023).

MongoDB (2023a) *Schemas — Atlas App Services*. Available at: https://www.mongodb.com/docs/atlas/app-services/schemas/ (Accessed: 27 February 2023).

MongoDB (2023b) *What Is NoSQL? NoSQL Databases Explained*, *MongoDB*. Available at: https://www.mongodb.com/nosql-explained (Accessed: 27 April 2023).

MongoDB (2023c) *What Is The MERN Stack? Introduction & Examples*, *MongoDB*. Available at: https://www.mongodb.com/mern-stack (Accessed: 27 February 2023).

mongoose (2023) *Mongoose v7.1.0: Schemas*. Available at: https://mongoosejs.com/docs/guide.html (Accessed: 27 April 2023).

Morin, A. (2019) *What Is Personalized Learning*, *Understood*. Available at: https://www.understood.org/en/articles/personalized-learning-what-you-need-to-know (Accessed: 3 May 2023).

Mozilla Corporation (2023) *Getting started with React - Learn web development | MDN*. Available at: https://developer.mozilla.org/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks/React\_getting\_started (Accessed: 28 April 2023).

Reinman, A. (2023) *Nodemailer :: Nodemailer*. Available at: https://nodemailer.com/about/ (Accessed: 4 May 2023).

Render (2023) *Pricing | Render · Cloud Hosting for Developers*, *Pricing | Render · Cloud Hosting for Developers*. Available at: https://render.com/pricing (Accessed: 29 April 2023).

Sanie, M. (2022) *5 Phases of Software Development Life Cycle and Risk Assessment*, *Cprime*. Available at: https://www.cprime.com/resources/blog/5-phases-of-software-development-life-cycle-and-risk-assessment/ (Accessed: 24 January 2023).

Sass team (2023) *Sass: Syntactically Awesome Style Sheets*. Available at: https://sass-lang.com/ (Accessed: 21 March 2023).

Sendinblue (2023) *Sendinblue*. Available at: https://app.sendinblue.com/billing/account/marketing/plans?headerContent=false (Accessed: 29 April 2023).

Singla, C. (2020) ‘Functional vs Non Functional Requirements’, *GeeksforGeeks*, 28 April. Available at: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/ (Accessed: 26 January 2023).

Software Testing Help (2023) *What Is SDLC (Software Development Life Cycle) Phases & Process*, *Software Testing Help*. Available at: https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/ (Accessed: 24 January 2023).

*Teach ’n Go - Modern School Management Software* (2023). Available at: https://www.teachngo.com/ (Accessed: 28 February 2023).

Terra, J. (2023) *What is Client-Server Architecture? Everything You Should Know | Simplilearn*, *Simplilearn.com*. Available at: https://www.simplilearn.com/what-is-client-server-architecture-article (Accessed: 4 May 2023).

Tudor, C. (2022) ‘The Impact of the COVID-19 Pandemic on the Global Web and Video Conferencing SaaS Market’, *Electronics*, 11(16), p. 2633. Available at: https://doi.org/10.3390/electronics11162633.

Zabriskie, M. and Sarjeant, J., Jakob (2023) *Getting Started | Axios Docs*. Available at: https://axios-http.com/docs/intro (Accessed: 28 April 2023).

Přílohy

Přílohy obsahují nedílné součásti této práce. Nejdůležitější přílohou je samotný kód. Další přílohy jsou interní materiály, diagramy use case, diagramy podnikových procesů, model databáze a testovací scénáře.

1. Kód a demo aplikace

Kód implementované aplikace je dostupný pod licencí MIT na adrese <https://github.com/petrpepe/course-management-is>. Demo aplikace je dostupné na adrese [https://sis-cr.onrender.com](https://sis-cr.onrender.com/). Pro přístup do dema aplikace je nutné zadat přihlašovací email a heslo. Existují tři uživatelé a to student, lektor a admin. Student má přihlašovací údaje student@a.a a heslo student, lektor má lektor@a.a a heslo lektor a admin a jeho přihlašovací údaje jsou admin@a.a a heslo admin. Lze si vyzkoušet jakoukoli roli a uživatele.

1. Interní materiály

Potřebné materiály společnosti pro zobrazení vývoje společnosti a informací o dané společnosti. Pomáhají určit požadavky na systém a ukazují základy pro podnikové procesy a jejich následnou analýzu.

1. Testovací scénáře

|  |  |  |  |
| --- | --- | --- | --- |
| **Testovací scénář - 1** | | | |
| **Název scénáře** | Zapsání docházky lektorem | | |
| **Stránky a komponenty** | Stránka konkrétní lekce | | |
| **Cíl scénáře** | Zapsat docházku studentům | | |
| **Aktéři** | Uživatel, informační systém | | |
| **Podmínky** | Uživatel je přihlášen, uživatel má roli lektora a je autorizován | | |
| **Základní scénář** | 1 | Uživatel | Vidí tabulku s docházkami a klikne na daný řádek se studentem a zapíše |
| 2 | IS | Uloží docházku danému studentovi |
| 3 | Uživatel | Vidí zapsanou docházku |
| **Alternativní scénář** | 1 | Uživatel | Vidí tabulku s docházky a klikne na daný řádek se studentem a zapíše |
| 2 | IS | Nastane chyba |
| 3 | Uživatel | Vidí chybovou hlášku |

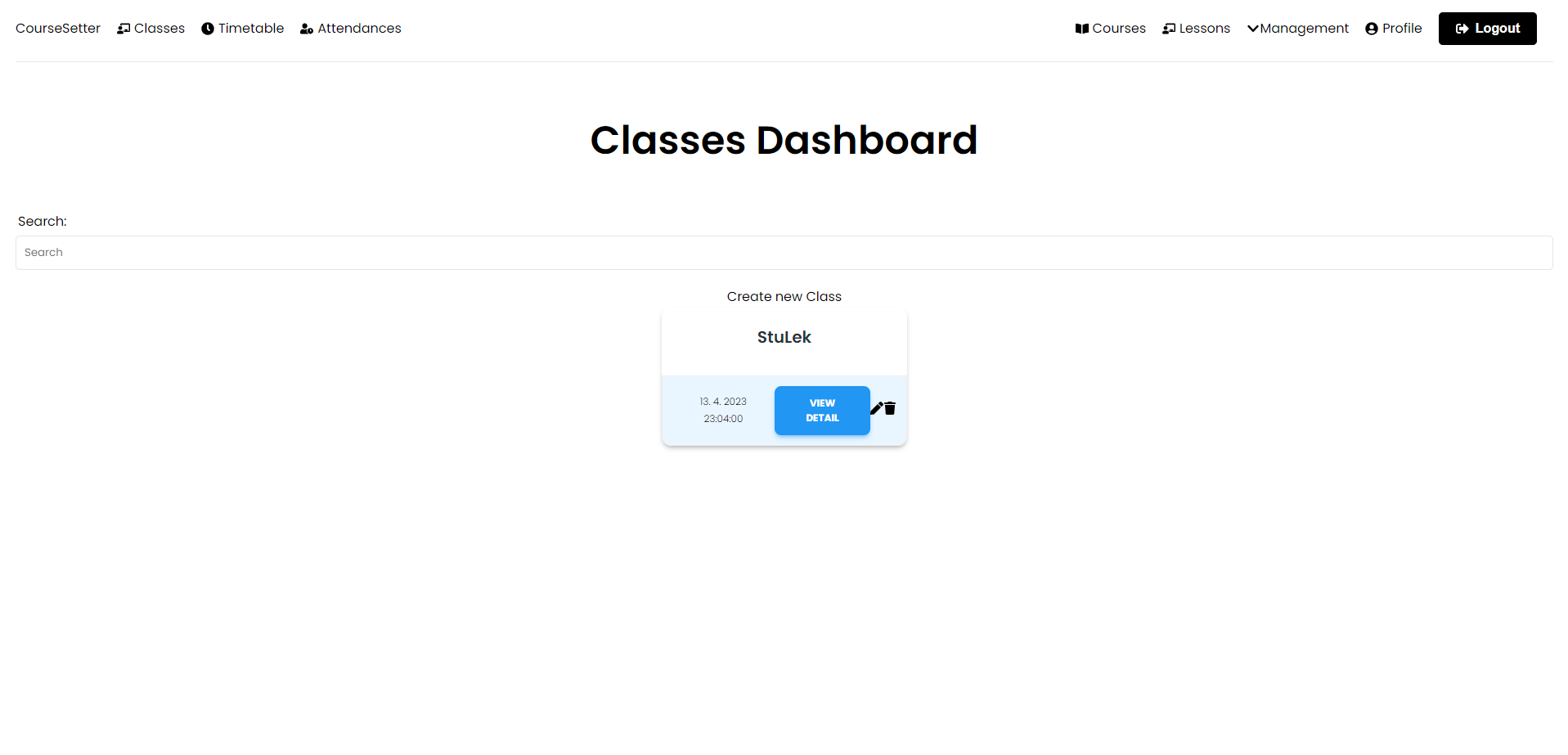
|  |  |  |  |
| --- | --- | --- | --- |
| **Testovací scénář - 2** | | | |
| **Název scénáře** | Průběh online lekcí | | |
| **Stránky a komponenty** | Stránka konkrétní lekce | | |
| **Cíl scénáře** | Ověřit funkčnost videohovoru, zobrazení docházek a osnov | | |
| **Aktéři** | Uživatel, informační systém | | |
| **Podmínky** | Uživatel je přihlášen, uživatel má roli lektora a je autorizován | | |
| **Základní scénář** | 1 | Uživatel | Vidí tabulku s docházkami, vidí osnovy a okno s videohovorem |
| 2 | IS | Propojí uživatele s druhým uživatelem |
| 3 | Uživatel | Slyší a vidí druhého uživatele a jeho obrazovku a může sdílet video a obrazovky |
| **Alternativní scénář 1** | 1 | Uživatel | Vidí tabulku s docházkami, vidí osnovy a okno s videohovorem |
| 2 | IS | Nastane chyba |
| 3 | Uživatel | Nevidí okno s videohovorem a popíše co vidí |
| **Alternativní scénář 2** | 1 | Uživatel | Vidí tabulku s docházkami, vidí osnovy a okno s videohovorem |
| 2 | IS | Nastane chyba |
| 3 | Uživatel | Vidí okno s videohovorem, ale nelze se spojit s dalším uživatel nebo ho neslyší/nevidí |
| 4 | Uživatel | Zkusí se znovu připojit, popíše, co vidí |

|  |  |  |  |
| --- | --- | --- | --- |
| **Testovací scénář - 3** | | | |
| **Název scénáře** | Přihlášení a odhlášení uživatele | | |
| **Stránky a komponenty** | Stránka přihlášení | | |
| **Cíl scénáře** | Uživatel se přihlásil a následně je odhlášen | | |
| **Aktéři** | Uživatel, informační systém | | |
| **Podmínky** | Uživatel existuje v databázi | | |
| **Základní scénář** | 1 | Uživatel | Vidí formulář s přihlášením a vyplní email a heslo a klikne na tlačítko přihlásit se |
| 2 | IS | Zkontroluje kombinaci údajů a vrátí uživatele z DB a jeho token |
| 3 | Uživatel | Je přihlášen a klikne na tlačítko odhlásit se |
| 4 | IS | Odhlásí uživatele a zneplatní token |
| 5 | Uživatel | Je odhlášen |
| **Alternativní scénář** | 1 | Uživatel | Vidí formulář s přihlášením a vyplní email a heslo a klikne na tlačítko přihlásit se |
| 2 | IS | Nastane chyba |
| 3 | Uživatel | Vidí chybovou hlášku |

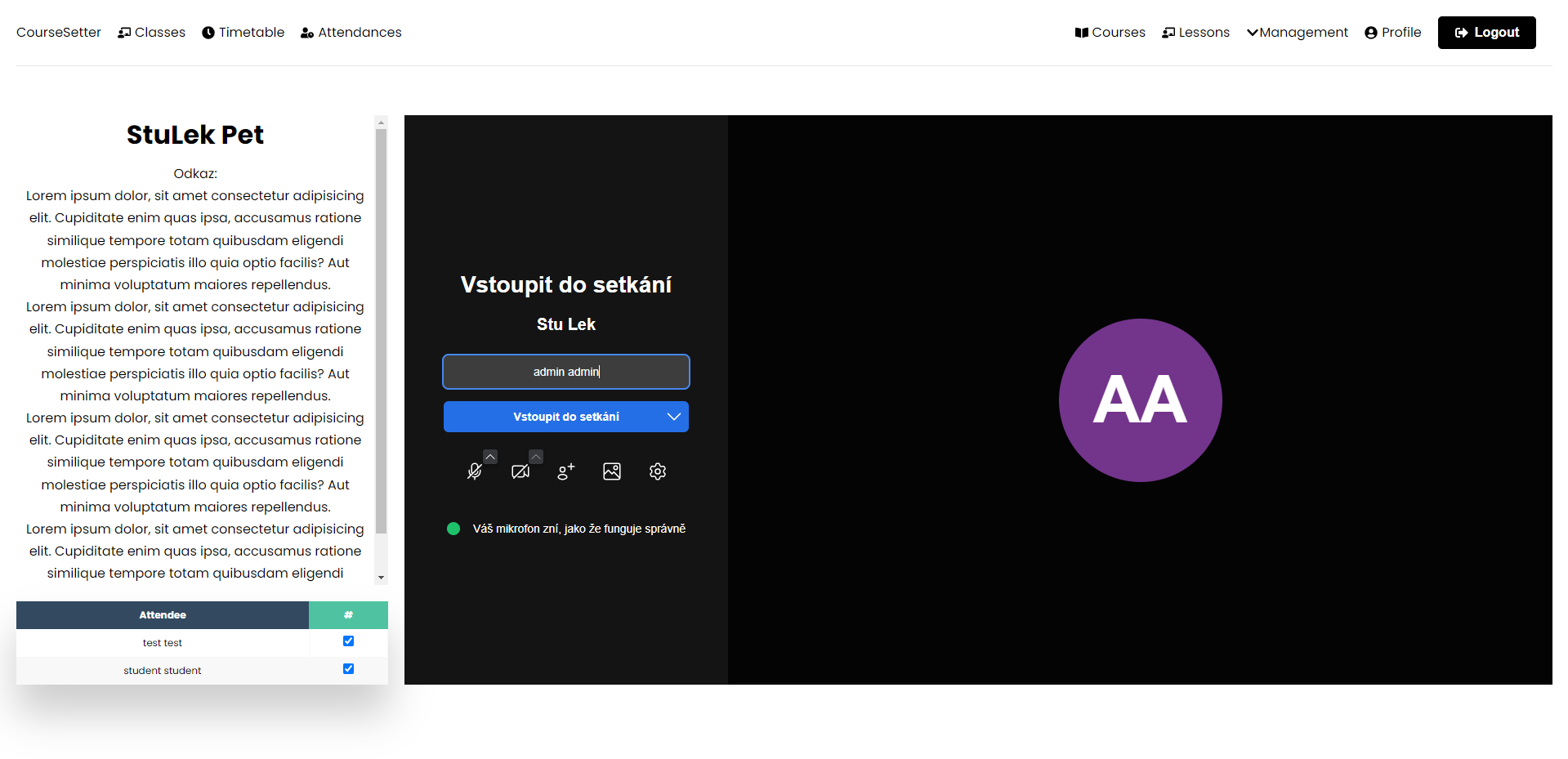
|  |  |  |  |
| --- | --- | --- | --- |
| **Testovací scénář - 4** | | | |
| **Název scénáře** | Zobrazení tříd a jejich detailu | | |
| **Stránky a komponenty** | Stránka seznamu tříd a detailu třídy | | |
| **Cíl scénáře** | Zobrazení seznamu tříd a zobrazení detailu jedné třídy | | |
| **Aktéři** | Uživatel, informační systém | | |
| **Podmínky** | Uživatel je přihlášen, uživatel má roli lektora nebo studenta a je autorizován | | |
| **Základní scénář** | 1 | IS | Zobrazí seznam tříd, ve kterých je daný uživatel zapsaný |
| 2 | Uživatel | Vidí seznam tříd, ve kterých je zapsaný s tlačítkem „Detail“ |
| 3 | Uživatel | Klikne na tlačítko „Detail“ |
| 4 | IS | Zobrazí detail třídy |
| **Alternativní scénář** | 1 | IS | Nezobrazí seznam tříd, uživatel není nikde zapsaný |
| 2 | Uživatel | Vidí prázdný seznam s textem „Nejsi zapsaný v žádné třídě“ |
| **Alternativní scénář** | 1 | IS | Nezobrazí seznam tříd, z důvodu chyby |
| 2 | Uživatel | Vidí chybovou hlášku |

|  |  |  |  |
| --- | --- | --- | --- |
| **Testovací scénář - 5** | | | |
| **Název scénáře** | Zobrazení rozvrhu uživatele | | |
| **Stránky a komponenty** | Stránka rozvrhů | | |
| **Cíl scénáře** | Zobrazení rozvrhu uživatele | | |
| **Aktéři** | Uživatel, informační systém | | |
| **Podmínky** | Uživatel je přihlášen, uživatel má roli lektora nebo studenta a je autorizován | | |
| **Základní scénář** | 1 | IS | Zobrazí domácí obrazovku |
| 2 | Uživatel | Vidí domácí obrazovku“ |
| 3 | Uživatel | Klikne na tlačítko „Rozvrhy“ |
| 4 | IS | Zobrazí týdenní rozvrh s lekcemi daného uživatele |
| **Alternativní scénář** | 1 | IS | Zobrazí prázdný rozvrh |
| 2 | Uživatel | Vidí prázdný seznam s textem „Nemáš žádnou lekci tento týden.“ |
| **Alternativní scénář** | 1 | IS | Nezobrazí rozvrh, z důvodu chyby |
| 2 | Uživatel | Vidí chybovou hlášku |

1. Ukázky výsledné aplikace



Obrázek - Seznam tříd (zdroj: autor)



Obrázek - Detail lekce (zdroj: autor)

Obsah obrázku stůl

Popis byl vytvořen automaticky

Obrázek - Rozvrh třídy (zdroj: autor)

1. Název společnosti CODE Rebels [↑](#footnote-ref-1)
2. Název společnosti Render.com [↑](#footnote-ref-2)