TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

# Webová aplikace pro time-management on-line výuky

## Diplomová práce

| | |
|---|---|
| *Studijní program:* | N6209 Systémové inženýrství a informatika |
| *Studijní obor:* | Manažerská informatika |
| *Autor práce:* | **Bc. Vítězslav Vejnar** |
| *Vedoucí práce:* | Mgr. Tomáš Žižka, Ph.D.<br>Katedra informatiky |

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

**Zadání diplomové práce**

# Webová aplikace pro time-management on-line výuky

*Jméno a příjmení:* **Bc. Vítězslav Vejnar**
*Osobní číslo:* E19000351
*Studijní program:* N6209 Systémové inženýrství a informatika
*Studijní obor:* Manažerská informatika
*Zadávající katedra:* Katedra informatiky
*Akademický rok:* **2020/2021**

**Zásady pro vypracování:**

1. Problematika on-line výuky a aplikace pro její podporu
2. Analýza aplikací pro online výuku
3. Design uživatelského rozhraní
4. Vývoj aplikace a struktura její databáze
5. Uživatelské testování aplikace
6. Zhodnocení navržených řešení

*Rozsah grafických prací:*
*Rozsah pracovní zprávy:* 65 normostran
*Forma zpracování práce:* tištěná/elektronická
*Jazyk práce:* Čeština

**Seznam odborné literatury:**


- FLANAGAN, David, 2020. *JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language.* 7. vyd. Sebastopol: O'reilly media. ISBN 978-1491952023.
- KRUG, Steve, 2015. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability.* 3. vyd. Londýn: Pearson education. ISBN 978-9332542860.
- TATROE, Kevin a Peter MACINTYRE, 2020. *Programming PHP: Creating Dynamic Web Pages.* 4. vyd. Sebastopol: O'reilly media. ISBN 978-1492054139.
- ULLMAN, Larry, 2017. *PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide.* 5. vyd. Berkeley: Peachpit press. ISBN 978-0134301846.
- WOOD, Brian, 2019. *Adobe Illustrator Classroom in a Book (2020 release).* San Francisco: Adobe Press. ISBN 978-0136412670.
- PROQUEST. 2020 *Databáze článků ProQuest [online].* Ann Arbor, MI, USA: ProQuest. [cit. 2020‑10-15]. Dostupné z: http://knihovna.tul.cz


*Konzultant:* Ing. Miroslav Jasso

*Vedoucí práce:* Mgr. Tomáš Žižka, Ph.D.
Katedra informatiky

*Datum zadání práce:* 1. listopadu 2020
*Předpokládaný termín odevzdání:* 31. srpna 2022

L.S.

Ing. Aleš Kocourek, Ph.D.
děkan

doc. Ing. Klára Antlová, Ph.D.
vedoucí katedry

V Liberci dne 1. listopadu 2020

# Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

1. května 2021 Bc. Vítězslav Vejnar

# Anotace

Tato diplomová práce se věnuje tématu vývoje webové aplikace pro efektivní time-management on-line výuky. V teoretické části se nejprve zaměřuje na specifika on-line výuky, její historický vývoj a současný stav.

V následujících kapitolách pak analyzuje vybrané aplikace pro videokonference a zkoumá jednotlivé vlastnosti webových aplikací, jejich komponentů a prostředků užívaných během jejich vývoje.

V rámci praktické části se práce věnuje návrhu aplikace pro zajištění efektivnější komunikace mezi nabídkou on-line kurzů a zájemci. Dále také realizaci vývoje jejího ukázkového modulu pro správu individuálně vyučovaných hodin. Podrobně rozebírá jednotlivé komponenty tohoto modulu.

Hlavním cílem této diplomové práce je navrhnout webovou aplikaci pro time-management on-line výuky se systémem automatizované komunikace mezi lektorem a studentem propojenou s vybraným nástrojem pro videokonferenční hovory.

# Klíčová slova

webová aplikace, on-line výuka, videokonference, time-management, vývoj webových aplikací, PHP, Google API, Gmail API, Google Calendar API, Google Meet

# Abstract

## Web application for online education time-management

This diploma thesis covers the topic of development of a web application focused on effective on-line education time-management. The theoretical part first delves into the specifics of on-line education, it‘s historical development and it‘s current state.

The forthcoming chapters analyse selected videoconference solutions and research the properties of web applications, their components and the means of their development.

The main focus of the empirical part of the thesis is the design of a web application which may provide effective means of communication between the supply and the demand of on-line courses. The design part is followed by the analysis of the process of creation of the selected module from the app, which shall demonstrate the workflow of management of the individually taught courses. The components of said module are thoroughly analysed.

The main goal of this diploma thesis is to design a web application for the purposes of on-line education time-management which utilizes a system of automated communication inbetween the tutor and the student. Said application shall be connected with a selected videocall solution.



# Keywords

web application, on-line education, videoconference, time-management, web application development, PHP, Google API, Gmail API, Google Calendar API, Google Meet

# Poděkování

Děkuji všem, kteří se přímo či nepřímo podíleli na vzniku této práce. Velice děkuji vedoucímu práce Mgr. Tomáši Žižkovi Ph.D. za trpělivé vedení a konstruktivní poznámky, kterými mi velkou měrou pomohl nalézt správný směr při psaní. Mé díky patří také Ing. Miroslavu Jassovi, který mi poskytl důležité rady o fungování back-endu a Googlovských API a Bc. Vojtěchu Láskovi, který mi pomohl vyjasnit výsledný vzhled aplikace. Dále děkuji také všem účastníkům testovacích streamů projektu, kteří si našli čas a poskytli důležité poznámky prostřednictvím dotazníků.

Uznání patří také mým přátelům, kteří se velkou měrou zasloužili o mé zdárné dopsání práce a pomohli mi v kritických chvílích zachovat chladnou hlavu. V době zavřených kaváren také děkuji všem, jejichž byt jsem mohl na nějakou dobu okupovat, protože neumím pracovat u svého vlastního stolu. Svým kolegům ze ŠKODA Academy děkuji za shovívavost s mou osobou a za to, že byli připraveni za mě kdykoli zaskočit, abych mohl zdárně dokončit tuto práci. Mé rodině děkuji za podporu, kterou mi poskytovala po celou dobu studia. Poslední, komu bych zde chtěl poděkovat, je můj táta, díky kterému vím, že když se člověk nevzdá, dokáže toho mnohem víc, než si ostatní možná mysleli a že vzdávat se bez boje opravdu nemá smysl.

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam zdrojových kódů

# Seznam grafů

# Seznam použitých zkratek

| | |
|---|---|
| *API* | Application Programming Interface (sbírka procedur, funcí, či protokolů nějakého programu, které může programátor využívat) |
| *CSS* | Cascading Style Sheets (jazyk používaný pro změnu vzhledu webových stránek) |
| *ERD* | Entity Relationship Diagram (diagram znázorňující vztahy datových entit) |
| *GUI* | Graphic User Interface (grafické uživatelské rozhraní) |
| *HTML* | Hyper-text Mark-up Language (značkovací jazyk používaný pro tvorbu webových stránek) |
| *HTTP* | Hyper-text Transfer Protocol (komunikační protokol určený pro komunikaci mezi klientem a webovým serverem) |
| *ID* | Identifier (identifikátor) |
| *json* | JavaScript Object Notation (formát souboru) |
| *Kč* | Česká koruna |
| *LMS* | Learning Management System (systém pro správu učebních materiálů) |
| *MIT* | Massachusets Institute of Technology (univerzita ve městě Cambridge ve spojených státech amerických) |
| *MOOC* | Massive Open On-line Course (druh on-line kurzu pro široké publikum) |
| *MS* | Microsoft (ve spojeních jako *MS Teams* či *MS Forms*) |
| *MVC* | Model-View-Controller (architektura využívaná při webovém vývoji) |
| *OOP* | Object Oriented Programming (objektově orientované programování) |
| *SQL* | Structured Query Language (jazyk využívaný pro komunikaci s relačními databázemi) |
| *UI* | User Interface (uživatelské rozhraní) |
| *UX* | User Experience (dojem uživatele z užívání aplikace, uživatelská přívětivost) |
| *URI* | Uniform Resource Identifier (jednotný identifikátor zdroje) |
| *USD* | Americký dolar |
| *WBS* | Work Breakdown Structure (metoda používaná v projektovém řízení) |

# Úvod

Současná pandemie koronaviru způsobila zásadní změny ve společnosti. Většina sektoru vzdělávání byla, alespoň dočasně, nucena přistoupit k vyučování v on-line podobě. Společně s tímto nutným opatřením se rozšířilo i velké množství často, pro pracovníky ve vzdělávání, do té doby méně známých nástrojů, které umožňují alespoň částečně suplovat prezenční výuku.

Řešení pro videokonferenční hovory jsou na trhu v současné době zastoupena velkou měrou a jejich možnosti pro podporu on-line výuky se neustále rozšiřují. Jejich používání si nicméně vyžaduje velkou dávku pozornosti a plánování, která zahrnuje i komunikaci mezi studenty a vyučujícími spojenou s domlouváním vhodných termínů. Právě tato komunikace může být jedním z neefektivních bodů on-line setkávání v době, která je stále velice uspěchaná.

Zejména pak soukromí lektoři poskytující individuální vzdělávání za úplatu mohou pociťovat zvýšenou náročnost komunikace v přípravné fázi jednotlivých hodin, mimo jiné při domlouvání termínů lekcí se studenty. Řešením pro ně by mohlo být využití dedikovaného nástroje, který by umožňoval správu kurzů spolu s možností jejich nabídky a smluvení termínů výuky za využití automatizované komunikace se zájemcem o kurz. Zmíněný prostředek by mohl sloužit jako základ platformy pro prodej kurzů, která by po zaplacení poplatku okamžitě generovala potřebné zdroje k uskutečnění on-line hodiny. Například odkaz na videokonferenční hovor by tak mohl být pro všechny zainteresované uživatele generován automaticky po odsouhlasení lekce lektorem. Ideálním typem takového nástroje by tak díky dostupnosti mohla být webová aplikace.

Cílem této diplomové práce je tak prozkoumat využitelné prostředky a provést návrh takového řešení spolu s vyvinutím ukázkové aplikace pro referenci v případě navazujícího vývoje a demonstraci některých z jeho funkcí. Navržený nástroj by měl být propojený s nějakým ze široce využívaných řešení pro videokonference.

# 1 On-line výuka

V souvislosti s pandemií nemoci covid-19 narostlo v letech 2020 a 2021 využívání prostředků on-line výuky, v této kapitole se zaměříme na specifika tohoto druhu vyučování.

## 1.1 Definice a historie pojmu

Pokud si rozebereme pojem on-line výuka, můžeme vycházet z definic jeho částí. Průcha a kol. (2013) definují v Pedagogickém slovníku slovo *výuka*, potažmo *vyučování* jako jistou činnost odehrávající se každodenně ve školních třídách. Tato definice však není pro on-line způsob výuky uplatnitelná, a tak je potřeba čerpat i z druhého nabízeného významu, kterým je jistý druh lidské činnosti, jehož podstatou je interakce mezi učitelem a žákem či žáky. Pro výraz *on-line* existuje mnoho definic, v britské angličtině výraz označuje činnosti či jevy spojené s využíváním či využívající počítač. V češtině pak slovo *on-line* indikuje připojení k internetu – respektive jeho přítomnost, při absenci internetového připojení lze použít výraz *off-line.*

On-line výuka tak může být definována jako vzdělávání, které je uskutečňované prostřednictvím internetu, který je využívaný k zprostředkování interakcí mezi učitelem a žáky a ke sdílení výukových materiálů (What is online education?, 2021).

Jedním z největších benefitů on-line vzdělávání je jeho přístupnost – materiály jsou dostupné studujícím okamžitě prostřednictvím příslušného internetového rozhraní. Tato dostupnost se však stala samozřejmostí až v posledních desetiletích v souvislosti s masovým nástupem internetu. Historie samotné distanční výuky nicméně sahá mnohem dále, prvními příklady tohoto způsobu vyučování byly korespondenční kurzy. První zdokumentovaný příklad takového kurzu se objevil již v roce 1728 v novinách *Boston Gazette*, jedno z jejichž vydání obsahovalo inzerát nabízející takový způsob sebevzdělávání za pomoci lektora. Ve Spojených státech amerických později v průběhu devatenáctého století začaly nabízet korespondenční kurzy i tradiční školské instituce.

Vynálezy a rozšíření jednotlivých druhů médií umožnily další rozvoj distanční výuky, a tak postupně vznikaly kurzy distančního vzdělávání využívající radiového a televizního vysílání. S rozšířením telefonní technologie vznikaly také kurzy založené na komunikaci

učitele a žáka po telefonu. Roku 1976 vznikla v USA první „virtuální vysoká škola“, Coastline Community College fungovala bez jakéhokoli reálného kampusu a nabízela široké portfolio telefonických kurzů.

První vysokoškolské kurzy založené na použití internetu vznikaly ve Spojených státech v průběhu osmdesátých let, velký vývoj pak zaznamenaly v letech devadesátých. Roku 1996 byla plně akreditována první čistě webově založená univerzita v USA. Dále roku 1997 začalo větší množství institucí využívat systém Interactive Learning Network, tento e-learningový systém byl přelomový zejména tím, že využíval relační databázi (fungoval tak podobně jako dnešní LMS). Společnost Blackboard Inc., která byla založena téhož roku, vyvinula standardizovanou platformu pro správu on-line kurzů, která umožnila přechod do on-line prostředí více vzdělávacím institucím.

V prvních dvou desetiletích 21. století došlo k celosvětovému rozmachu užívání internetu, spolu s tímto fenoménem se rozšířily i možnosti on-line výuky. V USA například docházelo k nasazování nových platforem pro správu materiálů pro on-line výuku, content management systém WebCT (Web Course Tools) se stal velice úspěšným již v roce 2003 a to nejen ve Spojených státech, ale i ve světě. Měl více než 6 milionů uživatel z řad studentů v 55 zemích světa.

Nové možnosti pro on-line výuku v podobě možnosti sdílení videí se objevily také se startem platformy YouTube v roce 2005, roku 2009 dokonce sama YouTube spustila speciální odnož YouTube EDU, která nabízela zdarma velké množství on-line přednášek. V roce 2012 společnost Udacity začala nabízet kurzy MOOC (Massive open online courses), které umožňují vzdělávání velkého množství účastníků zároveň. Tento trend následovaly i univerzity jako například MIT a Harvardova univerzita se svojí platformou edX (Miller, 2014).

### 1.1.1 Online vzdělávání v rámci ČR

V Česku se prostředky online vzdělávání začaly využívat přibližně od roku 2000. Navzdory tomu, že v té době ještě ČR nebyla členem Evropské unie, byl jedním z impulsů pro začátek využívání prvků e-learningu evropský projekt eEuropa a elektronizace státní sféry. Počáteční projekty byly realizovány zejména v prostředí univerzit, jedním z takových byl například společný projekt virtuální univerzity, který realizovaly Vysoká škola báňská v Ostravě, Ostravská univerzita a Slezská univerzita v Karvinné. Zmíněný

projekt napomáhal zejména žákům kombinovaného studia, které propojovalo prezenční a distanční formu výuky. ČVUT v Praze se později začalo zabývat metodikou e-learningového systému *WebCT*, ten byl tehdy jedním z nejšíře využívaných systémů tohoto typu.

V současnosti (mimo období nutné on-line výuky) jsou v českém školství e-learning a další prostředky on-line výuky využívány zejména jako podpůrné prvky, často pro distanční vzdělávání žáků mimo denní formu studia. Pro žáky denní formy studia se pak může e-learning stát užitečnou oporou při studiu ať už ve formě úložiště studijních materiálů či cvičení zahrnujících probíranou látku. Různé formy e-learningových rozhraní nebo aplikací pro koordinaci on-line výuky umožňují také jednoduchou koordinaci a správu zadání jednotlivých úkolů společně s prostředky pro jejich vyhodnocení. On-line prvky lze využívat také jako prostředek komunikace mezi učitelem a žákem i během prezenční výuky. V poslední době se tak rozšiřuje využívání příslušných softwarů také pro interaktivní vedení hodiny, při kterém má žák příležitost například odpovídat na otázky, sám otázky pokládat či se přihlásit pomocí příslušného zařízení.

On-line forma výuky má své místo také ve firemním vzdělávání, kde najchází uplatnění zejména při zprostředkování zákonných a profesních školení, odborné certifikace pro specializované činnosti či jiných interních školení. Rozšiřování kvalifikace zaměstnanců je díky tomuto přístupu v mnoha případech méně časově i finančně náročné (Co je to e-learning a jaká je jeho historie, 2016).

## 1.2 Prostředky on-line výuky

Pro on-line výuku existuje velké množství dostupných prostředků, aplikace využívané pro on-line výuku se liší svým účelem a funkcionalitami, jejich nutným specifikem je možnost komunikace (potažmo sdílení dat) přes internet. Komunikaci všeobecně můžeme rozdělit na synchronní a asynchronní, přičemž při on-line výuce můžeme oba tyto typy komunikace kombinovat.

### 1.2.1 Nástroje synchronní komunikace

Prostředky synchronní komunikace dovolují komunikaci v reálném čase, k prostředkům tohoto typu tak můžeme zařadit například chatovací aplikace, které nám umožňují tzv. *instant-messaging* (tento pojem bychom mohli volně přeložit jako okamžité zasílání

zpráv), dalším příkladem pak mohou být i aplikace pro videohovory. Hlavním znakem synchronní komunikace je okamžitá možnost příjemce zprávy reagovat (Specifika online komunikace: Online disinhibition effect a flame wars, 2019). V ČR v současné době mezi nejpopulárnější aplikace pro synchronní komunikaci v rámci výuky patří například aplikace Google Meet či MS Teams, z nichž každá nabízí jiný set nástrojů, jejich specifikům se práce více věnuje v kapitole 2. Každý z prostředků synchronní komunikace je vhodný pro jiné účely, jejich příklady můžeme vidět v tabulce níže.

*Tabulka 1: Příklady nástrojů synchronní komunikace a jejich využití (Synchronous vs. Asynchronous, 2015)*

| Nástroj | Využití |
|---|---|
| Konferenční hovory (audio) | Diskuze a rozhovory |
| Videokonference | Komplexnější diskuze s vyšší úrovní interakce |
| Instant messaging | Rychlá komunikace ad-hoc |
| Sdílení práce pomocí systému interaktivní tabule | Vývojová či designová spolupráce |
| Sdílení dokumentů | Společná práce na dokumentech |

### 1.2.2 Nástroje nesynchronní komunikace

Tyto nástroje komunikace nevyžadují připojení k síti v reálném čase. Příkladem můžou být například e-mail, různá diskuzní fóra či virtuální výuková prostředí. Pro komunikaci skrze tyto prostředky není nutné okamžité zapojení účastníků. Komunikovaný obsah v mnohých případech bývá více komplexní. Pro on-line výuku jsou nejstěžejnějšími nástroje obsažené v LMS (Pilková, 2016). Podobně jako synchronní, i asynchronní nástroje se velmi často liší účelem, příklady některých z nich naleznete v tabulce níže.

*Tabulka 2: Příklady nástrojů asynchronní komunikace a jejich využití (Synchronous vs. Asynchronous, 2015)*

| Nástroj | Využití |
|---|---|
| Fóra | Dlouhodobá diskuze/dialog |
| Zasílání zpráv (e-mail) | Komunikace one-to-one nebo one-to-many |
| Video/Audionahrávky | Komunikace či vyučování |
| Komentované prezentace | Komunikace či vyučování |
| Knihovny dokumentů | Správa zdrojů výuky/práce |

| Nástroj | Využití |
|---|---|
| Databáze | Správa dat |
| Dotazníky a hladování | Zachycení informací a trendů |
| Sdílené kalendáře | Koordinace aktivit |
| Odkazy na webové stránky | Poskytnutí zdrojů a referencí |

Existují také prostředky, které nemůžeme zařadit mezi čistě synchronní či asynchronní, příkladem tohoto stavu jsou například sociální sítě, příkladem nesynchronně přidávaného obsahu zde můžou být jednotlivé příspěvky uživatelů. Synchronní komunikace pak probíhá prostřednictvím chatů přidružených k jednotlivým sociálním sítím, mnohé z těchto služeb mají i vestavěné aplikace pro videohovory, jejich využití pro komunikaci je tak velice široké. V rámci školství mají ale sociální sítě využití spíše jako nástroj pro prezentaci a propagaci školy (Pilková, 2016).

## 1.3 Součást distančního vzdělávání v době pandemie

Podle novely školského zákona ze srpna roku 2020, která nabyla účinnosti v 25.8.2020 je distanční výuka v odpovídajícím rozsahu podle rámcového vzdělávacího programu a školního vzdělávacího programu plnohodnotnou náhradou výuky na základních, základních uměleckých a středních školách pro případ krizových opatření. Předpokládá se, že tato distanční výuka bude realizována on-line pomocí příslušných informačních technologií s případnými individuálními konzultacemi (Zákon 561/2004 Sb., 2004).

Pro vysoké školy platí obdobná úprava zavedená v zákoně o vysokých školách, dle § 95c uvedeného zákona je využití nástrojů distančního způsobu komunikace bez ohledu na formu studia studijního programu možné jak pro výuku, tak i pro konání zkoušek, včetně státních zkoušek. Tuto možnost lze opět využít při vyhlášení krizových opatření (Zákon č. 111/1998 Sb., 1998).

On-line výuka se stala součástí distanční výuky na většině stupních školského systému v roce 2020 v návaznosti na mimořádná opatření ministerstva zdravotnictví související se světovou pandemií koronaviru SARS-CoV-2. Tato opatření byla vydána kvůli podchycení epidemické situace v zemi. Poprvé v historii České republiky tak byla dlouhodobě celoplošně nasazena distanční výuka s využitím on-line prvků.

On-line výuka v českých školách nemá jednotnou formu, pro její vedení vydalo ministerstvo školství, mládeže a tělovýchovy pouze doporučení. Předpokládá se využití nějaké komunikační aplikace pro komunikaci mezi školou a žáky. V ideálním případě by přes tuto platformu mělo být možné jak jednoduše komunikovat, tak i vést videohovory, kterými je možné do jisté míry suplovat prezenční vyučování. Mezi doporučené aplikace pro zmíněné účely patří například aplikace Google Meet či Microsoft Teams (Doporučené postupy pro školy v období vzdělávání na dálku, 2020).

# 2 Komunikační aplikace pro distanční výuku

Následující kapitola se věnuje rešerši komunikačních aplikací používaných v rámci distanční výuky v období koronavirové pandemie. Klíčem pro výběr blíže zkoumaných aplikací využívaných při on-line výuce byla osobní zkušenost autora ať už z pohledu studenta (*Google Meet, Zoom*) či učitele (*Microsoft Teams*) a také doporučení vedoucího práce (*Jitsi Meet*).

Kromě přehledu aplikací a jejich možností slouží tato kapitola také jako pomůcka pro výběr komunikační aplikace zahrnuté v projektu, jehož tvorbě se autor věnuje v praktické části. Kritérii pro výběr vhodné aplikace jsou zejména možnost užívání aplikace zdarma po neomezenou dobu, možnost užívání mimo rámec profesní organizace a volně dostupné API pro vytváření videohovorů v dané aplikaci.

## 2.1 Google Meet

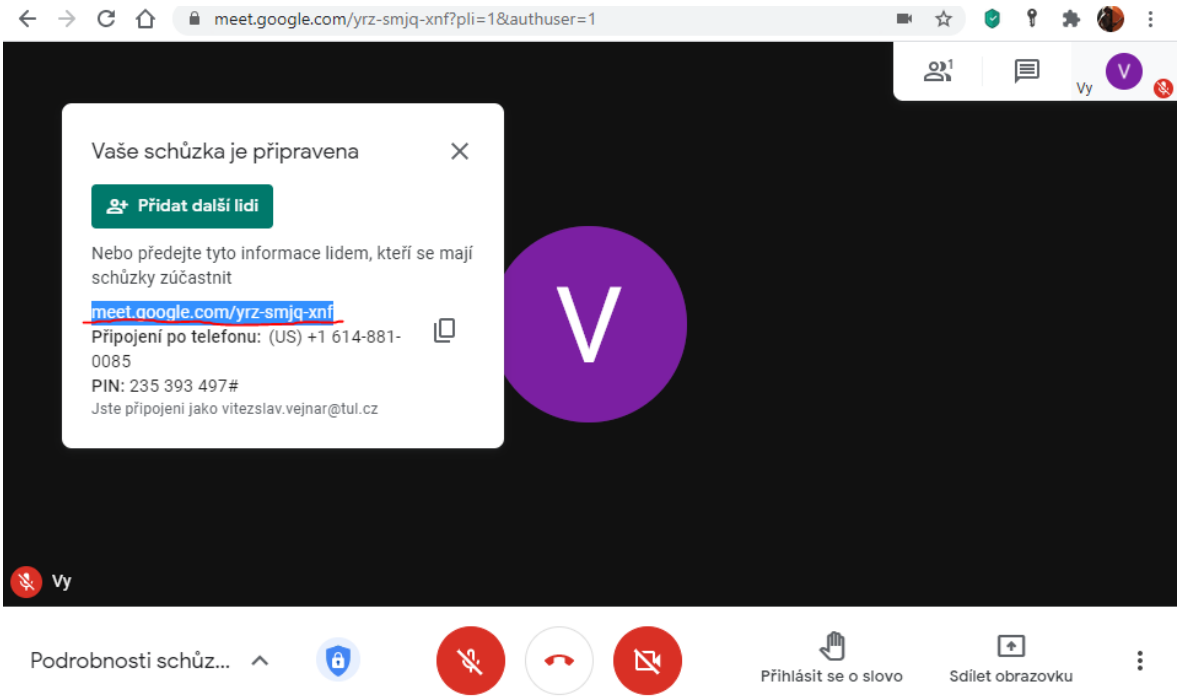
Aplikace dříve nazývaná Hangouts Meet nese svůj současný název od dubna 2020 a patří do rozsáhlé knihovny webových aplikací od firmy Google. Kromě webové verze existuje i mobilní verze této aplikace jak pro Android, tak pro iOS. Základní funkcí Google Meet je organizace videohovorů v oddělených místnostech. Za uplynulý rok tato aplikace získala na popularitě mimo jiné mezi učiteli, kteří ji začali používat pro účely distanční výuky.

V bezplatné verzi nabízí Google Meet funkce organizování skupinových schůzek do sta účastníků, přičemž každý hovor s více než dvěma účastníky může trvat maximálně jednu hodinu, hovory do dvou účastníků jsou omezené na délku 24 hodin. Bezplatné funkce zahrnují dále připojení z prohlížeče i výše zmiňované mobilní aplikace, titulky pro neslyšící a možnosti sdílení obrazovky a prezentování.

Předplatné přidává další funkce nad rámec bezplatných možností, Google v současnosti nabízí dva typy předplatného pro mimoškolní prostředí – Google Workspace Essentials určené pro malé týmy (8 USD/aktivní uživatel/měsíc – stav k únoru 2021) a Google Workspace Enterprise (cenu nutné dohodnout s prodejním týmem Google) pro korporátní prostředí (Ceny Služby Google Meet, 2021).

Pro vzdělávací instituce je Google Meet nabízen jako součást služby Google Workspace for Education s odlišnými možnostmi předplatného (od bezplatné až po 5 USD za studenta na měsíc). Ačkoli tento balík aplikací od Googlu dává vzdělávacím institucím větší možnosti než běžnému uživateli již v bezplatné verzi, za některé funkce Google Meet si organizace musí také připlatit.

Školské instituce mají již v základu možnost využívat funkce Google Meet prostřednictvím svých centrálních účtů (běžný uživatel musí využívat účet Google), délka hromadných videohovorů pro ně není omezena a mohou využívat například možností nahrávání a virtuální tabule. Za některé funkcionality, jako jsou dotazníky, prezenční listiny či možnost rozdělení účastníků do skupin přímo v hovoru, si ale i vzdělávací instituce musí připlatit (Google Workspace for Education editions, 2021).




*Obrázek 1: Prostředí hovoru Google Meet se zvýrazněným odkazem pro připojení (výstřižek z prostředí webové aplikace)*

Princip používání Google Meet je jednoduchý, pořadatel schůzky buď naplánuje nebo ihned započne schůzku buď skrze webovou aplikaci Meet či skrze mobilní aplikaci. Obě tyto akce vygenerují schůzku s unikátním označením, zpravidla desetimístným řetězcem písmen mezi něž jsou na dvou místech vloženy spojovníky. Zmíněné unikátní ID schůzky je pak možné sdílet s ostatními účastníky buď formou přímé pozvánky, anebo skrze jiné

komunikační služby, stačí aby se pak dotyční přihlásili do schůzky skrze odkaz ve tvaru meet.google.com/‘*označení schůzky*‘. V prostředí hovoru je pak možné využívat kromě samotného zvukového a videohovoru také chat a další funkce, jejichž rozsah se odvíjí od zvoleného předplatného.

Google dokumentaci API přímo pro Google Meet bohužel veřejně nepublikuje, k jednotlivým schůzkám je nicméně možné přistupovat skrz Google Calendar API, konkrétně skrze její část Events, která obsahuje veškerá specifika, která ke správě jednotlivých schůzek potřebujeme, společně s uceleným souborem příslušných metod. Skrze tuto API je možné nastavit potřebná uživatelská oprávnění schůzek a veškeré ostatní podrobnosti spojené s plánováním videohovoru (čas konání, přizvané hosty, apod.). API Google kalendáře je možné využívat prostřednictvím vývojářské konzole Google (Events | Calendar API | Google Developers, 2021).

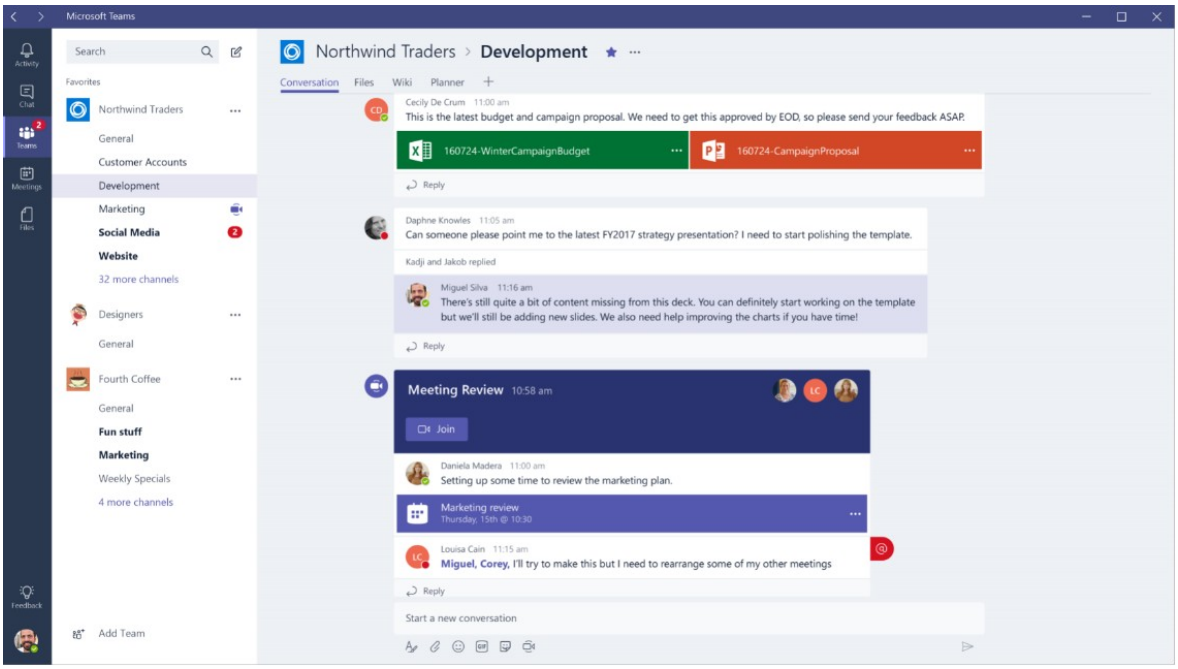
## 2.2 Microsoft Teams

Toto řešení pro týmovou spolupráci nabylo na popularitě ve školství podobně jako předchozí zmíněná aplikace zejména po startu pandemie covid-19. Oproti Google Meet přináší řadu výhod, ty je možné využívat i zdarma, nutná je pouze registrace na příslušných stránkách za pomoci účtu microsoft.

Aplikace Microsoft Teams je již ucelenějším řešením, funguje jak na počítačích skrze prohlížeč nebo ve formě desktopové aplikace, tak i na mobilních zařízeních využívajících operační systémy Android a iOS skrze příslušnou aplikaci. Funkce základních aplikací MS Office (*Word, Excel*, *Powerpoint*) je možné využívat přímo v rámci desktopové aplikace Teams, přičemž jednotlivé vytvářené soubory se ukládají na cloudové úložiště OneDrive (Microsoft Teams, 2021).

Práce skrze MS Office je organizována v týmech uživatelů, každý z týmů má k dispozici své úložiště souborů, kde je možné při online výuce například sdílet výukové materiály. V rámci týmů je možné také pořádat videohovory, plánovat jednotlivé schůze, organizovat zadání jednotlivých úkolů a poskytovat zpětnou vazbu. Kromě týmové spolupráce je možné využívat také chat pro individuální komunikaci. Díky větší provázanosti všech nástrojů nežli u balíku Google (abychom mohli využívat stejné funkce s Google Meet, musíme k němu používat také Google Chat a Google Classroom) je možné Microsoft

Teams považovat za poněkud ucelenější řešení (Hughes, 2020).. Pro školy, které nedisponují LMS může díky svým funkcím aplikace Teams sloužit jako uspokojivá alternativa.



*Obrázek 2: Uživatelské rozhraní MS Teams - Prostředí týmu (Dolejš, 2019)*

V základu je používání aplikace Teams zdarma, omezení vyplívající z neplacené verze spočívají zejména v omezené velikosti úložiště, v menším množství možností zabezpečení, které může uživatel využívat a v omezení délky hovorů. Pro podnikové použití jsou k dispozici tři možnosti předplatného (v ceně od 4,20 EUR do 19,70 EUR měsíčně za uživatele s ročním závazkem), které se vážou na předplatné kancelářského balíku Microsoft 365. V bezplatné verzi jsou pro tyto subjekty k dispozici pouze schůzky o délce do 60 minut s maximálním počtem 100 účastníků a není možné pořizovat záznamy schůzek. Cloudové úložiště je omezeno na 10 GB sdílených všemi týmy uživatele (Srovnání online možností Microsoft Teams | Microsoft Teams, 2021).

Pro vzdělávací instituce nabízí Microsoft aplikaci Teams bezplatně jako součást možnosti bezplatného využívání online verze balíku Microsoft 365, přičemž pro ně není omezena délka hovorů, nýbrž pouze kapacita cloudového úložiště OneDrive. Existují i placené programy Microsoft 365 pro školy, které rozšiřují knihovnu aplikací, které mohou studenti

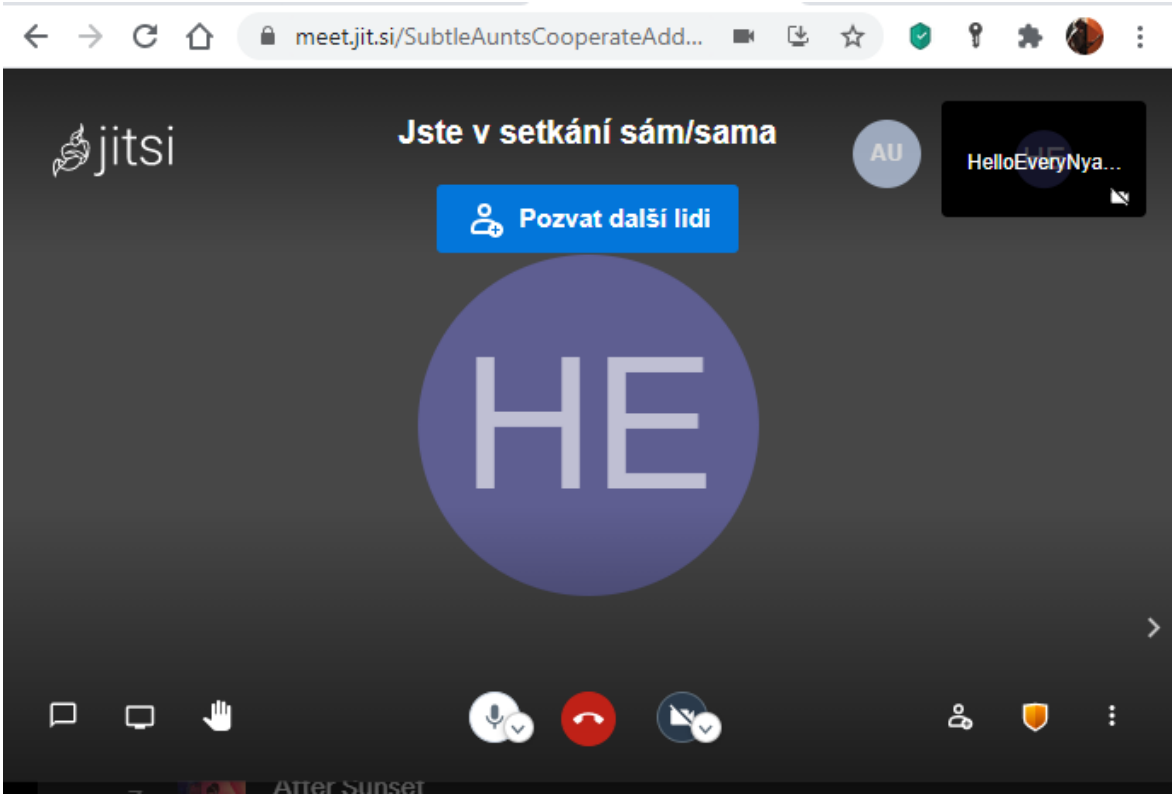
a učitelé využívat. V hovorech skrze aplikaci Teams lze využívat množství pomůcek, například sdílení obrazovky, digitální tabule či oddělených skupinových místností pro práci na skupinových zadáních (Microsoft Teams pro vzdělávání, 2021).

API pro práci s jednotlivými prvky MS Teams je součástí systému Microsoft Graph, který slouží jako přístupový bod pro práci s Microsoft 365. Skrze tuto rozsáhlou API je možné uskutečňovat většinu funkcí správy týmů, plánování, komunikace i videohovorů v rámci Teams. Pro jednotlivé metody v rámci této API existuje dokumentace v jazycích C#, Javascript, Objective-C a Java. Konkrétně k vlastnostem jednotlivých hovorů (schůzek) je přistupováno skrze zdroj *call* (Use the Microsoft Graph API to work with Microsoft Teams - Microsoft Graph v1.0 | Microsoft Docs, 2020).

## 2.3 Jitsi Meet

Méně známé řešení nabízí elegantní alternativu k aplikacím velkých korporací. V základu je Jitsi Meet odlehčenou aplikací, která nabízí základní nástroje pro potřeby videokonferencí. Verze zdarma má pro běžné uživatele výhodu například oproti Google Meet či MS Teams v tom, že hovory v ní nejsou časově omezené. Oproti tomu je toto řešení více omezené co se počtu uživatelů v rámci videokonferencí týká, je možné, aby se připojilo pouze 50 účastníků na jeden hovor.

Tuto platformu je možné využívat jak v prohlížeči, tak i na mobilních zařízeních ve verzích pro Android a iOS. Výhodou je, že skrze Jitsi Meet může sdílet obrazovku více uživatelů najednou, kalendář této platformy je možné také propojit se softwarem od Googlu i Microsoftu a také s platformou Slack, která v oblasti on-line kooperace také nabývá na popularitě (Jitsi Meet, 2020).




*Obrázek 3: Prostředí hovoru Jitsi Meet (výstřižek z prostředí webové aplikace)*

Mezi funkce, které jitsi nabízí, patří také tzv. end-to-end šifrování, tj. typ šifrování, který umožňuje zašifrování dat takovým způsobem, aby je mohlo přečíst pouze určité zařízení na druhé straně komunikace. Tato procedura umožňuje zajistit, aby odesílaná data nebylo možné odposlouchávat cestou k přijímači. Další výhodou je, že toto šifrování znemožňuje cílenou manipulaci s odesílanými daty během jejich cesty (What end-to-end encryption is, and why you need it, 2021).

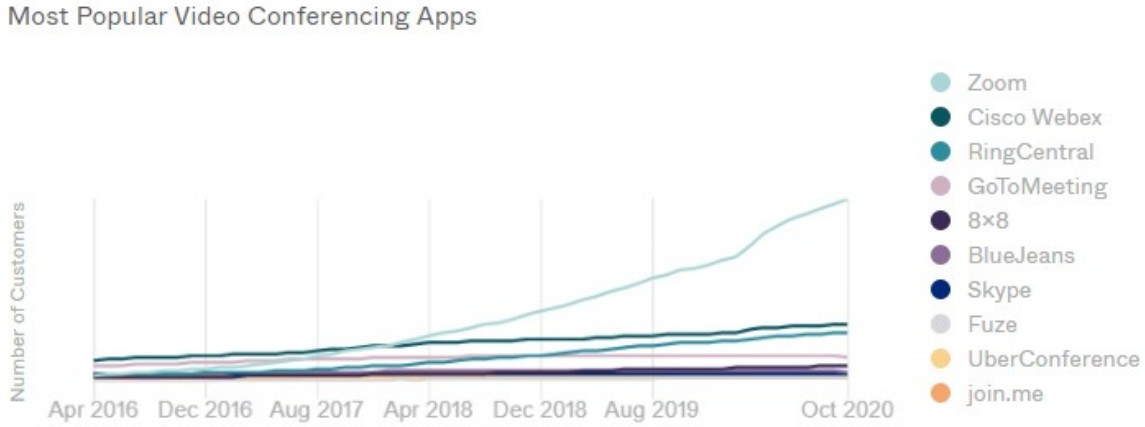
Placenou alternativou k Jitsi je aplikace 8x8 Meet, ta k možnostem zdarma přidává také možnosti kontaktování hostů po telefonu, pořizování záznamu hovoru, živé streamování na YouTube nebo rozšířené možnosti moderace hovorů či analytiky. Toto rozšíření lze pořídit za cenu začínající na 12 USD/měsíc na uživatele jako součást platformy pro komunikaci prostřednictvím chatu či videohovorů 8x8. Tato placená alternativa je cílená spíše na podnikové využití, pro využití ve vzdělávání nenabízí 8x8 žádné speciální předplatné (Voice, Video, Chat, Contact Center | 8x8, 2020).

Pro zájemce, kteří chtějí využívat platformu jitsi jako součást své aplikace, existuje možnost využití služby Jitsi as a service, jejíž cena se odvíjí od množství aktivních uživatel aplikace za měsíc, základní předplatné je možné pořídit v ceně 99 USD/měsíc s limitem 300 aktivních uživatelů za měsíc, pro účely vývoje existuje také bezplatná verze předplatného s limitem do 25 aktivních uživatelů měsíčně (Jitsi as a Service, 2020).

API pro aplikaci Jitsi Meet a návody k jejímu použití jsou k dispozici na dedikovaných stránkách v rámci repository github. Balík nástrojů umožňuje vývojářům vsadit hovory prostřednictvím platformy Jitsi přímo do jejich aplikace a konfigurovat jejich nastavení a vzhled GUI samotného prvku (Developer Guide (Web) · Jitsi Meet Handbook, 2021).

## 2.4 Zoom

Tato aplikace zaznamenala od počátku pandemie covid-19 obrovský nárůst počtu uživatel (jenom v regionu EMEA (Evropa, Střední východ a Afrika) zaznamenal v roce 2020 Zoom 227% nárůst počtu unikátních uživatel oproti předchozímu roku). Aplikace se dlouhodobě (již od roku 2018) pyšní také titulem aplikace pro videokonference s největším počtem uživatel, od března 2020 (začátek globální covidové pandemie) se stala také aplikací tohoto typu s největším počtem aktivních uživatel (Businesses at Work, 2021).




*Obrázek 4: Graf nejpopulárnějších aplikací pro videokonference podle celkového počtu uživatel (Businesses at Work, 2021)*

Aplikace Zoom zahrnuje mnoho funkcionalit potřebných pro videohovory již v základní bezplatné verzi, v mnohých ohledech nabízí více, nežli konkurenční aplikace – mezi základní funkce patří například vytváření oddělených místností v rámci hovoru, virtuální pozadí či oddělený chat, ve kterém mohou účastníci odesílat jak veřejné zprávy všem, tak

i soukromé pouze určeným uživatelům. Hlavním omezením aplikace Zoom v bezplatné verzi je, mimo limit 100 účastníků na hovor (který je vcelku vysoký v porovnání například s Jitsi Meet), časový limit 40 minut pro každý hovor. Placené verze začínají na ceně 139,90 EUR za roční licenci a zahrnují například více možností zabezpečení hovorů, cloudové úložiště nahrávek hovoru o kapacitě 1 GB (v bezplatné verzi je nahrávky možné ukládat pouze na lokální úložiště), navýšení možné délky hovoru až na 30 hodin nebo streamování prostřednictvím Zoom na sociálních sítích.

Zoom kromě několika základních plánů nabízí také řadu rozšiřujících funkcionalit například pro firmy pořádající webináře (navýšení počtu možných účastníků – hostů až na 1000 osob), pro lepší spolupráci a větší možnosti konferencí (aplikace pro dedikovaná zařízení pro účel videokonferencí – možnost vybavení zasedacích místností se speciálním hardwarem aplikacemi Zoom) nebo pro potřeby telefonování prostřednictvím aplikace (Zoom Video Conferencing Plans & Pricing, 2021).

Firma nabízí také plán nákupu licencí pro vzdělávací instituce, který je možné pořídit od ceny 84 EUR za licenci na rok, k dispozici je v počtu minimálně 20 licencí (minimálně tedy za 1 680 EUR/rok. Funkcionality předplatného pro vzdělávací instituce jsou srovnatelné s podnikovým předplatným, je možné taktéž využít různých přídavných balíčků k tarifu (Education Plan - Zoom, 2021). V souvislosti s pandemií koronaviru Zoom uvolnil pro školy v postižených regionech v několika zemích světa časový limit schůzek v bezplatné verzi aplikace, mezi tyto země patří i Česká republika. O odstranění limitu v rámci hovorů musí každá instituce sama požádat pomocí formuláře na internetových stránkách aplikace (Gallagher, 2020).

Díky prudkému nárůstu aktivních uživatelů se v roce 2020 dostalo aplikaci zvýšené pozornosti bezpečnostních expertů, což vedlo k několika poněkud kontroverzním zjištěním. Bylo zjištěno několik bezpečtnostních slabin, které mohly být využity ke krádežím hesel systému Windows či k nabouráním se do zařízení Mac a napojení se na jeho webkameru a mikrofon. Kromě nedostatků bezpečnosti vzbuzovaly obavy také obchodní podmínky, které byly ošetřeny natolik ve prospěch společnosti Zoom Video Communications, že jí dovolovaly téměř volně obchodovat s daty uživatel (Irwin, 2020).

Pomocí API aplikace Zoom lze využívat většinu funkcí její webové verze včetně kompletního souboru operací potřebných pro vytváření a správu schůzek. Součástí API je také generátor kódu pro velké množství programovacích jazyků včetně například PHP, Javascriptu nebo Pythonu. API Zoomu je tak přístupné pro velké množství implementací (Introduction to Zoom API, 2021).

# 3 Webové aplikace

V této kapitole se věnuji úvodu do problematiky webových aplikací, zejména jejich definici a prostředkům jejich fungování a tvorby.

## 3.1 Význam pojmu

Webovou aplikaci je možné definovat jako program, ke kterému má uživatel možnost přistupovat skrze prostředí internetového prohlížeče. Jedná se tedy o program uložený na vzdáleném serveru. Užití webových aplikací je velice široké, mohou sloužit potřebám jednotlivce i v podnikovém prostředí. Typickým příkladem může být například e-mailový klient ve webovém prohlížeči nebo internetový obchod.

Mezi benefity webových aplikací patří zejména jejich velká přístupnost. Nemusí být instalovány aby mohly být používány, zároveň k nim lze přistupovat skrze různá zařízení (počítače, mobilní zařízení…), většinou nejsou závislé na konkrétním prohlížeči a mohou tak být prohlíženy nezávisle na tom, který používáme. Oproti nativním aplikacím také nejsou vyvíjeny pro specifický hardware (Web application (Web app), 2019).

Mezi veřejností dochází často k zaměňování pojmů webová aplikace a web, tyto pojmy se od sebe nicméně liší. Web představuje soubor několika navzájem propojených webových stránek, které sdílejí doménové jméno. Může být uložen na jednom či více webových serverech. Webová aplikace může být součástí webu, web ale s žádnou webovou aplikací operovat nemusí. Rozdílné charakteristiky webů a webových aplikací můžeme vidět v tabulce níže.

*Tabulka 3: Rozdíly mezi webovou aplikací a webem (Difference between Website and Web Application, 2021)*

| Parametr | Webová Aplikace | Web |
|---|---|---|
| Účel | Vytvořena za účelem interakce s uživatelem | Prezentace vlastního obsahu, z většiny statického |
| Uživatelská interakce | Uživatel může manipulovat s daty pomocí webové aplikace | Web poskytuje multimediální a textový obsah, který může uživatel vidět, ale nemůže ho ovlivňovat |

| Parametr | Webová Aplikace | Web |
|---|---|---|
| Ověření uživatele | Webové aplikace často požadují ověření uživatele | Pro weby bez webových aplikací není ověření uživatele důležité |
| Komplexnost | Nabízí složitější funkce, nežli pouhý web | V základu pouze zobrazuje data |
| Ucelenost softwaru | Webová aplikace není sama o sobě kompletní, k fungování potřebuje web | Web je kompletním produktem zobrazitelným skrze prohlížeč |
| Spuštění provozu | Každá změna vyžaduje, aby celý projekt byl rekompilován. | Malé změny nepotřebují rekompilaci, stačí změna HTML kódu |

## 3.2 Prostředky fungování webových aplikací

Všeobecně se dá říci, že webové aplikace využívají prostředků webového serveru, aplikačního serveru a databáze, každá z těchto součástí má v jejich běhu vymezenou roli (Web application (Web app), 2019).

### 3.2.1 Webový a aplikační server

Pro fungování webových aplikací slouží webový server jako úložiště zdrojového materiálu stránek, na kterých je aplikace implementována. Dodává statický obsah (například HTML stránky, soubory, obrázky či video) jako odpověď na HTTP požadavky ze strany webového prohlížeče.

Aplikační server představuje prostředí vývoje a běhu webové aplikace, může sloužit stejným účelům, jako webový server, jeho hlavní smysl nicméně spočívá ve zprostředkování interakce mezi klientem a kódem aplikace na straně serveru – podporuje tak tvorbu dynamických webových stránek. Kromě dynamického obsahu (ten může představovat výsledky jednotlivých transakcí) může aplikační server v odpověď na požadavek klienta také poskytovat i nástroje pro analytiku. Klienta aplikačního serveru může kromě webového prohlížeče představovat také vlastní uživatelské rozhraní aplikace. V současnosti jsou webový a aplikační server většinou součástí jednoho celku, většina webových serverů totiž v dnešní době podporuje různé moduly pro skriptovací jazyky, na druhé straně se také zvyšuje množství aplikačních serverů, které používají HTTP jako svůj primární protokol.

Mezi nejpopulárnější open-sourcové webové a aplikační servery patří například Nginx (v prosinci roku 2019 ho využívalo 25 % z milionu nejpoužívanějších webových stránek, využívají ho například aplikace Netflix a Dropbox), Apache HTTP Server (byl nejvyužívanějším webovým serverem do doby nástupu Nginx), Apache Tomcat (nejvyužívanější aplikační server na bázi jazyka Java) a Glassfish (Web Server vs. Application Server, 2020).

### 3.2.2 Databáze

Databáze spravují data potřebná pro běh aplikace a zároveň slouží jako úložiště aplikací generovaných dat. Představují sbírku informací či dat strukturovaných podle určitého klíče. V dnešní době je většina databází modelována jako soubor tabulek, ve kterých jsou data uložena v řádcích rozdělených do sloupců, které představují jejich jednotlivé charakteristiky (tomuto typu databáze se říká relační databáze), nicméně existují i další typy databází, které mohou být strukturovány jiným způsobem. Pro komunikaci s téměř všemi relačními databázemi je využíván jazyk SQL, jeho prostřednictvím lze pomocí jednotlivých výrazů (*query*) vybírat, tvořit či měnit data (What Is a Database?, 2021).

### 3.2.3 Srovnání s třívrstvou architekturou

Potřebné prostředky pro fungování webové aplikace tak zrcadlí třívrstvou architekturu. Prezentační vrstvu zde představuje webový server, který dodává statická data a tím vytváří vlastní UI na straně klientu – tedy webového prohlížeče, mobilní aplikace nebo desktopové aplikace. Aplikační (logickou) vrstvu, ve které se zpracovávají data získaná v prezentační vrstvě, představuje aplikační server a datovou vrstvu, kde jsou spravována a ukládána data, představuje databáze. V třívrstvé aplikaci jsou data vždy zpracovávána aplikační vrstvou, datová a prezentační vrstva k sobě navzájem nemají přímý přístup (Three-Tier Architecture, 2020).

## 3.3 Prostředky vývoje webových aplikací

Když uvažujeme vhodnost prostředků vývoje, můžeme si je opět rozdělit podle třívrstvé architektury. Její jednotlivé části tak slouží jako klíč k dělení následujících podkapitol.

### 3.3.1 Prezentační vrstva

Tato část třívrstvé architektury je složkou přímé komunikace s uživatelem, tvoří uživatelské rozhraní webové aplikace. Nejběžnějšími nástroji pro práci na prezentační vrstvě webů (front-endový vývoj) jsou *HTML*, *CSS* a *JavaScript* (Three-Tier Architecture, 2020).

Jazyk HTML je používaný pro popis struktury obsahu webové stránky. Jednotlivé části obsahu jsou v něm označovány pomocí takzvaných *tagů* (značek). Různé druhy tagů jsou využívány k určování prvků na stránce. Můžeme říct, že jejich prostřednictvím tvoříme kostru webu. Kromě určení struktury webu určujeme pomocí HTML také vztahy mezi jednotlivými prvky na stránce (například když použijeme tag *figure* pro zastřešení obrázku a jeho popisku a *figcaption* pro samotný popisek, určujeme tím, že daný obrázek s popiskem spolu souvisí) (Vodnik, 2017).

CSS ovlivňuje vzhled jednotlivých stavebních bloků stránek – jednotlivých prvků. Ovlivňuje jejich rozvržení, formát a celkovou prezentaci. Jednotlivá pravidla tohoto jazyka nastavují vzhled ať už jednotlivým prvkům určeným různými *tagy,* skupinám prvků či celým dokumentům nebo jejich souborům. Prostřednictvím CSS můžeme měnit základní hodnoty jako například barvy nebo typy písma, ale také s jeho pomocí můžeme v omezené míře pomocí tzv. pseudotříd definovat chování jednotlivých prvků na stránce či zajistit responzivní chování stránek.

Jazyk JavaScript je využíván zejména pro definování chování webu v návaznosti na akce uživatele, dovoluje tak webovým vývojářům navrhovat interaktivní webové stránky s dynamickým chováním. Příkladem prvku vytvořeného pomocí tohoto jazyka může být například výzva k potvrzení, která se objeví po zadání dat do on-line formuláře, ve speciálním boxu. Využití JavaScriptu je nicméně velice široké a dovoluje nám mnohem více, nežli samotné HTML a CSS. Kromě definování chování stránky a vytváření dynamického obsahu můžeme s jeho pomocí také ukládat různá data o uživateli (Kolowich Cox, 2020). Některé z funkcí JavaScriptu mohou být nahrazeny například jazykem PHP. Operace jazyka PHP probíhají na straně serveru, kdežto operace JavaScriptu na straně klienta (webový prohlížeč) (PHP vs JavaScript, 2021).

### 3.3.2 Aplikační a datová vrstva

Pro aplikační vrstvu jsou typicky využívané jazyky jako *Python*, *Java*, *PHP* nebo *Ruby*, typicky tato vrstva komunikuje s datovou vrstvou za využití volání jednotlivých funkcí API. Tato vrstva, která je vložená mezi prezentační a datovou vrstvu, může být považována za jádro webové aplikace, jsou v ní zpracována data, která byla získána od uživatele v rámci prezentační vrstvy.

V datové vrstvě jsou ukládána a spravována data zpracovaná aplikací. Tuto vrstvu může představovat relační databáze (například *PostgreSQL*, *MySQL, MariaDB, Oracle*). Mezi další typy možných databází patří tzv. *NoSQL* databáze (*Cassandra*, *CouchDB*, *MongoDB*), pro manipulaci s daty těchto databází není nutné používat jazyk SQL, existují pro ně i jiné více flexibilní prostředky (No v jejich názvu znamená.”not only”, tedy nejenom) (Three-Tier Architecture, 2020).

# 4 Aplikace pro time-management on-line výuky

Touto kapitolou začíná praktická část práce. Věnuje se konceptu vyvíjené aplikace, způsobům jejího využití, hlavním uživatelským aktivitám a představení ukázkového modulu pro demonstraci možného způsobu realizace aplikace.

## 4.1 Námět pro vznik aplikace

V situaci nutné on-line výuky, která nastala během pandemie viru Covid-19, se učitelé každodenně potýkají s velkým množstvím aplikací pro podporu komunikace na dálku a distanční výuky. Některé z aplikací umožňují práci s rozvrhem, ukládání dokumentů určených jako učební podpora a poskytují nástroje pro plánování a konání videokonferencí. Tyto programy nicméně často mívají licenční podmínky ošetřené tak, že jsou přístupné k bezplatnému využívání pouze školským institucím (například *MS Teams*). Jejich řešení v prohlížeči jsou často složitá nebo mají omezené funkcionality oproti desktopovým verzím. Zároveň možnosti integrace těchto aplikací se stávajícími LMS bývají omezené.

Aplikace, jejímž návrhem se zabývá tato práce, se zaměřuje jednak na řešení těchto problémů, ale také může sloužit jako základ pro systém využívaný k rychlému sdílení znalostí mezi soukromými lektory a potenciálními zájemci. Tento problém je v rámci momentálně dostupných prostředků řešený pouze okrajově a dostupné nástroje nedosahují efektivity uzavřeného řešení. Náš výstup by tak měl představovat jednoduchý systém, který zefektivní proces organizace on-line výuky a umožní co nejplynulejší domluvu termínů konzultací/hodin mezi učitelem a žákem.

Tato aplikace pak může nalézt využití jednak v rámci organizace soukromých lekcí, ale i jako součást systémů podpory výuky vzdělávacích institucí.

## 4.2 Definice možných způsobů využití

Předpokládejme aplikaci zaměřenou primárně na organizaci kurzů, v níž na jedné straně stojí nabídka kurzů (lektoři či vzdělávací instituce) a na straně druhé poptávka po vzdělání (potenciální účastník kurzu). V rámci této kapitoly definujeme jednotlivé možnosti využití dané aplikace zmíněnými subjekty.

### 4.2.1 Uživatelské role

Navrhovaná aplikace předpokládá v základu dvě uživatelské role a jednu roli administrátorskou. Dvě základní role představují prodejce kurzů, neboli lektora, a studenta, tedy příjemce vzdělávání. Zástupci obou těchto rolí se účastní dialogu potřebného pro naplánování on-line lekcí. Student má oprávnění k procházení nabídky vypsaných kurzů a k registraci sebe sama na ně. Může také prostřednictvím uživatelského rozhraní zasílat žádosti o individuální hodiny lektorům nabízených kurzů.

Oprávnění lektora jsou založena na oprávněních studenta, k nimž se přidávají další pověření související se správou a vedením kurzů. Lektor tak nemusí jednotlivé kurzy pouze vypisovat a vést, ale může vystupovat i jako student v rámci kurzů a hodin, které nebyly vypsané jím samotným. Pokud je uživatelský účet zařazen jako lektor, nabývá pravomoci vytvářet nové kurzy, modifikovat jejich popis a vlastnosti (pouze u kurzů vypsaných přihlášeným uživatelem). Ke každému kurzu má také možnost vytvářet lekce, které může vytvořit na základě žádosti potenciálního studenta, anebo v rámci vlastní iniciativy (možnost vhodná například pro hromadné kurzy).

Všichni uživatelé mají možnost také měnit údaje ve svém uživatelském účtu a spravovat vlastní kalendář, ve kterém se objevují hodiny, na které se uživatelé přihlásili nebo, v případě lektorů, hodiny, které sami vyučují.

Administrátor aplikace má kromě všech oprávnění studentů a lektorů také přístup k funkcím umožňujícím celkovou moderaci provozu aplikace. Je tak oprávněn i mazat uživatelské účty, měnit údaje jednotlivých uživatelů, manipulovat s údaji vypsaných kurzů či zapisovat studenty na kurzy.

### 4.2.2 Cílové skupiny a způsoby využití aplikace

Při vymezení cílových skupin navrhované aplikace se můžeme řídit již výše definovanými uživatelskými rolemi. Jako student může působit kdokoliv se zájmem o další vzdělání zprostředkovávané skrze danou platformu. Způsob užívání se u jednotlivých studentů pak bude lišit podle toho, zdali jsou studenty nějaké instituce využívající naši aplikaci pro správu výuky, anebo vyhledávají-li kurzy pro svůj volný čas na vlastní pěst. Začněme tak vymezením způsobů využití pro jednotlivé subjekty vystupující jako nabídka vzdělávání uvnitř aplikace.

### *Soukromí lektoři*

Aplikace může sloužit jako místo, kde mohou nabízet své lekce soukromí lektoři na volné noze. Pokud bychom při vývoji upřednostnili tuto skupinu, cílem by bylo zaměřit se co nejvíce na komunikaci mezi zájemci o kurzy a nabídkou.

Vzhledově a funkčně bychom se tak mohli přiblížit e-shopům či on-line tržištím, kde je stěžejní zejména správa nabízených produktů (v našem případě kurzů) na straně prodávajícího. Na straně kupujícího je pak důležité vytvořit přehledné rozhraní pro výběr kurzů a koupi předplatného lekcí.

Při využití aplikace tímto způsobem je důležité také zajištění dostatečné moderace kurzů ze strany provozovatele on-line tržiště nebo systému hodnocení jednotlivých lektorů samotnými žáky, aby byla zajištěna kvalita nabízeného obsahu a zamezilo se tak negativnímu pohledu na prostředníka (firma spravující aplikaci) jakožto zprostředkovatele potenciálně nekvalitních služeb.

Hlavním zaměřením aplikace se při využití soukromými lektory stává rezervace on-line hodin na základě žádosti studenta. Je tak nutné zajistit, aby bylo možné zobrazovat vytíženost jednotlivých lektorů při výběru termínu a následně možnost přijmout nebo odmítnout požadavek na straně lektora, případně nabídnout alternativní termín. Dalším důležitým krokem je také integrace platební brány a vytvoření vhodného schématu plateb, například v podobě předplaceného kreditu, který by uživatelé využívali pro platbu jednotlivých lekcí a který by byl stržen až ve chvíli odsouhlasení lekce lektorem. Správce kurzového tržiště by pak z jednotlivých transakcí mohl inkasovat provizi.

### *Instituce prodávající kurzy*

Některé organizace mohou preferovat více uzavřený přístup při prodeji svých služeb, v jejich případě by se navrhovaná aplikace mohla stát součástí e-shopu nabízejícího vzdělávací služby. Obsah a přístup jednotlivých lektorů by v tomto případě byl ovlivňován institucí vlastnící webové stránky. Při zakomponování aplikace do systému tohoto typu se stává důležitým i zabezpečení omezeného přístupu k vytváření lektorských účtů – ty by měl být schopen vytvořit pouze administrátor systému, každý vyučující by měl při svém nástupu takový účet získat předpřipravený.

Omezení přístupu na pouze ověřené lektory je tedy pro tuto variantu řešeno přímo na straně instituce poskytující kurzy. Systém uživatelských hodnocení jednotlivých lektorů a kurzů nicméně stále nachází uplatnění při rozhodování, zda chce daná instituce daného lektora nadále zaměstnávat. Správa kurzů by mohla být přizpůsobena potřebám jednotlivých organizací, pokud volí více centrální přístup, mohou umožnit vytváření nových kurzů ve svém systému pouze ze strany administrátora, pokud ovšem netrvají na absolutní kontrole, je možné vytváření kurzů i prostřednictvím lektorských účtů. Správa kurzů a změny jejich údajů by v rámci efektivity využívání systému měly být umožněny přímo lektorům.

Další nutnou funkcionalitou by se pro potřeby pokrytí nabídky vzdělávacích institucí stalo zajištění hromadných hodin, které v koncepci on-line tržiště nepředstavovaly tak důležitý prvek. Při vytváření kurzu před jeho publikací lektor určuje, zdali se jedná o kurz individuální či hromadný, případně na kolik je omezen počet návštěvníků hodin daného kurzu. Předpokládá se, že hromadné kurzy mají určitý sylabus, podle kterého se budou řídit, je tak pro ně nutné stanovit také začátek kurzu, do kdy se může účastník sám zapisovat nebo rušit svoji účast. Každou hodinu hromadných kurzů poté vypisuje sám lektor, načež je na e-mailové adresy žáků zasláno upozornění o nové události v rámci kurzu. Individuální kurzy by mohly probíhat stejným principem, jako bylo popsáno u předchozí varianty.

Integrace platební brány ve webovém rozhraní aplikace je v tomto případě také nutná. Vzhledem k systému prodeje kurzů jako celku a ne pouze jednotlivých hodin můžeme přistoupit i k variantě úhrady peněžních prostředků přímo za konkrétní kurz, dříve navržené využití systému uživatelského kreditu by bylo pro tento způsob provozu také možné.

### *Školy*

Navrhovaná aplikace může nalézt uplatnění i v prostředí, které není primárně orientované na zisk. Pro školy může sloužit jako doplněk stávajícího LMS či jeho částečná alternativa. Pokud bychom aplikaci měli integrovat do školní sítě tímto způsobem, bude potřeba vysoce omezit možnost přístupu k ní. Uživatelské účty, ať už lektorů kurzů či jednotlivých studentů, by měl mít práva zakládat pouze pověřený administrátor aplikace či správce školní sítě.

Využitelný je jak modul individuálních, tak i hromadných lekcí, přičemž můžeme předpokládat, že více využívané budou skupinové lekce. Rezervace individuálních termínů může být využita například pro plánování on-line konzultací.

Aby mohla koncipovaná aplikace a systém, do kterého bude integrována, být využívána jako odlehčený LMS, bylo by nutné vyřešit také funkce, které nabízejí takové systémy. Mezi tyto funkce patří například sdílení podkladů pro výuku se studenty, zadávání úkolů, možnost nahrávání žáky vypracovaných úkolů či vytváření on-line testů. Jako vhodnější se tak možná jeví integrace navržené aplikace do provozu nějakého školou již využívaného LMS nebo jiného široce rozšířeného řešení tohoto typu. Je možné také pojmout realizaci aplikace pouze jako rezevační systém pro jednotlivé lekce a konzultace.

Při využití pro školy již není nutné řešit integraci platební brány. Během vytváření takto uzavřeného systému předpokládáme, že realizace plateb přímo v něm nebude potřeba. Pokud by škola zároveň poskytovala vzdělávání za úplatu, bylo by možné navrhnout pro tento účel oddělený modul.

### *Studenti*

Jak už bylo zmíněno výše, způsoby využití navrhované aplikace ze strany studentů se úzce odvíjí od povahy instituce, která nabízí kurzy. Studenti vyhledávající kurzy soukromých lektorů a ti, kteří využívají nabídky některé z institucí nabízejících on-line vzdělávání primárně vyhledávají z vlastní iniciativy způsoby rozvoje svých znalostí. Nabízí se úvaha, nebylo-li by vhodné integrovat interní kalendář aplikace s nějakým široce využívaným on-line organizérem. Mohli bychom totiž předpokládat, že tito potenciální zákazníci budou mnohdy zaměstnanými lidmi, kteří potřebují mít ucelený přehled o svém časovém rozvrhu.

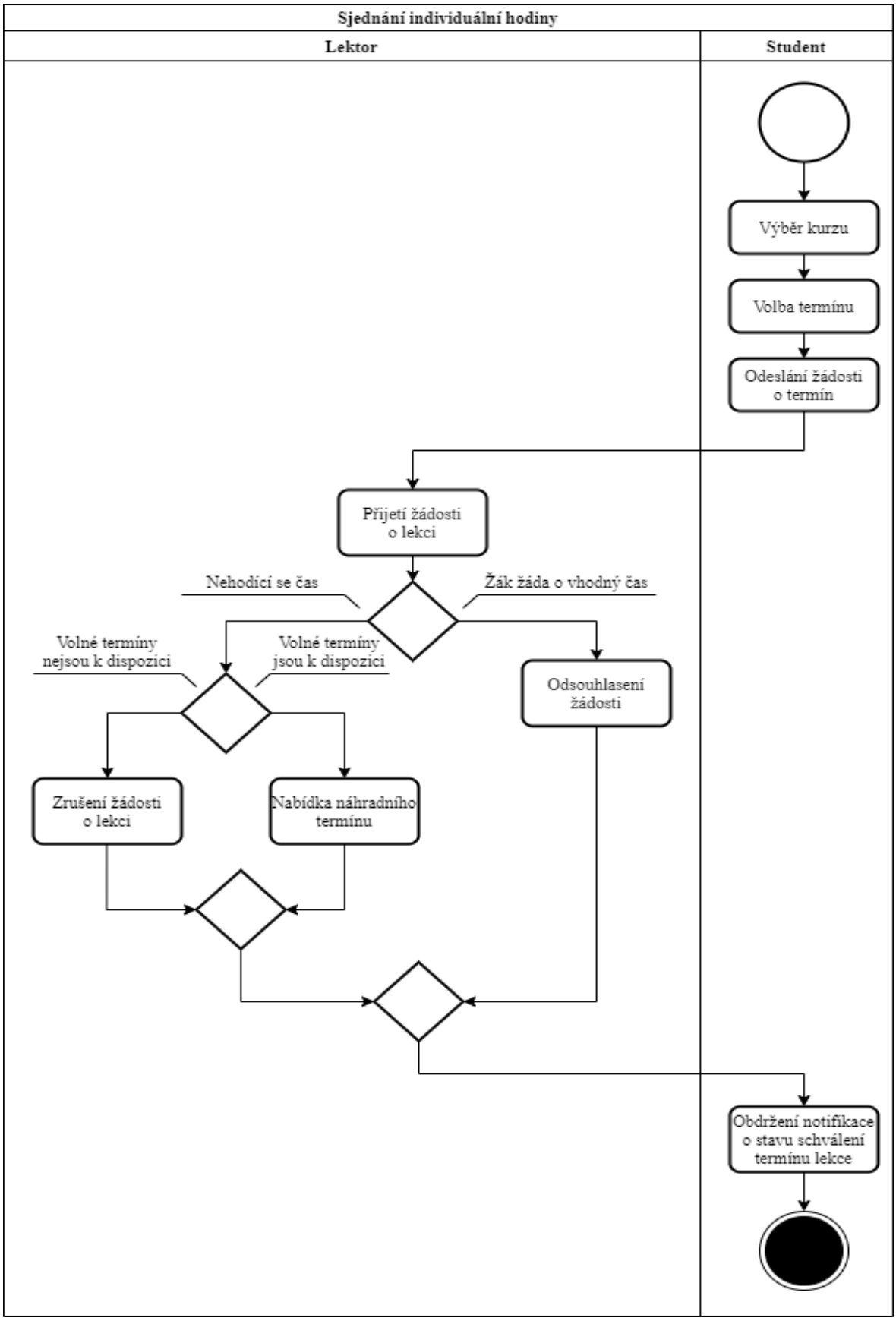
Co se týče studentů ve školách (které primárně nenabízejí kurzy za úplatu), můžeme předpokládat, že jejich hlavním důvodem využití našeho systému bude přístup k rozvrhu výuky a jednoduché přihlášení do naplánovaných hodin. Klíčový prvkem výuky na školách je také komunikace mezi učitelem a studenty. Pokud by tak vyvíjená aplikace nebyla integrována jako součást nějakého LMS, které umožňuje efektivní komunikaci mezi vyučujícím a členy kurzu, mohli bychom ji rozšířit o modul umožňující vhodný druh kontaktu (instant messaging, možnost zasílání e-mailů skrze formulář, apod.).

## 4.3 Aktivity uživatelů v rámci aplikace

Předpokládáme, že uživatelé budou v rámci aplikace provádět dvě hlavní aktivity podle svých preferencí. První z těchto činností je realizace soukromé hodiny, chcete-li konzultace. Druhou aktivitou je pak organizace lekcí skupinového kurzu.

Obě ze zmíněných aktivit mají společný předpoklad, lektoři musí před jakoukoli akcí ze strany studenta zadat do aplikace příslušný kurz. Aby mohl úplně nový uživatel vytvořit svůj kurz, musí si při prvním vstupu do aplikace vybrat roli lektora, která přisoudí jeho účtu příslušná oprávnění. Jakmile proběhne inicializace uživatelské role, je pro uživatele možné přejít k vytvoření a konfiguraci kurzu. Založení kurzu je dvoukrokové, přičemž v prvním kroku uživatel volí pouze název kurzu a zadává popis, který později vidí potenciální studenti. V druhém kroku lektor určí druh kurzu a následně doplní potřebná data podle zvoleného způsobu výuky (*hromadný* nebo *individuální*). Jakmile jsou zadána všechna potřebná data, je možné nabídku kurzu publikovat.

Pokud si uživatel při prvním příhlášení do aplikace vybere roli studenta, může ji začít využívat k vyhledávání kurzů a přihlašování se na ně. Prvním krokem studenta je tedy vyhledání kurzu, který ho zaujme. Dále se postup liší pro individuální a skupinově vyučované lekce. Při vyžádání termínu pro lekci jeden na jednoho (viz Obrázek 5) si student s využitím svého osobního kalendáře přímo v aplikaci zvolí termín, který mu vyhovuje a odešle automatizovanou žádost lektorovi daného kurzu. Lektor si následně žádost zobrazí ve své e-mailové schránce a prostřednictvím přiloženého odkazu ji může buď odsouhlasit, zamítnout, nebo navrhnout jiný termín. Pokud termín učiteli vyhovuje, zvolí možnost potvrzení lekce, dá vytvářené události jméno a odsouhlasí ji, načež je vytvořen odkaz na schůzku ve webovém rozhraní komunikační aplikace, který je poskytnut oboum stranám.




*Obrázek 5: Diagram aktivit pro proces smluvení termínu individuální lekce*

Když termín učiteli nevyhovuje, může zvolit buď možnost navržení jiného termínu, nebo úplného zamítnutí žádosti. Při zamítnutí žádosti schvalovací proces končí neúspěchem a student dostává zprávu o zrušení žádosti o lekci, která může být doplněna o odůvodnění ze strany lektora. Rozhodne-li se lektor pro variantu navržení jiného termínu, vybere prostřednictvím svého kalendáře v aplikaci jiný termín nebo termíny a odešle automaticky generovanou zprávu studentovi, který jeden z termínů může zvolit, načež se o této skutečnosti odešle notifikace lektorovi, který hodinu potvrdí a připraví on-line lekci. Další možností je pro studenta zamítnutí nabízených termínů, při kterém končí schvalovací proces neúspěchem a lektorovi je odeslána notifikace o zamítnutí nabídnutých termínů.

Skupinové kurzy vyžadují méně komunikace mezi lektorem a žákem, předpokládá se u nich, že termíny hodin se budou pohybovat v rozmezí určitých předem stanovených časů. Pokud si student vybere tímto způsobem vyučovaný kurz, jedinou nutnou aktivitou z jeho strany je se na daný kurz zapsat. Jakmile tuto akci provede, objeví se lektorovi na seznamu účastníků kurzu. Po ukončení období zápisu (předpokládaně v nějaký pevně stanovený časový úsek před začátkem kurzu) se znemožní přihlášení na kurz dalším studentům a zároveň je umožněn výpis lekcí kurzu lektorem. Ten ve svém kalendáři v aplikaci nadefinuje termíny hodin v předem smluveném časovém úseku, které po dokončení potřebných nastavení odešle, čímž se propíšou do kalendářů účastníků a lektora samotného. Společně s uložením termínů do kalendářů účastníků se také vytvoří odkaz na kurzem využívanou místnost v komunikační aplikaci, který je posléze k termínům přiřazen a zobrazen v interním kalendáři aplikace, odkud mohou uživatelé do jednotlivých hodin vstupovat.

Stejné činnosti jako studenti (*zapisování se na kurzy*, *žádosti o individuální lekce*) mohou provádět i lektoři v případě kurzů, které vypsali jiní učitelé. Takto nadefinované aktivity se vztahují zejména na využití aplikace pro propojení institucí či fyzických osob nabízejících vzdělávání za úplatu se zájemci o tento druh výuky v on-line prostředí.

## 4.4 Ukázkový modul aplikace

Pro demonstraci možného využití navrhované webové aplikace byl vyvinut jeden z předpokládaných modulů primárně zaměřený na organizaci soukromých hodin či individuálních konzultací. Umožňuje založení kurzu lektorem, následné zadání jednotlivých specifikací vyučované látky a parametrů kurzu. Studenti (potažmo lektoři

v případě cizích kurzů) se mohou do jednotlivých kurzů přihlašovat po jejich vyhledání prostřednictvím příslušné nabídky. Pro hromadné kurzy funguje systém zápisu žáků, u kurzů vyučovaných systémem jeden na jednoho je možné zažádat o termín lekce.

Schvalování individuálních lekcí probíhá prostřednictvím e-mailové komunikace, která je aplikací automatizována. Jakmile lektor odsouhlasí požadovaný termín, lekce se propíše do kalendářů účastníků. Po odsouhlasení poptávky hodiny je naplánován hovor prostřednictvím komunikační aplikace, na který je poptávajícímu i lektorovi poskytnut odkaz v rozhraní aplikace a také pomocí notifikačního e-mailu. Jakmile je čas se k hovoru připojit, je možné využít kromě UI aplikace také přímo rozhraní systému pro videohovory.

Rozhraní aplikace i její kód jsou psány v angličtině, aby mohly sloužit jako reference i případným zájemcům neovládajícím český jazyk. Jednotlivá specifika vývoje tohoto ukázkového modulu a návrhy na jeho další rozšíření jsou předmětem následujících kapitol.

## 4.5 Výběr nástroje pro videohovory

Při výběru nástroje využívaného pro samotné vyučování skrze hlasový či videohovor, na který by měla koncipovaná aplikace odkazovat, bylo rozhodnutí uskutečněno zejména podle dostupnosti daného nástroje pro běžného uživatele bez nutnosti využívání premiového předplatného. Cílem bylo zpřístupnit užívání aplikace široké veřejnosti. Při analýze komunikačních aplikací v kapitole 2 víme, že pro bezplatné použití jsou otevřeny všechny zkoumané aplikace s omezeními zejména v podobě omezení délky hovoru. Ze srovnání podle tohoto parametru nejhůře vyšla aplikace Zoom s limitem 40 minut na jednu schůzku v bezplatném režimu, tato doba je pro účel navrhované aplikace bohužel nedostačující, a tak byla možnost využití nástroje Zoom zamítnuta.

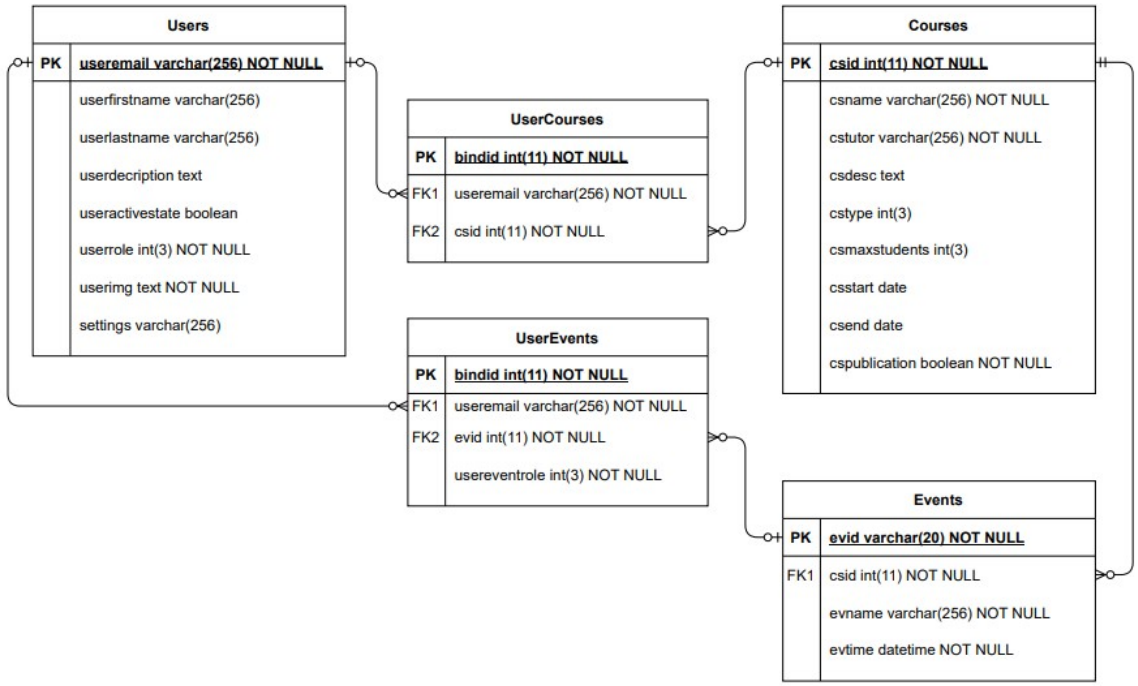
Nejlépe ze srovnání podle povolené délky hovorů při bezplatném využití se ukázal Jitsi Meet, nicméně stejně jako aplikace Zoom byl i tento nástroj nakonec zamítnut, jelikož při jeho využití by musely pro integraci s uživatelským kalendářem být využity nástroje třetí strany. Zároveň je Jitsi Meet při využití v externích aplikacích koncipovaný spíše jako nástroj, který se vkládá přímo do UI vyvíjeného řešení. V rámci vytvářené aplikace není záměrem hovory přímo realizovat, pouze na ně odkazovat, je to tak další z důvodů, proč byl Jitsi Meet vyřazen z výběru potenciálně využitelných řešení.

Do finálního výběru tak postoupily nástroje Google Meet a Microsoft Teams, oba tyto nástroje mají v bezplatných verzích časový limit pro videohovory stanovený na 60 minut (během pandemie viru Covid-19 byl tento limit u obou aplikací v průběhu času navyšován nebo dokonce rušen, limit 60 minut platí tedy pro období před pandemií). Jak nástroj od společnosti Google, tak i alternativa od Microsoftu nabízí velké množství nástrojů pro podporu výuky a jejich využití je možné kombinovat i s nástroji pro sdílení výukových materiálů či kooperaci v reálném čase (v případě *Google Meetu* nástroje *Google Workspace; MS Teams* má tyto nástroje přímo integrované v rámci desktopové aplikace, anebo webového prostředí). Obě řešení mají také možnost autentizace uživatele pomocí uživatelského účtu, který umožňuje využívání více nástrojů daného ekosystému najednou. *Google Meet* je nicméně primárně určen pro využití v prostředí prohlížeče (nebo skrze mobilní aplikaci), zatímco u *MS Teams* jsou funkce webové aplikace oproti její desktopové variantě omezené. Co se týče vhodnosti API obou nástrojů, potřebné nástroje pro propojení s navrhovanou aplikací obsahují obě dvě rozhraní.

Po srovnání obou možných variant byla zvolena aplikace *Google Meet.* Důvodem k jejímu výběru byla jednak její dobrá dostupnost prostřednictvím webového prohlížeče a možnost využití účtu *Google* přímo v rámci vyvíjené aplikace. Předpokládáme totiž, že existuje více potenciálních uživatelů naší platformy, kteří již mají účet *Google,* nežli těch s účtem *Microsoft*. Při vývoji využivajícím *Google API* můžeme využívat také zdroje velkého množství dalších aplikací (*Gmail, Google Kalendář, ...*).

# 5 Databáze aplikace

Tato kapitola se podrobně věnuje rozvržení databáze ukázkového modulu aplikace. Zmíněná databáze slouží k ukládání dat souvisejících s uživatelskými účty a vytvořenými kurzy. Je realizována jako relační databáze na platformě *MySQL*. Pro její správu byl během vývoje využíván nástroj *phpMyAdmin.* Lze jí rozdělit na dva vzájemně propojené celky, z nichž jeden uchovává uživatelská data a druhý data kurzů a jednotlivých událostí.




*Obrázek 6: Entity Relationship Diagram (ERD) navrhované databáze*

## 5.1 Databáze uživatelů

Část databáze spravující data uživatelských účtů obsahuje celkem pět datových tabulek. Ty jsou podrobně popsány v následujících podkapitolách. Na data z těchto tabulek můžeme souborně nahlížet jako na atributy entity *User* (Uživatel).

### 5.1.1 Základní data účtů

V základní tabulce *users* (uživatelé) se ukládají data jednotlivých uživatelů, pro jednoznačnou identifikaci jednotlivých záznamů byl zvolen přirozený primární klíč v podobě *e-mailu* uživatele, je uložen ve sloupci *useremail* ve formátu varchar a může mít

délku až 256 znaků. Dále se ukládá křestní jméno a příjmení uživatele do sloupců *userfirstname* a *userlastname*, obě opět jako varchar s možná nadhodnocenou maximální délkou 256 znaků.

Atributy v tabulce *users* doplňuje *userdescription* (popis uživatele), ten byl původně zamýšlen pro využití v případě dostupnosti uživatelských profilů, které v ukázkovém projektu zahrnuty nejsou, nicméně do budoucna je tato vlastnost zde připravena jako místo pro vlastní prezentaci, kam by mohli uživatelé vkládat například své zájmy nebo v případě lektorů své reference, zkušenosti či informace o osvědčeních a jiné důkazy odbornosti. Tento prvek je realizován ve formátu text, což umožňuje zadávání neomezeně dlouhých popisků. Další atribut *useractivestate* (boolean) označuje, zdali je účet aktivní a uživatel si jej přeje zachovat. Pokud by si uživatel například přál svůj účet z aplikace vymazat, nastaví se tento příznak na hodnotu *false* a po v systému určené době (např. *14 dní*) je účet vymazán z databáze uživatelů. Tato funkcionalita není součástí ukázkového projektu, je pouze navržena jako možné řešení do budoucna, pro její implementaci by bylo potřeba vyřešit kontrolu konzistence jednotlivých záznamů souvisejících zejména s uživatelem zadanými či navštěvovanými kurzy.

### 5.1.2 Doplňková data uživatelských kurzů

Čtyři další tabulky navázané na záznamy o uživatelích slouží k ukládání doplňkových dat. Jsou jimi *userroles* (uživatelské role), *userimgs* (uživatelské obrázky), *useradresses* (uživatelské adresy) a *calendarsettings* (nastavení kalendáře).

Tabulka *userroles* slouží k uložení uživatelské role, kterou si uživatel zvolí při prvním přihlášení do aplikace. Jejím primárním klíčem je e-mailová adresa uživatele, ta je stejně jako v hlavním záznamu uživatele v tabulce *users* uložena ve sloupci *useremail* a je zároveň primárním a cizím klíčem ve vztahu k hlavnímu uživatelskému záznamu (mezi těmito tabulkami zároveň platí kardinalita one-to-one, díky tomu je možné, aby cizí klíč byl zároveň primární). Jednoznačná identifikace záznamů je řešena stejným způsobem ve všech tabulkách popisovaných v tomto segmentu. Ve sloupci *userrole* je pak uložena v podobě celočíselného záznamu role uživatele v aplikaci. Tento atribut je nastaven tak, aby mohl pojmout až 999 různých uživatelských rolí, nicméně v současné době jsou definované pouze tři. Účty s číslem 0 vystupují jako *studenti*, číslo 1 označuje *lektory*, pro *administrátory* aplikace může být vyhrazeno například číslo 999 (v ukázkovém

modulu aplikace nejsou administrátorské nástroje využívány). Ostatní volné pozice mohou být využity například při nastavení sofistikovanějších uživatelských oprávnění v budoucích verzích aplikace.

Další z tabulek – *userimgs* – slouží k uložení odkazu na profilový obrázek uživatelského účtu, do sloupce *userimg* je možné uložit libovolně dlouhý řetězec označující místo uložení příslušného souboru v místním úložišti či v síti internet. V ukázkovém projektu jsou do této tabulky ukládány odkazy na profilové obrázky účtů *Google*.

Do tabulky *useradresses* (v rámci ukázkového modulu není využívána) může být uložena adresa, která je navázána na *e-mail* uživatele, ke každému uživatelskému účtu je tak možné uložit pouze jednu adresu. Data uložená v této tabulce by mohla být využívána pro zefektivnění procesu objednávky a platby kurzů v případě, že by bylo nutné zadání fakturační adresy.

Poslední ze souboru doplňkových dat jsou uložena v tabulce *calendarsettings*, data v této tabulce by měla být využívána pro nastavení uživatelského kalendáře. Parametry ovlivněnými tímto záznamem mohou být například způsob zobrazení kalendáře, nebo omezení možností přístupu k určitým dnům pro ostatní uživatele – lektoři by mohli změnou tohoto parametru například regulovat, v jaké dny a v jaký čas je možné domluvit si s nimi individuální lekce. Nastavení kalendáře by v této tabulce bylo uložené jako řetězec s pevně definovaným formátem, kde by znaky na určitých pozicích reprezentovaly určitá nastavení. Jelikož součástí ukázkového modulu není nastavení uživatelského profilu, nejsou v něm nastavení kalendáře implementována, v produkční verzi aplikace by nicméně byla téměř nutným prvkem. Alternativou k tomuto databázovému řešení by pak mohlo být využití cookies.

## 5.2 Databáze kurzů

Tento druhý celek v rámci datového úložiště aplikace se skládá ze dvou dílů, z nichž jeden slouží ke správě dat kurzů a druhý jednotlivých událostí v rámci kurzů (jednotlivé lekce). V rámci tohoto menšího celku můžeme rozlišovat dvě různé entity, jednou z nich je *Course* (Kurz) a druhou *Event* (Událost).

### 5.2.1 Parametry kurzů

Tabulka *courses* (kurzy) slouží k ukládání veškerých údajů o kurzech nabízených v rámci aplikace. Aby bylo možné vytvořit více kurzů pod stejným jménem, slouží jako primární klíč zástupné identifikační číslo (*celočíselné ID*) kurzu, které můžeme nalézt ve sloupci *csid,* každý kurz tak má své unikátní číslo, které s vložením nového kurzu stoupá o jednotku. Jméno kurzu, které při jeho vytváření zadává lektor, je uloženo ve sloupci *csname* jako řetězec formátu varchar, jeho maximální délka je 256 znaků. Učitel, který kurz vyučuje, je zapsán ve sloupci *cstutor*, konkrétně je zde uložen e-mail, pod kterým se daný lektor přihlašuje.

Dalším atributem v tabulce *courses* je popis kurzu (*csdesc*), který je textové podoby a může nabývat jakékoli délky, přičemž by z něj mělo být patrné, na jaký kurz se žáci přihlašují. Příznak *cstype* (typ kurzu) pak ovlivňuje povahu kurzu, tj. zdali je kurz skupinový nebo určený pro individuální studium (pouze učitel a žák – soukromé hodiny). Typ kurzu dovoluje v databázi až 999 různých možností (je nadefinován jako celočíselný údaj o délce maximálně 3 znaky), v případě potřeby je tak možné rozšířit typy nabízených kurzů. Přihlédneme-li ale ke skutečnosti, že typ kurzu je jedním z rozhodujících znaků, které ovlivňují celé flow aktivit v aplikaci, je přidání nového druhu kurzů vysoce náročnou záležitostí. V ukázkovém projektu je hodnota *cstype* 1 přidělena individuálním kurzům, zatímco *cstype* 2 kurzům skupinovým.

U hromadných kurzů je nutné ošetřit maximální počet účastníků na jedné lekci, respektive v jednom běhu kurzu, hodnota maximálního počtu studentů je pro tyto účely uložena ve sloupci *csmaxstudents,* musí být celočíselná a je omezená na hodnoty s maximálně třemi číslicemi. Můžeme předpokládat, že toto omezení nebude potřeba překročit, zejména díky omezením většiny on-line komunikačních nástrojů s videohovory, které povolují většinou mnohem menší maximální počet účastníků hovoru. Specifikem skupinových kurzů je v navrhované aplikaci také pevně daný začátek a konec, tyto údaje mohou být v tabulce uloženy ve sloupcích *csstart* (datum začátku) a *csend* (datum konce).

Důležitým parametrem je také příznak *cspublication*, který je definován jako boolean, nabývá hodnot *true* nebo *false* a značí, zdali je kurz již viditelný pro zájemce. Každý nově vytvořený kurz má tuto hodnotu nastavenou jako *false*, lektor může kurz publikovat teprve po zadání potřebných údajů podle typu kurzu.

### 5.2.2 Události

Tabulka *events* (události) slouží pro ukládání jednotlivých událostí vázaných na kurzy (chcete-li lekcí či konzultací). Jejím primárním klíčem je přirozený identifikátor v podobě unikátního id hovoru aplikace *Google Meet*. Pro zajištění propojení s tabulkou kurzů je v této tabulce jako cizí klíč zahrnuto *csid*.

Jméno, které lektor pro danou událost zvolí, je ukládáno do sloupce *evname* a jeho maximální délka je 256 znaků. Pro uložení času schůzky slouží atribut *evtime*, do kterého je možné ukládat hodnoty ve formátu *datetime*, ten odpovídá následujícímu schématu: rrrr-mm-dd hh:mm:ss – například 2021-05-13 10:30:00 odpovídá času 13. května 2021 v 10:30.

V současné verzi ukázkového projektu se počítá s pevnou dobou trvání lekcí o délce jedné hodiny. Pokud by měla být umožněna variabilní délka lekcí, bylo by potřeba tuto skutečnost zanést také do databáze, to by bylo možné například prostřednictvím vložení přímo času konce schůzky ve formátu *datetime*, příslušný sloupec by pak mohl být pojmenován *evend*. Druhou možností zaznamenání doby trvání schůzky by bylo zaznamenat přímo její předpokládanou délku, například v sekundách jako *evlength*, toto řešení by nicméně bylo méně praktické z hlediska uživatelských vstupů.

Dalším nedostatkem současné verze databáze je, že neumožňuje využít jedno id schůzky *Google Meet* pro více zaznamenaných lekcí. Aby se nám tato možnost zpřístupnila a zároveň bylo toto id stále možné využívat jako primární klíč, bylo by možné zavést do tabulky *events* nový sloupec, který by pevně daným způsobem upravoval periodické opakování schůzky v čase, takto by byly ošetřeny pravidelně se opakující události. Pro nepravidelně se vyskytující události by tento typ řešení byl ale poněkud komplikovaný.

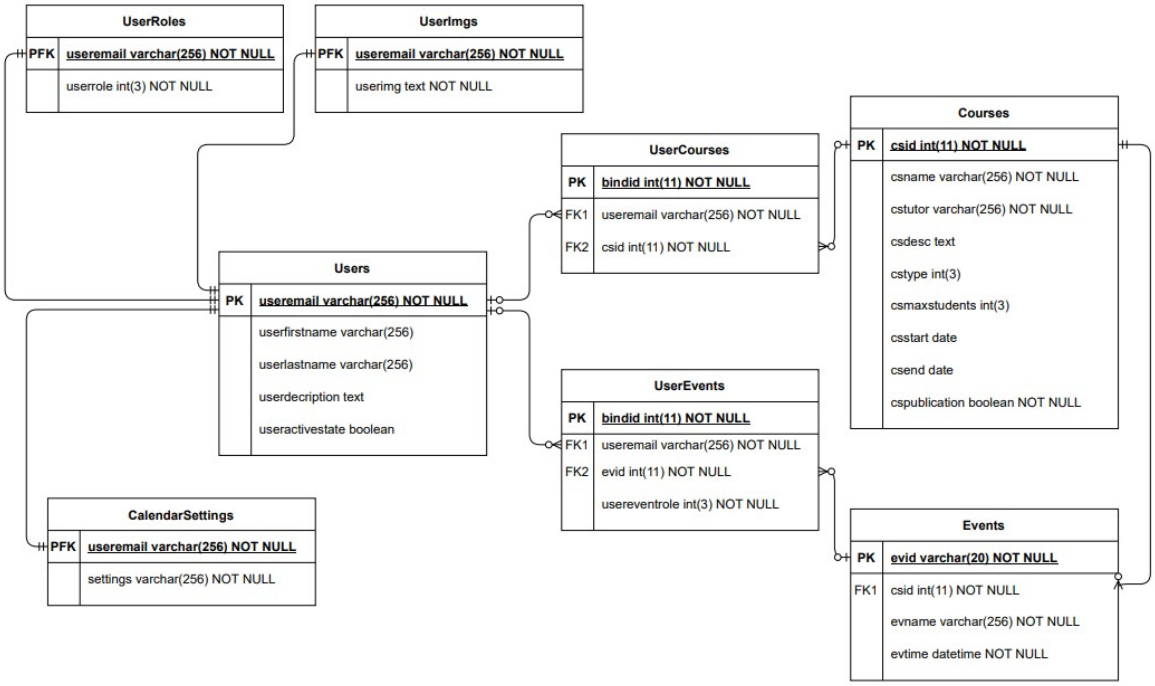
## 5.3 Propojení uživatelů s kurzy

Dohromady se data kurzů a uživatelů potkávají prostřednictvím dvou tabulek (mimo tyto dvě tabulky stojí propojení kurzů s jejich lektory popsané v kapitole 5.2.1) *usercourses* a *userevents*.

Tabulka *usercourses* slouží k evidenci výhradně studentů zapsaných na skupinové kurzy, zaznamenávání vyučujících do ní nemá smysl, jelikož by šlo o duplicitní záznam, jsou již zaznamenáni v tabulce *courses* jako *cstutor*. Při přihlášení uživatele na skupinový kurz

je do této tabulky uložen jeho přihlašovací e-mail do sloupce *useremail* společně s *csid* příslušného kurzu, obě tyto hodnoty tak fungují jako cizí klíče. Jako primární klíč je u každého propojení přiděleno tzv. *bindid* (ID propojení), které je vyšší o jednotku pro každý nový záznam v tabulce.

Pro asociaci událostí a jejich účastníků slouží tabulka *userevents*. Ta stejně jako předchozí tabulka využívá *bindid* jako zástupného primárního klíče. Při vytvoření schůzky je v této tabulce vytvořen záznam pro každého jejího účastníka, přičemž se zaznamenají dva cizí klíče v podobě *useremail* ze záznamu uživatele (tabulka *users*) a *evid* příslušné konference, vyjma těchto dvou údajů je také u každého uživatele zaznamenána role, kterou v rámci setkání zastává, vyučující jsou zaznamenáni s hodnotou *usereventrole* 1, studenti s hodnotou 0. Toto doplňkové určení role umožňuje, aby se na cizí kurzy mohli přihlašovat i ostatní lektoři.




*Obrázek 7: ERD upravené do podoby odrážející strukturu tabulek v databázi*

*ERD diagramy* v plné velikosti jsou součástí *přílohy A*.

# 6 Vývoj aplikace

Tato kapitola se podrobněji věnuje prostředkům a postupům použitým během vývoje ukázkového modulu aplikace. Můžeme předpokládat, že podobné postupy by mohly být uplatněny i při vývoji produkčních verzí aplikace. Jako pracovní název pro vyvíjenou aplikaci byl zvolen anglický výraz *Haste* [heɪst], který po překladu do češtiny znamená *spěch.* Kompletní soubor zdrojů aplikace je možné nalézt v *příloze B.* Tato kapitola obsahuje ukázky zdrojového kódu vyvíjené aplikace. Aby ukázky kódu zabíraly méně prostoru v rámci textu práce, byly z některých z nich odstraněny zakomentované části kódu.

Většina aplikace je realizována v jednom souboru (*index.php*), zobrazení různých nabídek je umožněné skrze využití uživatelských rolí (uložených v *$_SESSION*) a superglobálních proměnných (*$_REQUEST, $_GET ...*).

## 6.1 Integrace Google API

Jelikož je naše aplikace silně navázána na nástroj *Google Meet*, je nutné využívat pro její fungování rozhraní *Google API*. Při vytváření projektu je nutné ve vývojářské konzoli Google (*Google Cloud Platform* – console.cloud.google.com) vytvořit klient, který nám umožní přistupovat k potřebným zdrojům. Údaje v tomto klientu umožňují autentizaci uživatele pomocí protokolu Oauth 2.0.

Při vytváření autentizačního klienta Google pro aplikaci je třeba nejprve ve vývojářské konzoli vytvořit příslušný projekt. Dalším krokem je definice parametrů obrazovky vyžadující svolení uživatele pro přístup k citlivým datům v rámci jeho účtu Google. Součástí těchto parametrů jsou jméno aplikace, e-mailový kontakt na uživatelskou podporu a e-mailová adresa vývojáře. Pro účely testování je nutné vložit do příslušné nabídky také adresy účtů testovacích uživatelů.

Dále je potřeba určit soubory oprávnění, ke kterým vyvíjená aplikace bude požadovat přístup. Tyto parametry jsou předpřipravené pro každou API knihovnu, kterou projekt využívá. Pokud během vytváření projektu vyvstane potřeba využití nové knihovny, je nutné ji v developerské konzoli povolit a vymezit povolení, která bude potřebné od uživatele získat. Jednotlivé soubory oprávnění dělí Google na veřejné (*non-sensitive*),

důvěrné (*sensitive*) a s omezeným přístupem (*restricted*) – podle typu uživatelských dat, ke kterým umožňují přístup nebo podle typu aktivit, které jejich prostřednictvím uživatel aplikaci povoluje. Pokud aplikace vyžaduje oprávnění ze skupin *sensitive* nebo *restricted*, je nutný uživatelský souhlas s jejich udělením.

Po nastavení obrazovky souhlasu s udělením oprávnění je nutné vygenerovat samotný webový klient a údaje nutné pro přístup k němu, v tomto kroku je nutné definovat povolené *URI* pro přístup z prohlížeče a povolená *URI* pro přesměrování po dokončení autentizace uživatele. Jakmile jsou tato nastavení dokončena, je možné vygenerovat unikátní client ID, které slouží pro identifikaci našeho Oauth klientu a přístup k němu. Je možné také stáhnout soubor ve formátu *json* s údaji námi vytvořeného klienta, který můžeme později využít pro jeho inicializaci.


```
{"web":{
    "client_id":"338389154925-d2e3j3mbsvncfa6j6svtdjfujtr80cat.apps.googleusercontent.com",
    "project_id":"diplomka-app",
    "auth_uri":"https://accounts.google.com/o/oauth2/auth",
    "token_uri":"https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url":"https://www.googleapis.com/oauth2/v1/certs",
    "client_secret":"sGsYWCEtLv4pRZpSL2aI25bW",
    "redirect_uris":["https://localhost/Diplomka/index.php",
        "https://localhost/Diplomka/mail.php",
        "http://localhost/Diplomka/index.php",
        "http://localhost/Diplomka/mail.php"],
    "javascript_origins":["https://localhost"]
    }
}
```


*Zdrojový kód 1: Údaje klientu vyvíjené aplikace*

### 6.1.1 Využívané API

Ukázková aplikace využívá celkem tři *API* nástrojů Google, těmi jsou *Google Calendar API*, *Gmail API* a *People API*. Přístup ke kalendáři Google uživatele je využíván pro vytváření lekcí a s nimi spojených schůzek na Google Meet. Aplikace pro tyto účely vyžaduje oprávnění k zobrazení, sdílení a mazání událostí a kalendářů, ke kterým má uživatel skrze Google Kalendář přístup.

*Gmail API* slouží v ukázkové aplikaci jako prostředek zasílání automatizovaných e-mailů při komunikaci studenta s lektorem, pro tento účel je vyžaduováno povolení ke správě konceptů zpráv a zasílání e-mailů z účtu uživatele.

Prostřednictvím *People API* získáváme doplňující informace o uživatelském účtu (díky využití tohoto rozhraní nám OAuth klient navrací širší informace v rámci identifikačního tokenu uživatele), například odkaz na profilový obrázek uživatele. Tímto API přistupujeme pouze k veřejným datům uživatele, která sám zpřístupnil ve svém Google účtu, pro získání těchto dat není nutné vyžadovat svolení uživatele.

*Tabulka 4: Oprávnění vyvíjené aplikace pro interakci s Google API*

| **API** | **Označení souboru oprávnění** | **Umožněné činnosti** | **Druh oprávnění** |
|---|---|---|---|
| *(Nevyžaduje speciální API)* | openid | Ztotožnění uživatele s osobními informacemi na Googlu | *veřejné* |
| *People API* | .../auth/userinfo.email | Zobrazení primárního Google účtu a e-mailové adresy | *veřejné* |
| *People API* | .../auth/userinfo.profile | Zobrazení osobních informací zveřejněných uživatelem | *veřejné* |
| *Google Calendar API* | .../auth/calendar | Zobrazení, úprava, sdílení a mazání všech uživatelem přístupných kalendářů | *důvěrné* |
| *Gmail API* | .../auth/gmail.compose | Správa konceptů a zasílání e-mailů z účtu uživatele | *s omezeným přístupem* |

### 6.1.2 ID Token

Pro účely identifikace autentizovaného uživatele je možné při použití Google klientu využít tzv. *identifikační (ID) token*. Tento soubor dat získáme pomocí příslušných metod po inicializaci klienta v samotném běhu programu, jakmile dojde k autentizaci uživatele. Obsahuje informace o přihlášeném účtu Google, jejich rozsah je ovlivněn povolenými oprávněními aplikace.

V rámci povolených oprávnění vyvíjené aplikace (viz Tabulka 4) je možné prostřednictvím id tokenu zobrazit například e-mailovou adresu přihlášeného účtu, jméno a příjmení jeho vlastníka nebo odkaz na profilovou fotku jeho účtu. V současné verzi aplikace (viz *příloha B*) je z těchto dat využívána zejména e-mailová adresa uživatele.

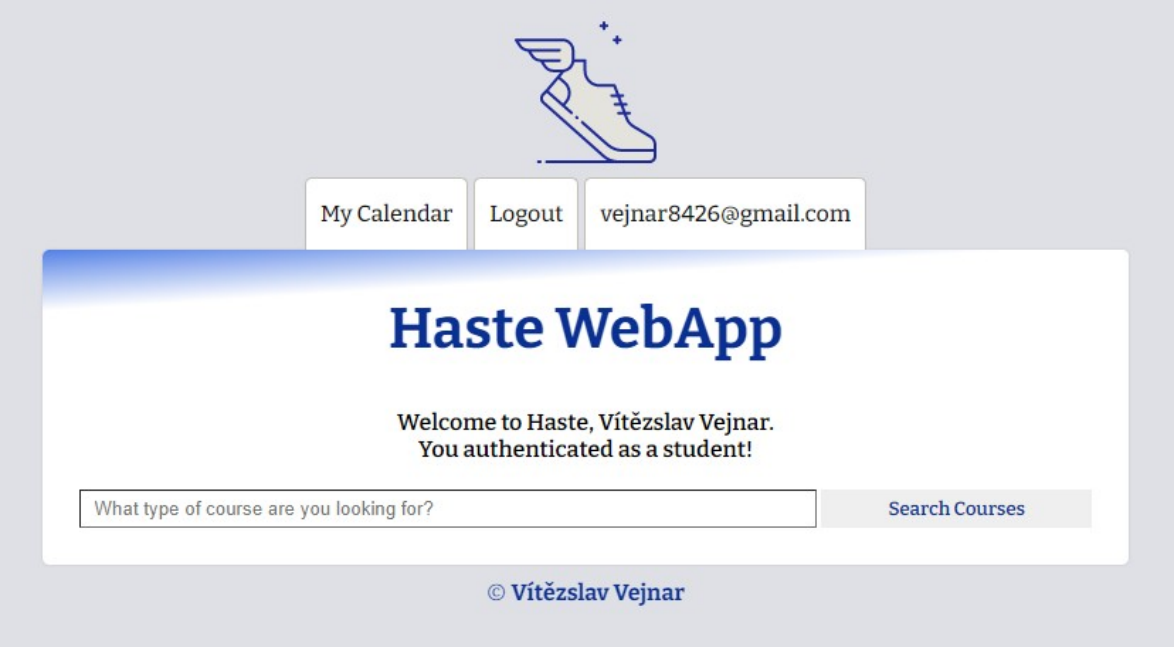
Důležitým specifikem *ID tokenu* je také jeho omezená platnost, ta je omezena na dobu jedné hodiny od autentizace uživatele. Po uplynutí této doby již přihlášení ke službám Google není funkční. Aby se tak uživatel nemusel po uplynutí jedné hodiny znovu

přihlašovat, je nutné zajistit, aby se platnost tokenu obnovila. To je možné provést prostřednictvím *obnovovacího (refresh) tokenu*, který je k dispozici prostřednictvím klienta aplikace.

## 6.2 Front-End

Struktura uživatelského rozhraní aplikace Haste je realizována za využití značkovacího jazyka *HTML* a je kompletně generována pomocí příkazů v jazyce *PHP* – na straně serveru – tedy prostřednictvím back-endu (několik příkladů generování podoby *UI* lze nalézt v kapitole 6.3). Grafická podoba projektu byla upravena prostřednictvím kaskádových stylů (*CSS*).

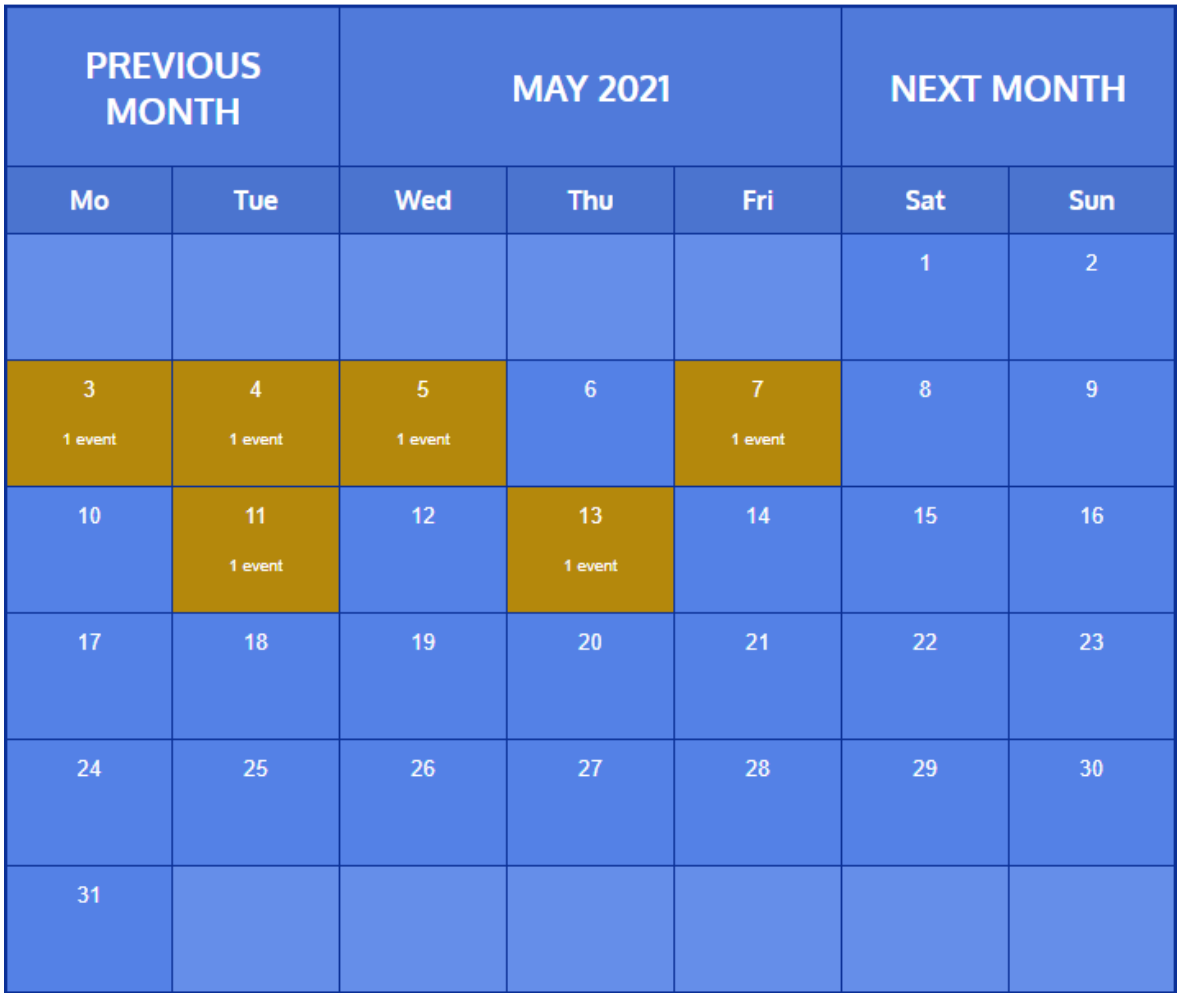
Barevně je webové uživatelské rozhraní laděné do odstínů modré se zlatými akcenty. Pro zvýšení přehlednosti jsou segmenty stránek stylované do dvou střídajících se barevných schémat (prostřednictvím tříd v *CSS* předpisech) – tmavého (využívaného pro přehledy kurzů) a světlého.




*Obrázek 8: Hlavní stránka aplikace Haste (přihlášení v roli studenta)*

V samotném uživatelském rozhraní (ve front-endu) nedochází v ukázkové aplikaci k žádné manipulaci s daty. Do budoucna, zejména při vytváření produkčních verzí by bylo dobré některé z funkcí do prezentační vrstvy přemístit. Tento krok by mohl vést k větší plynulosti užívání aplikace a lepšímu designu *UX* a *UI*. Otevřely by se nové možnosti, které

by umožnily, aby web a jeho vzhled více reagovaly na pokyny uživatele, například aby formuláře reagovaly na údaje zadávané uživatelem. Přesunutím některých funkcí na hlavní stránce nebo například v uživatelském kalendáři z aplikační vrstvy do prezentační bychom odstranili nutnost aktualizovat stránku při každé akci provedené uživatelem.



*Obrázek 9: Rozhraní kalendáře aplikace Haste*

Pro přepracování části projektu do front-endu a zlepšení designu *UI* se nabází využití některého z frameworků *JavaScriptu* – například *React* či *Vue.js*, anebo jazyka *TypeScript* – rozhraní *Angular*.

## 6.3 Back-End

Pro back-endový vývoj aplikace byl využit programovací jazyk *PHP*, byly zahrnuty také prostředky objektově orientovaného programování. Příklady jednotlivých aktivit s příslušnými fragmenty kódu jsou podrobně rozebrané v následujících podkapitolách.

### 6.3.1 Zprovoznění knihoven Google API (nástroj Composer)

Abychom mohli využívat nástroje z API knihoven Googlu, je kromě vytvoření přístupových údajů k autentizačnímu klientu aplikace nutné také nainstalovat knihovnu funkcí (chcete-li metod) *Google API* v příslušném programovacím jazyce přímo do zdrojového adresáře projektu, toto umožní její využívání v rámci aplikace.

Nejjednodušším způsobem integrace zmíněné knihovny pro jazyk *PHP* je instalace za použití nástroje *Composer*, který je volně ke stažení ze stránek getcomposer.org. Jakmile je dokončena instalace tohoto nástroje, je možné stáhnout a nainstalovat Google APIs Client Library pro PHP přímo prostřednictvím příkazového řádku spuštěním příkazu `composer require google/apiclient:"^2.7` v kořenové složce projektu. Do souborů projektu je pak nutné přidat příkaz pro vyžádání autoloaderu (viz Zdrojový kód 2), který zabezpečuje možnost využití všech metod nainstalovaných v rámci Google API bez nutnosti zahrnutí všech využívaných souborů se zdrojovými kódy do každé stránky zvlášť. Každou třídu ze zmíněné knihovny vyhledává automaticky na základě jejího jména.

```
require_once 'vendor/autoload.php';
```

*Zdrojový kód 2: Příkaz vyžádání autoloaderu*

Zmíněný autoloader je možné využít i pro přístup k uživatelem vytvořeným třídám, je potřeba tyto třídy pouze nadefinovat v souboru *composer.json* v kořenovém adresáři projektu. V tomto souboru je nutné určit namespace (jmenné označení souboru tříd v objektově orientovaném *PHP*), pod kterým se bude k uživatelem definovaným třídám přistupovat a umístění zdrojových souborů tříd relativně ke konfigurovenému souboru.

```
{
    "require": {
        "google/apiclient": "^2.7"
    },
    "autoload": {
        "psr-4": {
          "Haste\\":"classes/"
        }
    }
}
```

*Zdrojový kód 3: Soubor composer.json upravený pro využití uživatelem definovaných tříd*

### 6.3.2 Implementace architektury MVC

*Model-View-Controller (MVC)* architektura dělí aplikaci na tři hlavní komponenty, které nazýváme *Model*, *View* a *Controller*. Každá z částí tohoto schématu zabezpečuje jinou funkci. Jedná se o široce používanou architekturu v rámci webového vývoje.

*Model* zabezpečuje veškeré interakce s databází, je jedinou částí architektury *MVC*, která přímo ovlivňuje uložená data, skrze jeho metody můžeme data uložená v databázi zakládat, měnit či mazat, ale také například pouze vybírat. Jednotlivé komponenty modelu realizujeme jako třídy *OOP*, můžeme tak uplatnit dědičnost mezi *Modelem* a dalšími komponenty (*View* a *Controller* dědí od *Modelu*). K tomuto komponentu (potažmo *třídě*) by aplikace neměla přistupovat přímo, nýbrž skrze komponenty *View* a *Controller*.

*View* představuje soubor všech funkcí sloužících ke zobrazování dat z databáze na *UI*, k funkcím tohoto komponentu může aplikace přistupovat přímo, *View* pak za pomoci údajů získaných z aplikace volá metody *Modelu* a vrací ke zobrazení získaná data.

*Controller* slouží ke změnám dat v databázi na základě zadání z aplikace. Jeho prostřednictvím jsou volány metody *Modelu* sloužící například k mazání nebo aktualizaci dat v databázi (MVC Components, 2021).

V aplikaci *Haste* můžeme využití architektury *MVC* pozorovat při práci s daty v databázi (viz kapitola 5). V rámci autorem definovaných tříd jsou pro každý ze dvou větších celků databáze vyhrazeny tři třídy. Pro přistupování k datům uživatelů slouží třída *User*, data

kurzů a jednotlivých událostí je možné nahlížet a měnit skrze třídu *Course*. K oboum třídám sloužícím jako *Model* jsou doplněny třídy sloužící jako *View* (*UserView* a *CoursevView*) a *Controller* (*UserContr* a *CourseContr*).

Třídy *User* a *Course* dědí metody třídy *Dbh,* která slouží pro připojení k databázi projektu pomocí objektu třídy *PDO*.

```
<?php

namespace Haste;

class Dbh {
    private $hostName = "localhost";
    private $userName = "root";
    private $pwd = "";
    private $dbName = "hastewebapp";

    protected function connect() {
        $dsn = 'mysql:host='.$this->hostName.';dbname='.$this->dbName;
        $pdo = new \PDO($dsn, $this->userName, $this->pwd);
        $pdo->setAttribute(\PDO::ATTR_DEFAULT_FETCH_MODE, \PDO::FETCH_ASSOC);
        return $pdo;
    }
}
```

*Zdrojový kód 4: Připojení k aplikace databáze pomocí třídy Dbh*

Třídy sloužící jako *Model* obsahují soubor metod sloužících pro výběr a manipulaci dat z databáze (třída *User* pro tabulky entity *User* a propojovací tabulky, třída *Course* pro tabulky entit *Course* a *Event*). Každá z těchto metod zpravidla obsahuje definici šablony SQL dotazu, ke které jsou teprve při běhu programu přiřazeny hodnoty vstupních parametrů metody, čímž je dotaz připraven pro následné provedení.

```
class User extends Dbh{

    protected function getUser($email){
        $sql ="SELECT * FROM users WHERE useremail = ?";
        $stmt = $this->connect()->prepare($sql);
        $stmt->execute([$email]);

        $results=$stmt->fetchAll();
        return $results;
    }
```

*Zdrojový kód 5: Definice třídy User s příkladem metody pro získání dat uživatele z databáze*

Třídy *View* obsahují metody, které předávají do běhu programu hodnoty získané z databáze prostřednictvím SQL dotazů SELECT. Metody *View* tříd volají metody děděné ze svých *modelových* tříd, zpravidla pojmenované *get/jméno sloupce v tabulce/.*

```
protected function getUserRole($email){
        $sql ="SELECT * FROM userroles WHERE useremail = ?";
        $stmt = $this->connect()->prepare($sql);
        $stmt->execute([$email]);

        $results=$stmt->fetchAll();
        return $results;
}
```

*Zdrojový kód 6: Metoda getUserRole*

Samotné metody ve *View* třídách mají jména začínající na *show…*, můžeme tak využívat například metodu *showUserRole*, která vrátí záznam z tabulky *userroles*, ten obsahuje roli vyhledávaného uživatele. Záznamy z databázových tabulek jsou ve vyvíjené aplikaci navraceny jako asociativní pole – soubor dat, kde jsou jednotlivé záznamy indexovány jmény sloupců z příslušné prohledávané databázové tabulky.

```
public function showUserRole($email){
        $results = $this->getUserRole($email);

        return $results;
    }
```

*Zdrojový kód 7: Příklad metody pro předání dat z databáze do běhu aplikace*

V případě zmíněné funkce *showUserRole* (viz výše) dojde k zavolání metody *getUserRole* (viz Zdrojový kód 6). Tato metoda získá data z příslušného záznamu v tabulce *userroles* podle e-mailu předaného jako parametr volané metody a předá je nazpět do proměnné *$results*, která je navrácena jako výsledek volání metody *showUserRole*. Výsledná návratová hodnota tak má podobu dvojrozměrného pole, jelikož už metoda *getUserRole* vrátila zpět dané asociativní pole s indexy v podobě jmen jednotlivých sloupců databázové tabulky. Metoda *showUserRole* pak každému záznamu zjištěnému metodou vybírající data z databáze přiděluje číselný index. Tuto skutečnost pak musíme mít na paměti, když přistupujeme ke zjištěním datům. Pokud chceme tedy například zobrazit roli uživatele zjištěnou metodou *showUserRole*, je nutné k ní přistupovat jako k *$role[0]['userrole']*.

Třídy sloužící jako *Controller* obsahují metody, které umožňují manipulovat daty v databázi a zakládat nová data přímo z běhu programu, pro každou tabulku v databázi jsou vytvořeny metody pro založení nových záznamů, dále jsou dle potřeby připraveny také metody pro aktualizaci a mazání záznamů. Ke každé z metod *Controlleru* lze napárovat korespondující metoda příslušného *Modelu*. Většina funkcí spadajících pod třídy komponentů pro manipulaci s databází pouze volá související metodu z modelu, které předává parametry získané za běhu programu, načež je proveden databázový dotaz. Například metoda *createUser* (viz níže) ze třídy *UserContr* se vymyká, jelikož volá hned několik metod zakládajících nové záznamy v databázových tabulkách.

```
public function createUser($email, $firstname, $lastname, $role, $img){
        $this->setUser($email, $firstname, $lastname);
        $this->setUserRole($email, $role);
        $this->setUserImg($email, $img);
        $this->setCalendarSettings($email, "default");
    }
```

*Zdrojový kód 8: Metoda createUser - vytvoření záznamu uživatele*

Po zavolání zmíněné metody se vytvoří nový záznam v tabulkách *users*, *userroles*, *userimgs* a *calendarsettings*, čímž dojde k inicializaci uživatelského účtu v rámci aplikace *Haste*.

Všechny metody tříd, které zastupují *Model*, jsou definované jako *protected*, aby bylo zamezeno k nim přistupovat vně těchto a jejich dědičných tříd. Většina metod tříd sloužících jako komponenty *View* a *Controller* je vytvořena jako *public*, aby k nim bylo možné přistupovat odkudkoliv z běhu programu, výjimku tvoří některé metody sloužící k seřazení nalezených záznamů z databáze, které jsou v rámci *View* tříd definovány jako *private,* ty mohou být využity pouze v rámci dané třídy.

### 6.3.3 Autentizace uživatele

Funkce sloužící k autentizaci uživatele a generaci nutných údajů pro zprovoznění *OAuth2.0* klienta nalezneme v souboru *oauthhandler.inc.php* (*příloha B* – adresář *Haste/includes/*).

Klient, pro který máme připravené příslušné identifikační údaje (viz kapitola 6.1), musíme pro využití v aplikaci inicializovat za využití těchto údajů získaných jako *oauth credentials* ve vývojářské konzoli Google ve formě souboru *json* (viz Zdrojový kód 1). K inicializaci klienta ve vyvíjené aplikaci dochází prostřednictvím funkce *createClient* (viz Zdrojový kód 9).

```
function createClient($redirect_uri, $oauth_credentials){
    $scopes = array('https://www.googleapis.com/auth/userinfo.email',
    'https://www.googleapis.com/auth/userinfo.profile',
    'https://www.googleapis.com/auth/gmail.compose',
    Google_Service_Calendar::CALENDAR);

    $client = new Google\Client();
    $client->setAuthConfig($oauth_credentials);
    $client->setRedirectUri($redirect_uri);
    $client->setScopes($scopes);
    $client->setAccessType('offline');
    $client->setApprovalPrompt('force');
    $guzzleClient = new GuzzleHttp\Client(array('curl' =>
        array(CURLOPT_SSL_VERIFYPEER => false)));
    $client->setHttpClient($guzzleClient);

    return $client;
}
```

*Zdrojový kód 9: Inicializace OAuth klientu Google*

Jako parametry jsou této funkci předány údaje z výše zmíněného *json* souboru (*$oauth_credentials*) a také adresa, kam má být uživatel přesměrován po dokončení autentizace (*$redirect_uri*, v rámci aplikace Haste se jedná vždy o tu stránku, odkud vstupoval do procesu autentizace).

Proměnná *$scopes* obsahuje výčet všech potřebných oprávnění, která musí být klientu přisouzena, aby mohly být využívány veškeré funkce vyvíjené aplikace. Samotný klient je vytvořen jako objekt *Google\Client* a uložen do proměnné *$client*. Další příkazy slouží ke konfiguraci klienta pro použití v dané aplikaci. Metodou *setAuthConfig* je nastaven za pomoci údajů vytvořených v developerské konzoli. Prostřednictvím *setRedirectUri* je zaznamenána daná adresa přesměrování. Za pomoci *setScopes* jsou určena oprávnění, která bude klient po uživateli vyžadovat.

*Metoda setAccessType* s možností *offline* umožňuje aplikaci provést obnovení platnosti *ID tokenu* v případě, kdy uživatel není přítomen u prohlížeče. *setApprovalPrompt* s nastavením *force* zajišťuje, aby uživatel musel potvrdit při přihlášení oprávnění aplikace.

Proměnná *$guzzleClient* společně s funkcí *setHttpClient* slouží pouze pro zamezení chybám při volání *Google Calendar API* na lokálním serveru, mimo lokální server tyto dva příkazy není nutné využívat.

Jakmile je nastavení klienta hotové, je navrácen pro využití v dalším běhu aplikace. Využívají jej například funkce *loginHandle*, která zabezpečuje proces přihlášení a odhlášení a funkce *tokenHandle*, která zajišťuje získání *ID tokenu* a zajištění jeho platnosti.

```
function loginHandle($redirect_uri, $client){
    if (isset($_REQUEST['logout'])) {
        unset($_SESSION['id_token_token']);
        unset($_SESSION['userrole']);
    }
    if (isset($_GET['code'])) {
        $token = $client->fetchAccessTokenWithAuthCode($_GET['code']);
        $_SESSION['id_token_token'] = $token;

        header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
        return;
    }
}
```

*Zdrojový kód 10: Funkce loginHandle*

V průběhu funkce *loginHandle* je vyhodnocován stav procesu přihlášení uživatele, tj. jestli není potřeba uživatele buď odhlásit, anebo naopak přihlásit. V první fázi funkce testuje, zdali je nastavena *PHP* žádost (*PHP* request) s názvem *logout*, která se vyskytuje v případě, kdy někdo klikl na odhlašovací odkaz. V případě existence této *PHP* žádosti jsou pak vymazány hodnoty z proměnných v superglobální proměnné *SESSION* s indexy *id_token_token* (místo, kam se ukládá *ID token*) a *userrole* (zde bývá uložena role, ve které vystupuje přihlášený uživatel).

Pokud je přítomna superglobální proměnná typu *GET* s názvem *code* (ta je přítomna pokud se uživatel zrovna navrátil z procesu autentizace), dojde k získání přístupového tokenu pomocí kódu v ní uloženém, tento token je pak uložen do *SESSION* s indexem *id_token_token* a uživatel je přemístěn na adresu specifikovanou v proměnné *$redirect_uri*.

```
function tokenHandle($client){
    if (
        !empty($_SESSION['id_token_token'])
        && isset($_SESSION['id_token_token']['id_token'])
      ) {
        $date = date_create();
        $timestamp = date_timestamp_get($date);
        if ((($_SESSION['id_token_token']['created']+REFRESH_VALIDITY) - $timestamp) < ZERO){
          unset($_SESSION['id_token_token']);
          $authUrl = $client->createAuthUrl();
          return $authUrl;
        } else if((($_SESSION['id_token_token']['created']+
                $_SESSION['id_token_token']['expires_in']) - $timestamp) < REFRESH_LIMIT){
            $token = $client->refreshToken($_SESSION['id_token_token']['refresh_token']);
            $_SESSION['id_token_token'] = $token;
            $client->setAccessToken($_SESSION['id_token_token']);
            $logoutUrl = 'http://'.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF'].'?logout';
            return $logoutUrl;
        } else {
            $client->setAccessToken($_SESSION['id_token_token']);
            $logoutUrl = 'http://'.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF'].'?logout';
            return $logoutUrl;
        }
      } else {
        $authUrl = $client->createAuthUrl();
        return $authUrl;
      }
}
```

*Zdrojový kód 11: Funkce tokenHandle*

Ve funkci *tokenHandle* dochází ke zpracování ID tokenu získaného při přihlášení účtem Google. Pokud je v rámci superglobály *SESSION* token uložený, dojde k otestování jeho platnosti. V případě, že od času vytvoření tokenu uběhlo více, nežli jeden den (hodnota konstanty REFRESH_VALIDITY), je hodnota přihlášeného tokenu ze SESSION vymazána a je vytvořen odkaz pro atentizaci uživatele, který je možné využít jako odkaz na přihlášení (*refresh token* potřebný k obnovení platnosti přihlášení je totiž v tuto chvíli už neplatný).

Pokud je obnovovací token stále platný a uběhlo více nežli 60 minut od vygenerování současného ID tokenu, je tento token obnoven za použití obnovovacího tokenu a nastaven jako nový přístupový token pro přístup ke službám Google. Je vytvořen odkaz pro odhlášení uživatele.

Když nedojde k zjištění vypršení platnosti žádného z tokenů, je pouze nastaven současný token jako přístupový token a dojde k vytvoření odhlašovacího linku.

Pokud není token v SESSION uložen, není přihlášen žádný uživatel, je tak pouze vytvořen a navrácen odkaz pro autentizaci uživatele.

```
function oAuthHandler($redirect_uri, $oauth_credentials){
    $client = createClient($redirect_uri, $oauth_credentials);
    loginHandle($redirect_uri, $client);
    $url = tokenHandle($client);
    $token_data = tokenDataExtractor($client);

    $data = array(
        "url" => $url,
        "token_data" => $token_data
    );

    return $data;
}
```

*Zdrojový kód 12: Funkce oAuthHandler*

Funkce *oAuthHandler* je volána na každé stránce aplikace *Haste*, představuje souhrné řešení autentizace uživatele pro použití aplikace a zahrnuje v sobě funkce popsané výše spolu s funkcí *tokenDataExtractor*, která zajistí data dostupná skrze *ID token* a uloží je do asociativního pole. Jakmile je proces ověření přihlášení dokončen, vrací tato funkce přihlašovací nebo odhlašovací odkaz spolu s daty zjištěnými z *ID tokenu*.

### 6.3.4 Založení záznamu uživatele

Při prvním přihlášení (po dokončení autentizace uživatele skrze klienta *Google*) je nutné přisoudit uživateli roli pro užívání aplikace, tu si uživatel vybere sám skrze UI, na výběr má v ukázkovém modulu aplikace ze dvou rolí (viz kapitola 4.2.1).


*Obrázek 10: Formulář výběru role uživatele aplikace Haste*

Jakmile si uživatel roli vybere, jsou data z jeho *ID tokenu*, která byla navrácena jako výsledek běhu funkce *oAuthHandler* (viz kapitola 6.3.3), uložena do databáze. Tato data jsou uložena v proměnné *$oauth_return['token_data']*. Konkrétně jsou do databáze uloženy uživatelovo jméno a příjmení a odkaz na profilový obrázek jeho Google účtu, dále je uložena také jím zvolená role.

```
if(isset($_SESSION['id_token_token'])){
        if(isset($_GET['signup'])){
            //Vytvoření nového uživatelského záznamu, určitě by šlo trochu uklidit
            $role = $_GET['signup'];
            $usercontr = new Haste\UserContr();
            if($role==0 || $role==1){
                $usercontr->createUser($oauth_return['token_data']['email'],
                        $oauth_return['token_data']['given_name'],
                        $oauth_return['token_data']['family_name'],
                        $role, $oauth_return['token_data']['picture']);
                header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
                return;
            } else {
                echo "There was an issue during your sign-up process, try again.";
            }
        }
```

*Zdrojový kód 13: Založení záznamu uživatele v databázi*

Zvolená role uživatele je předávána aplikaci jako superglobální proměnná typu *GET* s názvem *signup*, ve které je uloženo příslušné číslo role (0 = *student*; 1 = *lektor*). Vytvoření záznamů v databázi probíhá skrze metodu *createUser* (viz Zdrojový kód 8) třídy *UserContr* (controller část databáze spravující uživatelská data), ta kromě výše zmíněných hodnot z *ID tokenu* založí v databázi také záznam uživatelských nastavení kalendáře nastavený na základní hodnotu *default*, který je připravený pro implementaci uživatelských nastavení dostupnosti kalendáře v budoucích verzích aplikace (viz kapitola 5.1.2).

### 6.3.5 Kontrola role přihlášeného

Tento proces probíhá při každé aktualizaci stránky, je-li přihlášen uživatel, tj. je-li nastavena hodnota superglobální proměnné *SESSION* na indexu *id_token_token* (autentizace uživatele viz kapitola 6.3.3). Role přihlášeného je zkontrolována pomocí porovnání databázového záznamu z tabulky *userroles* s předdefinovanými čísly jednotlivých rolí (viz kapitola 5.1.1). Získání dat o uživatelské roli probíhá pomocí metody *showUserRole* třídy *UserView* (view části databáze, která je určena pro správu uživatelských dat).

```
$db_user_role = $userview->showUserRole($oauth_return['token_data']['email']);
if(!empty($db_user_role)){
    $_SESSION['userrole']=$db_user_role[0]['userrole'];
    if($db_user_role[0]['userrole']==0){
        echo '<p class="welcome-text">You authenticated as a student!</p>';
    } else if ($db_user_role[0]['userrole']==1){
        echo '<p class="welcome-text">You authenticated as a tutor!</p>';
    } else if ($db_user_role[0]['userrole']==999){
        echo '<p class="welcome-text">You are an admin!</p>';
    }
    echo "<br>";
}
```

*Zdrojový kód 14: Kontrola role přihlášeného*

Jakmile je nalezena shoda mezi číslem role z databáze (uloženém v proměnné *$db_user_role*) a jedním z čísel určených rolí, je do *UI* vypsána hláška o autentizaci uživatele v dané roli, tj. zdali se přihlásil jako *lektor* či *student* (ukázkový modul aplikace nevyužívá roli *administrátora*).

Role zjištěná v tomto kroku je uložena do superglobální proměnné *$_SESSION[‘userrole’]*, její hodnota pak ovlivňuje, které aktivity jsou uživateli v dalším užívání aplikace zpřístupněny, všichni uživatelé mají k dispozici zobrazení svého plánovacího kalendáře a vyhledávání v rámci nabízených kurzů. Aktivity zakládání a modifikace kurzů jsou zpřístupněné pouze *lektorům*.

### 6.3.6 Vytváření a úprava kurzů

Některé ze segmentů hlavní stránky jsou zpřístupněny pouze uživatelům s rolí *lektora*. V ukázkové aplikaci se jedná o oddíly založení nového kurzu (*Course setup*) a přehledu kurzů pro správu (*Course dashboard*), ze kterého je možné přejít na detail kurzu pro správu.

```
if(isset($_POST['basicdata'])){
    $redirect = $_POST['url'];
    $name = $_POST['name'];
    $tutor = $_POST['tutor'];
    $desc = $_POST['desc'];
    $coursecontr->createCourse($name, $tutor, $desc, 0);
    header ('Location: '.filter_var($redirect, FILTER_SANITIZE_URL).'?coursesetup=success');
    exit();
}
```

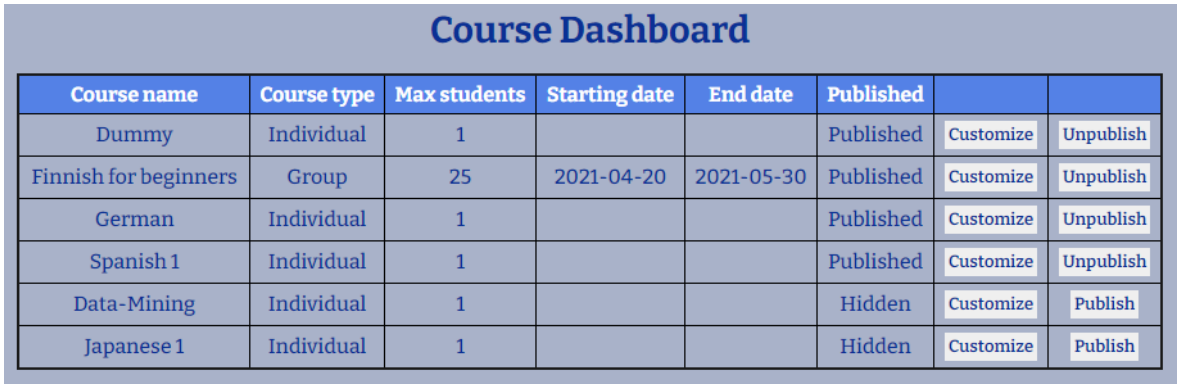
*Zdrojový kód 15:* Skript pro *založení záznamu nového kurzu do databáze*

Při založení kurzu uživatel vkládá název a popisný text nového kurzu do formuláře, který předává prostřednictvím superglobálních proměnných typu *POST* údaje skriptu *coursesetup.php*, tento skript je velice rozsáhlý a kromě kódu pro zprostředkování založení kurzu do databáze obsahuje kódy pro změnu údajů kurzů, jejich publikaci či odstranění. Na počátku běhu tohoto skriptu jsou inicializovány objekty *CourseView*, *CourseContr*, *UserView* a *UserContr*, které jsou pak využívány v jeho jednotlivých větvích.

To, která část skriptu *coursesetup.php* proběhne, je ovlivněno tím, který na něj napojený formulář uživatel momentálně využívá.

Tlačítko typu *submit*, které spouští založení kurzu se jmenuje *basicdata* (základní data). Po jeho stisknutí dojde k předání dat z k němu příslušného formuláře, mezi kterými jsou odkaz pro přesměrování, jméno kurzu, e-mail lektora (zakládající uživatel) a popis kurzu. Tato data (kromě odkazu pro zpětné přesměrování) jsou poté předána metodě *createCourse* třídy *CourseContr*, která vytvoří záznam kurzu v databázi. Tím je vytvořen záznam kurzu, který je zatím nepublikovaný a většina jeho dat je nedoplněná.

Nově zapsaný kurz se zobrazí ve speciální části hlavní stránky aplikace, která nese název *Course dashboard.* Prostřednictvím tohoto přehledu kurzů je možné provádět změny na přihlášeným uživatelem vypsaných kurzech, zobrazení náhledu jednotlivých kurzů a jejich publikaci, její zrušení nebo mazání kurzů. Varianta zobrazení přehledu uživatelem vytvořených kurzů je ovlivněna prostřednictvím superglobálních proměnných typu *REQUEST*, podle toho, která z nich je, či není aktivní, je generována příslušná podoba stránky.




*Obrázek 11: Rozhraní správy kurzů*

V případě, že není aktivní žádný z příslušných parametrů *REQUEST*, je zobrazena základní podoba přehledu kurzů, ve které je každý řádek (každý jednotlivý zobrazený kurz) sám o sobě formulářem (viz Zdrojový kód 16), který při odeslání dat provede skript *coursesetup.php*. Každý řádek kromě výpisů dat týkajících se daného kurzu získaných z databáze obsahuje i dvě tlačítka *Customize* (Upravit) a *Publish*/*Unpublish* (Publikovat/Zrušit publikaci), která umožňují přechod ke změně údajů daného kurzu, anebo změnu stavu jeho zobrazení směrem k zájemcům o studium.

```
echo "<form action='coursesetup.php' method='post'>
<tr><td><a href='".$redirect_uri."?showcourse&csid=".$encrypted_csid."'>
<span>".$course['csname']."</span></a></td>";
echo "<td>";
if($course['cstype']==1){
    echo "<span>Individual</span>";
} else if ($course['cstype']==2){
    echo "<span>Group</span>";
} else {
    echo "";
}
echo "</td>
<td><span>".$course['csmaxstudents']."</span></td>";
if($course['csstart']==0){
    echo "<td></td>";
} else {
    echo "<td><span>".$course['csstart']."</span></td>";
}
if($course['csend']==0){
    echo "<td></td>";
} else {
    echo "<td><span>".$course['csend']."</span></td>";
}
if ($course['cspublication']){
    echo "<td><span>Published</span></td>";
} else {
    echo "<td><span>Hidden</span></td>";
}
echo "<td class='rightcell'><input type='hidden' name='url' value=$redirect_uri>
<input type='hidden' name='csid' value='".$course['csid']."'>
<button name='change' type='submit'>Customize</button></td>";
if(!empty($course['cstype']) && !empty($course['csmaxstudents']) && $course['cspublication']==0){
    echo "<td class='rightcell'><button name='publish' type='submit'>Publish</button></td></tr></form>";
} else if(!empty($course['cstype']) && !empty($course['csmaxstudents']) && $course['cspublication']==1){
    echo "<td class='rightcell'><button name='unpublish' type='submit'>Unpublish</button></td></tr></form>";
} else {
    echo "<td class='rightcell'></td></tr></form>";
}
```

*Zdrojový kód 16: Kód pro vygenerování řádku rozhraní pro správu uživatelem vyučovaných kurzů*

Při kliknutí na tlačítko *Customize* dojde k uložení ID příslušného kurzu do superglobální proměnné *SESSION[‘csid’]* a přesměrování uživatele na aktualizovanou stránku, která již obsahuje *PHP REQUEST customize.* Ten způsobí, že je při výpisu *Course dashboard* každé id kurzu testováno na shodu s *csid* uloženým v *SESSION*, pokud se shodují, je řádek daného kurzu vypsán s formulářem pro změnu údajů uložených v databázi – namísto nalezených údajů jsou vygenerovány vstupní prvky formuláře pro zadání nových hodnot.

V průběhu úpravy dat kurzu je znemožněna změna jeho stavu publikace (příslušné tlačítko je skryté a zároveň je tlačítko *Customize* nahrazeno tlačítkem pro uložení dat). Řádky kurzů, u kterých nebyla zvolena úprava údajů, zůstávají v původním stavu.


*Obrázek 12: Řádek kurzu během zadávání změny údajů*

Při kliknutí na tlačítko *Save data* (Uložit data) je opět spuštěn skript *coursesetup.php*, konkrétně jeho větev *save*. V této větvi dochází před změnou údajů v databázi k řadě ověření správnosti vkládaných dat.

První podmínkou pro zdárné uložení úprav je, že musí být vyplněn maximální počet studentů v rámci kurzu, tato hodnota musí být vyplněna i pokud je zvolena volba *Individual* pro typ kurzu, přičemž se v tomto případě maximální počet studentů změní na jednoho během kontroly tohoto údaje. Pokud jsou zadány hodnoty startovního a konečného data, musí být konečné datum vždy později nežli počáteční datum kurzu.

```
$coursecontr->courseUpd($csid, $csname, $cstutor, $csdesc, $cstype,
                        $csmaxstudents, $csstart, $csend, $cspublication);
if(isset($_POST['saveindividual'])){
    $redirect .= '?showcourse&csid='.numhash($csid).'&changei=success';
    header ('Location: '.filter_var($redirect, FILTER_SANITIZE_URL));
    exit();
} else {
    $_SESSION['csid'] = $csid;
    $redirect .= '?customize&change=success';
    header ('Location: '.filter_var($redirect, FILTER_SANITIZE_URL));
    exit();
}
```

*Zdrojový kód 17: Zápis změn údajů kurzu do databáze (kód slouží pro dva různé formuláře)*

Pro skupinové kurzy, jejichž modul není v ukázkové aplikaci kompletně realizován, existují doplňující podmínky – například pro ošetření situace, kdy už jsou zapsáni na kurzu studenti a lektor by se snažil nastavit nižší maximální počet studentů, nežli je v současné době počet registrovaných.

Pokud jakoukoli z podmínek pro uložení dat uživatelský vstup nesplní, dojde k přesměrování uživatele zpět a k vypsání chybové hlášky oznamující neúspěch a popisující zjištěnou chybu. Typ dané chyby je určen za pomocí superglobální proměnné *$_GET['change']*, pokud se tato proměnná například rovná řetězci '*baddates'*, pak chyba uživatelského zadání spočívala ve špatném zadání datumů začátku a konce kurzu.

V případě, že jsou splněny všechny podmínky, dojde k aktualizaci dat upravovaného kurzu v databázové tabulce *courses*. Tato změna je provedena pomocí metody *courseUpd* třídy *CourseContr* (viz Zdrojový kód 17). Po zdárném uskutečnění aktualizace hodnot v tabulce je odstraněna hodnota *csid* měněného kurzu ze *SESSION*, načež je uživatel přesměrován zpět na přehled kurzů a na *UI* je vypsána hláška o úspěšném dokončení procesu.

Z přehledu kurzů je možné se kliknutím na jméno kurzu dostat k jeho detailnímu zobrazení, odtud je možné analogicky provádět změny vlastností kurzu uložených v databázi (s možností změny popisu náplně kurzu navíc) nebo uskutečnit změnu stavu publikace kurzu. Z detailního zobrazení lektorem vytvořeného kurzu je možné kurz také vymazat, v současné verzi aplikace (dostupná v *příloze B*) nicméně používání této funkce může způsobit nekonzistence v datech událostí zapsaných jednotlivými uživateli, tento problém je nutné do budoucích verzí aplikace vyřešit, aby bylo možné kompletně mazat data zadaná uživateli bez ztráty konzistence databáze.

### 6.3.7 Zaslání žádosti o individuální hodinu

Ve zobrazení pro roli studenta je na hlavní stránce viditelné pouze vyhledávací pole, které je možné využít pro vyhledání lektory publikovaných kurzů. Pokud uživatel nevyplní vyhledávací pole a klikne na tlačítko *Search courses* (Vyhledat kurzy), dojde k zobrazení všech publikovaných kurzů z databáze.


*Obrázek 13: Vyhledávání kurzů a zobrazená nabídka*

Vyhledání kurzů z databáze je provedené pomocí funkce *showPublishedCourses* třídy *CourseView*. Vyhledávaný výraz je předán prostřednictvím proměnné *$_POST['search']*, její obsah je pro účely vyhledávání ošetřen zástupnými znaky, aby bylo možné vyhledávat řetězce obsahující zadaný výraz.

Vyhledané kurzy se zobrazí uživateli v rozhraní podobném tomu pro správu vytvořených kurzů (viz kapitola 6.3.6). Seznam nabízených kurzů namísto funkcí pro správu kurzů nabízí možnost prokliknout se na detail zvoleného kurzu (příslušným tlačítkem nebo kliknutím na jméno kurzu, viz Obrázek 13).

```
echo "<div id='course-search'><form action='index.php?search' method='post'>
<input type='text' name='search' placeholder='What type of course are you looking for?'>
<button type='submit' name='submit-search'>Search Courses</button></form></div>";

if(isset($_POST['search'])) {
    $search = '%'.$_POST['search'].'%';
    $course_data = $courseview->showPublishedCourses($search);
    if(!empty($course_data)){
        echo "<br><div class='offered-courses'><h2>Searched Courses</h2>";
        echo "<table><tr><th>Course name</th>";
        echo "<th>Course type</th>
        <th>Max students</th>
        <th>Starting date</th>
        <th>End date</th>
        <th>Course Tutor</th>
        <th></th>
        </tr>";
        foreach($course_data as $course){
            $encrypted_csid = numhash($course['csid']);
            if($course['cspublication']){
                $tutor_details = $userview->showUser($course['cstutor']);
                $tutor_name = $tutor_details[0]['userfirstname']." ".$tutor_details[0]['userlastname'];
                echo "<tr><td><a href='".$redirect_uri."?showoffcourse&csid=".$encrypted_csid."'>
                <span>".$course['csname']."</span></a></td>";
                echo "<td>";
                if($course['cstype']==1){
                    echo "<span>Individual</span>";
                } else if ($course['cstype']==2){
                    echo "<span>Group</span>";
                } else {
                    echo "";
                }
                echo "</td>";
                if($course['cstype']!=1 && $course['cstype']!=0){
                    echo "<td><span>".$course['csmaxstudents']."</span></td>
                    <td><span>".$course['csstart']."</span></td>
                    <td><span>".$course['csend']."</span></td>";
                } else {
                    echo "<td></td>
                    <td></td>
                    <td></td>";
                }
                echo "<td><span>".$tutor_name."</span></td>
                <td><a href='".$redirect_uri."?showoffcourse&csid=".$encrypted_csid."'>
                <button>Details</button></a></td>";
            }
        }
        echo "</table></div>";
    } else {
        echo "<div class='offered-courses'>";
        echo "<p>There are no courses that would match your search. Try again!</p>";
        echo "</div>";
    }
}
```

*Zdrojový kód 18: Vyhledávací formulář a generování seznamu nalezených kurzů*

U individuálních kurzů má pak uživatel možnost zaslat lektorovi žádost o termín lekce prostřednictvím tlačítka *Request lesson* (Vyžádat lekci). Po kliknutí na toto tlačítko je uživatel přesměrován na aktualizovanou verzi stránky s nastavenými hodnotami *$_REQUEST['calendar']* a *$_REQUEST['request']* (tato hodnota zajišťuje odlišení zobrazení od běžného procházení kalendáře).


*Obrázek 14: Detail kurzu s možností vyžádání termínu lekce*

Na dané stránce dojde k inicializaci kalendáře (viz Obrázek 9) přihlášeného uživatele (objekt třídy *Calendar* – třída definovaná v rámci vyvíjené aplikace, obsahuje metody a vlastnosti objektu *Calendar*, které slouží k zobrazení uživatelského kalendáře a jeho naplnění událostmi z databázové tabulky *events* – kompletní kód viz *příloha B: /Haste/classes/Calendar.php*).


*Obrázek 15: Výběr času plánované lekce*

Prostřednictvím zobrazeného rozhraní je možné vybrat den a čas, který by uživateli vyhovoval pro požadovanou lekci (v rámci současné verze aplikace je délka všech vytvářených událostí stanovena na jednu hodinu – navrhovaná řešení pro budoucí verze viz kapitola 5.2.2). Kliknutím na tlačítko *Request the lesson* (Vyžádat lekci – viz Obrázek 15) dojde ke zpracování žádosti o soukromou lekci a jejímu odeslání.

```
else if(isset($_POST['send'])){
    $lesson_start = $_POST['start'];
    $_SESSION['requested']['time'] = $lesson_start;
    $_SESSION['requested']['start'] = $_SESSION['requested']['date'].' '.$lesson_start.':00';
    $redirect = './mail.php?sendrequest';
    header('Location: '.filter_var($redirect, FILTER_SANITIZE_URL));
    exit();
}
```

*Zdrojový kód 19: Zpracování uživatelem požadovaného času lekce*

Zpracování uživatelem vložených dat probíhá nejprve prostřednictvím skriptu *coursesignup.php*. V rámci tohoto skriptu dojde k přípravě dat, kdy požadovaný začáteční čas lekce je spojen s požadovaným datem a uložen do *$_SESSION['requested']['start'],* načež dojde k přesměrování na *'./mail.php?sendrequest'*. Navíc je do *$_SESSION['requested']['time']* uložen čas lekce pro pozdější využití v e-mailu s žádostí o lekci.

```
if(isset($_REQUEST['sendrequest']) && isset($_SESSION['requested'])){
    $confirm_uri = $redirect_uri.'?confirmrequest&student&#61;'
        .$_SESSION['requested']['user'].'&csid&#61;'.$_SESSION['requested']['csid'].
        '&date&#61;'.$_SESSION['requested']['date'].'&time&#61;'.$_SESSION['requested']['time'];
    $redirect_home = './index.php?messagesent';
    $email_subject = 'Haste Application Lesson Request';
    $email_body = 'User '.$_SESSION['requested']['user'].' is asking for an individual lesson of the course '
        .$_SESSION['requested']['csname'].'<br> for '.$_SESSION['requested']['start'].'.';
    $email_body .= '<br>If you wish to accept this request, click the following link: '.$confirm_uri;
    send_email($_SESSION['requested']['user'], $_SESSION['requested']['cstutor'], $email_subject, $email_body, $client);
    unset($_SESSION['requested']);
    header('Location: '.filter_var($redirect_home, FILTER_SANITIZE_URL));
    exit();
}
```

*Zdrojový kód 20: Příprava* a odeslání *e-mailu se žádostí lektorovi*

Jakmile dojde k přesměrování, je vytvořen odkaz sloužící k potvrzení žádosti o lekci lektorem, který se skládá z odkazu na potvrzovací stránku a hodnot sloužících k následnému vytvoření žádosti, které byly do *$_SESSION['requested']* uloženy během procesu výběru kurzu a času pro požadovanou lekci. Nadefinovány jsou také předmět a text e-mailu, načež je volána funkce *send_email* (viz Zdrojový kód 21), která za využití *Gmail API* zpracuje a odešle připravenou zprávu z e-mailové adresy žadatele na adresu lektora.

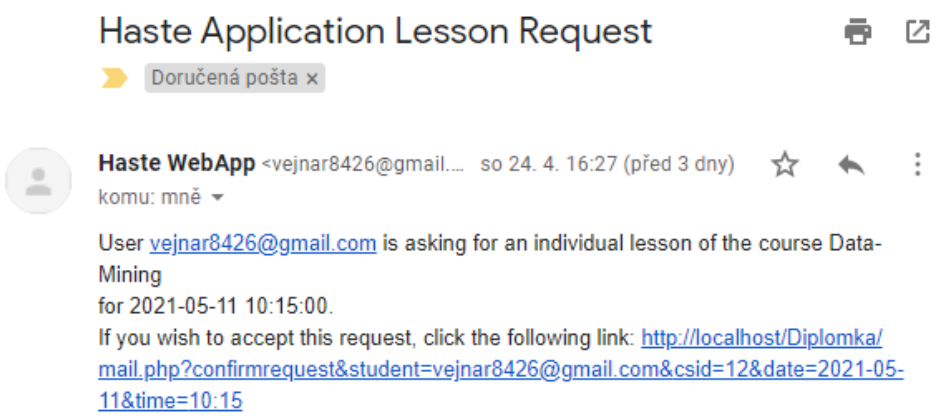
```
function send_email($sender, $recipient, $subject, $body, $client){
    $gmail_service = new Google_Service_Gmail($client);

    $strRawMessage = "From: Haste WebApp <$sender> \r\n";
    $strRawMessage .= "To: <$recipient>\r\n";
    $strRawMessage .= 'Subject: =?utf-8?B?' . base64_encode($subject) . "?=\r\n";
    $strRawMessage .= "MIME-Version: 1.0\r\n";
    $strRawMessage .= "Content-Type: text/html; charset=utf-8\r\n";
    $strRawMessage .= 'Content-Transfer-Encoding: quoted-printable' . "\r\n\r\n";
    $strRawMessage .= "$body\r\n";
    $mime = rtrim(strtr(base64_encode($strRawMessage), '+/', '-_'), '=');
    $msg = new Google_Service_Gmail_Message();
    $msg->setRaw($mime);
    $gmail_service->users_messages->send("me", $msg);
}
```

*Zdrojový kód 21: Funkce send_email*

### 6.3.8 Potvrzení žádosti o individuální lekci

Prostřednictvím automatizovaného e-mailu (viz kapitola 6.3.7) lektor kurzu obdrží žádost o soukromou lekci. Součástí této zprávy je i odkaz na speciální stránku aplikace určenou pro potvrzení uživatelských žádostí.




*Obrázek 16: Zpráva s žádostí o individuální hodinu*

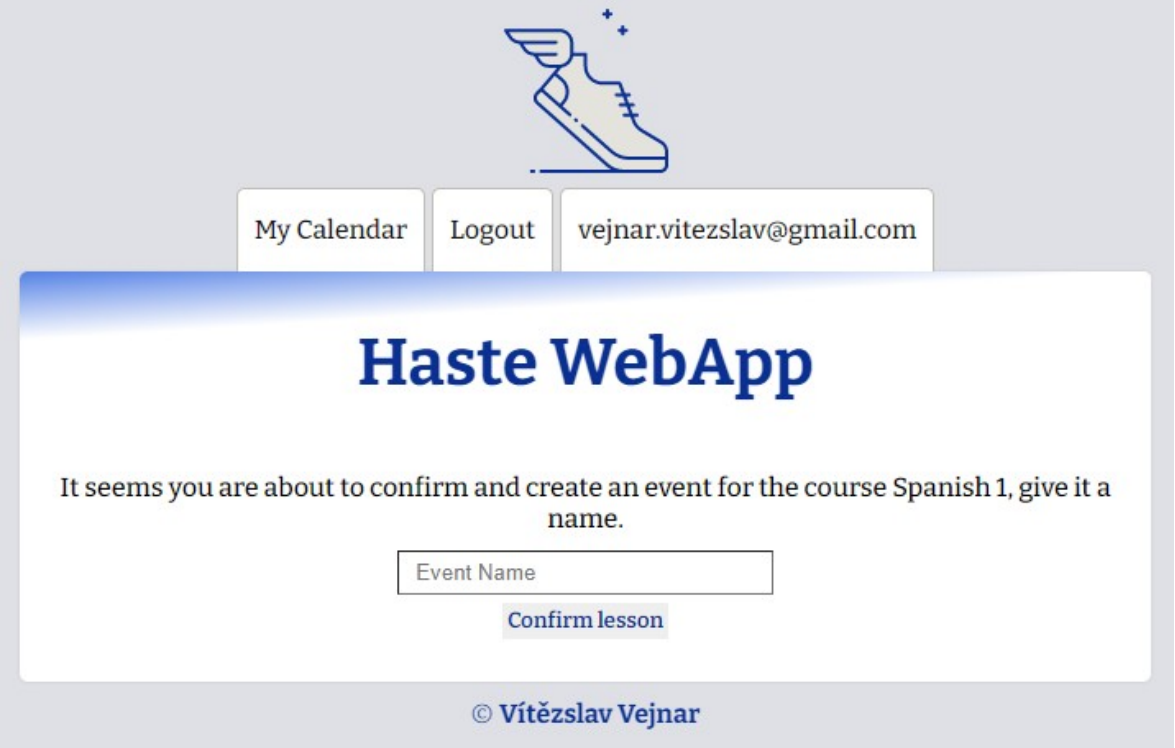
Po kliknutí na odkaz je uzivatel přenesen na zmíněnou stránku, kde dojde nejprve k uložení dat získaných z potvrzovacího odkazu do pole *$_SESSION['confirm']*. Potvrzovací odkaz v sobě v rámci superglobálních proměnných typu *GET* nese informace o žadateli o kurz, požadovaném kurzu a zvoleném termínu začátku lekce.

```
if(isset($_REQUEST['confirmrequest'])){
        $courseview = new Haste\CourseView();
        $course = $courseview->showCoursebyID($_GET['csid']);
        $csname = $course[0]['csname'];
        $startdatetime = $_GET['date'].'T'.$_GET['time'].':00.00';
        $endtime = $_GET['time'];
        $endtime = strtotime($endtime) + 60*60;
        $endtime = date('H:i:s', $endtime);
        $enddatetime = $_GET['date'].'T'.$endtime.'.00';
        $_SESSION['confirm']= array(
            'student' => $_GET['student'],
            'csid' => $_GET['csid'],
            'csname' => $csname,
            'start' => $startdatetime,
            'end' => $enddatetime
        );
    }
```

*Zdrojový kód 22: Zpracování údajů požadované události*

Součástí prvotního zpracování těchto dat je i odvození předpokládaného konce plánované události (o hodinu později nežli její začátek oba časové údaje jsou zároveň převedeny do formátu využitelného prostřednictvím *Google Calendar API* (podle standardu *RFC3359* pro hodnoty d*atetime*)).

Pokud je zjištěno, že není přihlášen žádný uživatel, je uživatel vyzván k přihlášení skrze *UI*. Jakmile dojde k autentizaci uživatele, nebo byla-li provedena již dříve, je vyzván k zadání jména potvrzované lekce před odesláním pokynu ke schválení. Po vyplnění políčka pro jméno je lekce potvrzena kliknutím na tlačítko *Confirm Lesson* (Potvrdit lekci).




*Obrázek 17: Pokyn k vložení jména zakládané události*

Po odeslání potvrzení lekce dojde ke spuštění skriptu *coursesignup.php*, konkrétně jeho větve navázané na aktivaci hodnoty *$_POST['confirmevent']*. Zde je zajištěno uložení zadaného jména události do proměnné *$_SESSION['evname']*, která je společně s polem *$_SESSION['confirm']* využita v dalším kroku po přesměrování zpět na stránku pro schvalování kurzů (*mail.php*).

```
else if(isset($_POST['confirmevent'])){
    $_SESSION['evname'] = $_POST['evname'];
    header('Location: '.filter_var($redirect, FILTER_SANITIZE_URL));
    exit();
}
```

*Zdrojový kód 23: Zpracování jména události*

Po přesměrování proběhne na dané stránce příprava dat pro předání funkci *create_meet* (viz Zdrojový kód 24), která je následně volána, tato funkce pracuje s *Google Calendar API*, při jejím použití dochází k vytvoření objektu události kalendáře Google, jehož vlastnosti jsou určeny pomocí hodnot nasbíraných během schvalovacího procesu.

```
function create_meet($client, $csname, $evname, $start, $end, $tutor, $student){
    $calendar_service = new Google_Service_Calendar($client);
    $reqid = generateRandomString();
    $evname .= ' from the course '.$csname;

    $event = new Google_Service_Calendar_Event(array(
        'summary' => $evname,
        'start' => array(
          'dateTime' => $start,
          'timeZone' => 'Europe/Prague'
        ),
        'end' => array(
          'dateTime' => $end,
          'timeZone' => 'Europe/Prague'
        ),
        'attendees' => array(
          array('email' => $student,
          'organizer' => FALSE),
          array('email' => $tutor,
          'organizer' => TRUE,
          'responseStatus' => 'accepted'
          )
        ),
        'reminders' => array(
          'useDefault' => TRUE
        ),
        'conferenceData' => array(
            'createRequest'=> array(
                'conferenceSolutionKey' => array(
                    'type' => 'hangoutsMeet'
                ),
                'requestId' => $reqid
            ),
        )
      ));

      $calendarId = 'primary';
      $event = $calendar_service->events->insert($calendarId, $event,
                ['conferenceDataVersion' => 1, 'sendUpdates' => 'all']);
      $meetid = $event->conferenceData['conferenceId'];
      return $meetid;
}
```

*Zdrojový kód 24: Funkce create_meet*

Vlastnost *summary* objektu *$event* slouží jako jeho jméno v rozhraní kalendáře Google, pro její určení je využito řetězce složeného ze jména události, propojovacího textu a jména kurzu, ke kterému se událost váže.

Začátek a konec události (vlastnosti *start* a *end*) jsou kromě času a data určeny také časovým pásmem. Jako návštěvníci události (vlastnost *attendees*) jsou zařazeni žadatel o kurz a lektor kurzu, přičemž vyučující je zároveň označen jako *organizátor* dané události, což mu dává právo zahájit posléze vytvořený hovor přes platformu *Google Meet,* narozdíl od studenta, který musí požádat o povolení ke vstupu do videohovoru.

Hodnoty v poli *conferenceData* slouží k inicializaci schůzky přes zvolenou aplikaci pro videokonference, kterou je *Google Meet* (tato volba je určena vlastností *conferenceSolutionKey.type* nastavenou na hodnotu *'HangoutsMeet'*). Parametr *createRequest.requestId* musí pro vytvoření nového hovoru pro každou zakládanou událost být unikátní pro každý běh, z tohoto důvodu je plněn náhodně vygenerovaným řetězcem *$reqid*.

Po inicializaci objektu události je určen kalendář, kam bude událost posléze vložena, v tomto případě jde o primární kalendář přihlášeného uživatele. Pro vložení události je využito metody *insert*, která spadá mezi metody pro správu událostí v *Google Calendar API*. Její průběh je možné ovlivnit různými parametry – povinným vstupem jsou pouze *$calendarid*, tedy označení kalendáře a tělo žádosti, tedy dříve vytvořený objekt *$event*, který v sobě sdružuje potřebné vlastnosti pro vytvoření naší lekce.

Další využité parametry představují doplňkové volby – využitím parametru *conferenceDataVersion* o hodnotě 1 zapříčiníme vytvoření odkazu na schůzku v *Google Meet* pro danou událost, parametr *sendUpdates* o hodnotě 'all' značí, že při jakékoli aktualizaci stavu schůzky obdrží její účastníci e-mailovou notifikaci.

Provedením takto nadefinovaného příkazu *insert* je vytvořená událost uložena do kalendářů obou účastníků a žadateli o individuální hodinu je zaslána pozvánka k události, čímž se dozví o schválení své žádosti.

Poslední akcí v *create_meet* je navrácení hodnoty *$meetid*, ve které je uložen identifikační řetězec hovoru vytvořeného pro událost v *Google Meet*. Tato hodnota je posléze využita při dokončení procesu vytváření události jako *id* záznamu události, který je vytvořen v databázové tabulce *events* pomocí metody *createEvent*.

Dále jsou do vytvořeny také záznamy v tabulce *userevents* pro oba účastníky lekce (*createUserEvent*), což umožní zobrazení vytvořené události také v interním kalendáři aplikace *Haste* obou účastníků společně s odkazem na připojení k videokonferenci *Google Meet*. Po dokončení těchto úkonů je uživatel přesměrován na hlavní stránku aplikace *Haste*, kde je zobrazeno oznámení o úspěšném potvrzení termínu události, také jsou vyčištěny proměnné *$_SESSION[‘confirm’]* a *$_SESSION[‘evname’]*.

```
else if (isset($_SESSION['evname'])){
            $redirect_home = './index.php?eventconfirmed';
            $csid = $_SESSION['confirm']['csid'];
            $csname = $_SESSION['confirm']['csname'];
            $evname = $_SESSION['evname'];
            $start = $_SESSION['confirm']['start'];
            $end = $_SESSION['confirm']['end'];
            $tutor = $oauth_return['token_data']['email'];
            $student = $_SESSION['confirm']['student'];

            $evid = create_meet($client, $csname, $evname, $start, $end, $tutor, $student);
            $coursecontr = new Haste\CourseContr();
            $coursecontr->createEvent($evid, $csid, $evname, $start);
            $usercontr = new Haste\UserContr();
            $usercontr->createUserEvent($tutor, $evid, 1);
            $usercontr->createUserEvent($student, $evid, 0);
            //print_r($event);
            unset($_SESSION['confirm']);
            unset($_SESSION['evname']);
            header('Location: '.filter_var($redirect_home, FILTER_SANITIZE_URL));
            exit();
        }
```

*Zdrojový kód 25: Uložení potvrzené události do Google kalendáře účastníků a interního kalendáře aplikace*

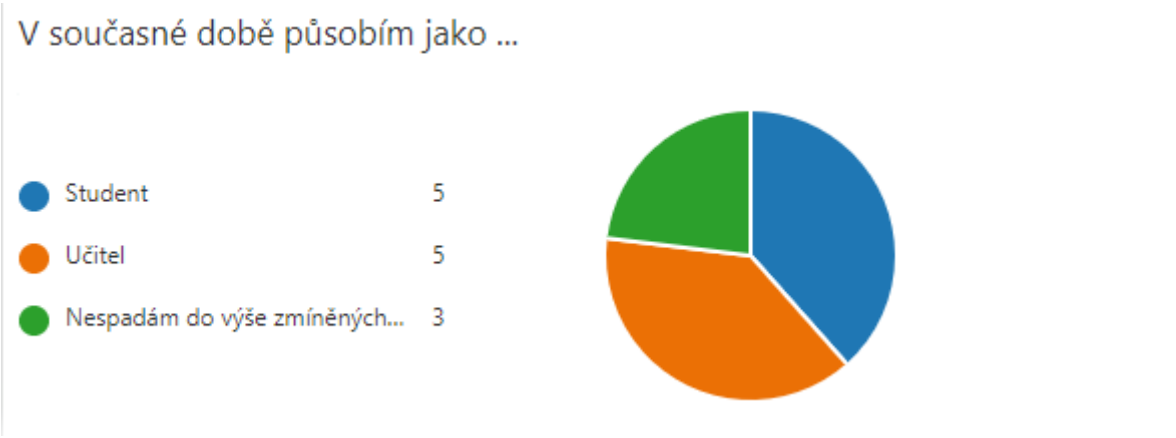
# 7 Testování uživatelské zpětné vazby

Po dokončení prací na ukázkovém modulu aplikace bylo jeho fungování demonstrováno prostřednictvím videohovoru se sdílením obrazovky třinácti potenciálním uživatelům. Po ukončení ukázky byli účastníci vyzváni ke shrnutí svých dojmů prostřednictvím vyplnění krátkého dotazníku. Kompletní soubor odpovědí je součástí *přílohy B*.

## 7.1 Vyhodnocení odpovědí účastníků

On-line dotazník, který účastníci videohovorů obdrželi k vyplnění, čítal celkem pět otázek, z toho tři uzavřené (dvě otázky typu *multiple-choice* a jedna s kvalitativním hodnocením pomocí Likertových škál) a dvě otevřené. Dotazník byl vytvořen v prostředí *MS Forms*.

První otázka slouží k rozdělení dotazovaných do třech skupin s ohledem na předpokládanou roli, kterou by mohli zastávat při využívání aplikace Haste. Při jejím vyhodnocení můžeme vidět, že mezi dotazovanými byli rovnoměrně zastoupeni *učitelé* a *studenti* (v každé skupině 5 dotázaných), dotazník vyplnily také tři osoby, které nespadají ani do jedné ze specifikovaných skupin.




Graf 1: Absolutní četnost odpovědí na první otázku dotazníku

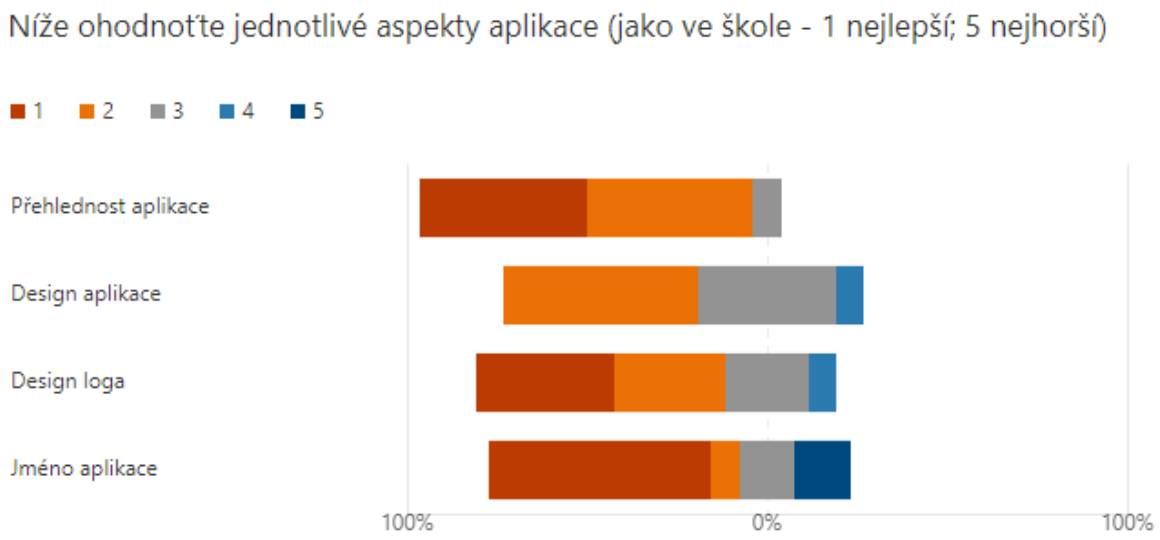
V druhé otázce měli uživatelé možnost ohodnotit některé z aspektů aplikace známkou od 1 do 5 jako ve škole (hodnocení 1 představuje nejlepší výsledek, hodnocení 5 naopak nejhorší) – hodnocenými vlastnostmi byly *přehlednost aplikace*, *design aplikace*, *design loga* a vhodnost *jména aplikace*.

Co se *přehlednosti* aplikace týče, reakce dotazovaných byly veskrze pozitivní, většina z nich nicméně vyjádřila názor, že by bylo možné tento aspekt vylepšit. Pro známku 1 hlasovalo šest osob, hodnocení 2 zvolilo taktéž šest dotazovaných. Jednomu člověku přišla aplikace přehledná méně a udělil tomuto aspektu známku 3.

*Design* aplikace se dle hodnocení ukázal jako slabší článek. Žádný z dotazovaných pro tuto vlastnost nezvolil známku 1. Číslem 2 ohodnotilo tento aspekt sedm osob. Hodnocení 3 zvolilo pět osob. Jeden účastník vybral známku 4.

Na vzhled samotného *loga* aplikace byly reakce o něco lepší, známkou 1 jej ohodnotilo pět osob, čtyři pak zvolily známku 2. Tři účastníci se přiklonili k hodnocení 3 a jeden dotazovaný vyjádřil svůj názor známkou 4.

Volba jména aplikace *Haste* byla podle osmi účastníků hodnotitelná známkou 1. Jeden dotázaný by toto jméno ohodnotil číslem 2. Možnost 3 zvolily dvě osoby. Dvěma účastníkům pak přijde jméno naprosto nevyhovující a hodnotili by ho známkou 5.

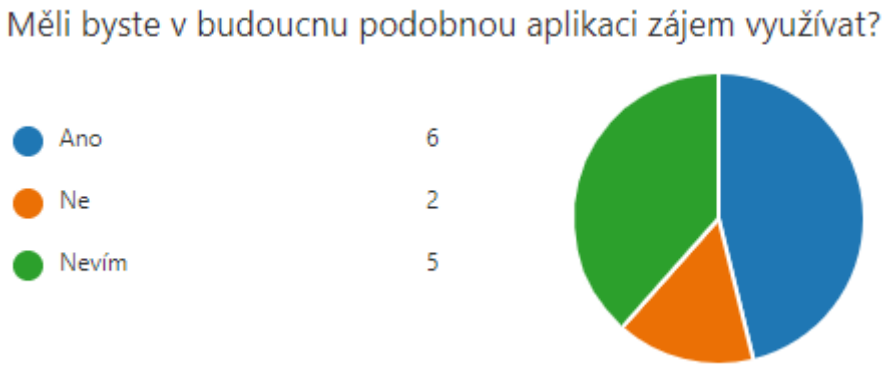



Graf 2: Znázornění relativní četnosti hodnocení jednotlivých aspektů aplikace

*Přehlednost* aplikace je pro její budoucí uživatele velice důležitým aspektem, veskrze pozitivní zpětná vazba na ni je tak důležitou zprávou, nicméně je vidět, že i v tomto směru by bylo možné zlepšení. Možnou cestou by mohla být úprava zobrazení nabízených kurzů, které v současné verzi aplikace působí značně monotónně a je lehké se v něm ztratit. Toto rozlišení by bylo možné provést například výpisem kurzů v podobě nabídky s ikonami namísto současného zobrazení v tabulce.

Slabým místem aplikace se ukázal být její vzhled. Přístupů k možným zlepšením je v tomto ohledu vícero. Barevné schéma aplikace by mohlo například zahrnout více kontrastní barvy. Dalším možným zlepšením pro *design* aplikace by mohl být její redesign na takzvaný *accordion* (akordeón), v tomto schématu jsou jednotlivé části webové stránky skryté za rozbalovací seznam a uživatel je může zobrazovat podle potřeby, což dělá stránku také přehlednější. Pro implementaci takového řešení by bylo vhodné například využití knihovny *Bootstrap* (sada nástrojů obsahující šablony založené na *HTML*, *CSS* a *JavaScriptu*).

Design *loga* a *jméno* aplikace obdržely poněkud rozporuplná hodnocení, i z této skutečnosti je vidět, že při přechodu do produkční verze je potřeba v těchto oblastech zvážit změny (více viz kapitola 8.2).




Graf 3: Absolutní četnost odpovědí na třetí otázku dotazníku

Poslední z uzavřených otázek dotazníku se zaměřila na to, jestli by dotazovaní rádi využívali v budoucnu aplikaci podobnou aplikaci *Haste*. Možnost *Ano* v tomto případě zvolilo šest účastníků, možnost *Ne* pouze dva z dotázaných. Zbylých pět zvolilo možnost *Nevím*.

Čtvrtá a pátá otázka dotazníku byly otevřené. Dotazovaní měli v odpovědi navrhnout, jaké designové změny by v aplikaci uvítali a jaké nové funkce by ocenili. Na otázku týkající se *designu* bylo poskytnuto celkem 13 odpovědí, otázka týkající se nových funkcí jich obdržela o jednu méně. Některými z těchto návrhů se zabývá následující kapitola.

## 7.2 Analýza navrhovaných zlepšení

Tato kapitola se věnuje zkoumání možného postupu řešení pro vybrané návrhy nových funkcí či změn designu aplikace, které vzešly z protokolů zpětné vazby.

### 7.2.1 Návrhy nových funkcí aplikace

Některé z návrhů se přímo dotýkají aktivit, které byly v ukázkovém modulu aplikace prozatím vynechány. Návrh zapracování možnosti *zaznamenávat prezenci studentů* může najít uplatnění při vedení skupinových kurzů. Tato funkce by při případné implementaci neznamenala velký problém. Jedním z jejích možných řešení je přidání sloupce pro zaznamenání přítomnosti na dané události do tabulky *userevents,* která už nyní propojuje uživatele a jednotlivé události.

Rozhraní pro zaznamenání přítomnosti na lekci by pak mohlo být rozšířením stávajícího kalendáře, kde by si lektor jednoduše rozkliknul událost v příslušném dni a kromě jejích podrobností by dostal i možnost zaznamenání prezence.

Velice užitečnou by se mohla stát také *možnost zrušení smluvené hodiny ze strany lektora s odůvodněním pro žáka* – s touto funkcí implementovanou v aplikaci by byl přidán další dobrý způsob komunikace se žákem. Kromě zrušení hodiny a jeho odůvodnění by mohl lektor také žákovi nabídnout náhradní termíny. Tato funkce by mohla být dostupná z rozhraní uživatelského kalendáře, kde by lektor po kliknutí na tlačítko například *Zrušit lekci* dostal možnost před odesláním oznámení zrušení lekce zadat odůvodnění pro studenta a nadefinovat možné náhradní termíny.

Zadané informace by následně byly odeslány na e-mail žáka společně s notifikací o zrušení lekce. Tato zpráva by mohla být řešena podobným způsobem jako žádost o individuální hodinu – opět prostřednictvím *Gmail API*, s pomocí *Google Calendar API* by pak bylo možné zrušit dříve vytvořenou událost v kalendáři.

Několik uživatelů zmínilo, že by ocenili *mobilní aplikaci*. Jít tímto směrem s aplikací *Haste* je jistě možné, příslušné *Google API* využívané při jejím prozatímním vývoji jsou dostupné i pro mobilní vývoj například v jazyce *Java.* Aplikaci je nicméně již v současné verzi možné využívat na mobilu, díky běhu v prohlížeči a responzivnímu designu je fakticky platformově nezávislá, nicméně dedikovaná mobilní verze aplikace by jistě přinášela větší uživatelský komfort.

V jedné z odpovědí byla zmíněna také *možnost dokončení celého schvalovacího flow v rámci aplikace*, tj. možnost využívání aplikace bez nutnosti přeskakování mezi samotnou aplikací a e-mailem. Toto řešení by jistě velice zvedlo uživatelskou přívětivost, mohlo by být implementováno jako součást *profilu* uživatele. V profilu účtu by kromě základních informací mohl mít každý lektor také seznam nevyřízených žádostí, které by přímo z daného přehledu mohl potvrzovat. Tatp funkce by vyžadovala přidání jedné databázové tabulky pro žádosti o individuální lekce, kde by byly provázány údaje o kurzu a obou účastnících možné smluvené hodiny.

Mezi navrhovanými funkcemi se také několikrát objevila možnost *nahrávání souborů souvisejících s kurzem* (průvodních či vyučovacích materiálů). Tato funkce by jistě byla velice užitečnou. Její přidání by mohlo spočívat v napojení *Google Disku lektora* na lektorem vyučované kurzy. Materiály ke kurzu by se poté mohly nahrávat do speciání složky uživatelova *cloudu*, kam by byl udělen přístup i studentům kurzu. K vytvoření tohoto řešení by bylo možné využít *Google Drive API*. Na daný disk by mohlo být ukládané i nahrané video ze schůzek kurzu v *Google Meet*.

Při implementaci funkce *sdílení souborů* výše zmíněným způsobem by bylo nutné důkladně ověřit zabezpečení soukromí lektora, aby studenti omylem nezískali přístup k jeho osobním souborům. Zároveň by možná tato funkce narazila na malou kapacitu *Google Disku* v bezplatné verzi.

### 7.2.2 Návržené změny designu

I přes relativně dobré hodnocení *přehlednosti* aplikace ve druhé otázce dotazníku byla tato vlastnost předmětem mnoha navrhovaných změn designu *UI*. Někteří účastníci ukázek shledali barevné schéma moc jednotvárným, rozlišili by více například jednotlivé části výpisu kurzů. Specificky navrženou změnou bylo například *odlišení barvy písma názvu kurzů* od ostatních položek. V detailu kurzu pro zájemce by pak někteří dotázaní přemístili tlačítko pro zapsání se na kurz/vyžádání kurzu mimo současné zobrazení v tabulce s detaily kurzu. Součástí jednoho z návrhů bylo také použití veselejších barev, nicméně v tomto případě se jedná spíše o osobní preferenci. Vhodnou úpravou barev by ale nejspíše bylo již výše zmíněné využití více kontrastních odstínů.

Někteří z účastníků zkoumání zpětné vazby konstatovali také, že design se zdá být velice statický a rádi by viděli na stránkách pohyblivé prvky. Jedním z konkrétních příkladů bylo *pohyblivé záhlaví.* Tímto konkrétním návrhem uživatel nejspíše navrhoval využití tzv. karuselu (*slideshow* grafiky) v záhlaví stránek aplikace. Realizace tohoto prvku by byla možná například za pomoci knihovny *Bootstrap*. Zahrnutí více pohyblivých prvků do *UI* by jistě udělalo aplikaci atraktivnější pro uživatele, v současné verzi by nicméně znamenalo nutnost přesunutí některých procesů z *back-endu*, do *front-endu*, a tak by se jednalo o náročnější úpravu.

Objevil se i ohlas, který zmiňoval, že design působí sice čistě, ale možná až příliš *korporátně.* Jako řešení bylo v této souvislosti navrženo využití sytých barev v kontrastu s čistě bílou. Ten samý uživatel by se také přiklonil k volbě *barevnějšího loga*. Další z návrhů řešil prvky komunikace s uživatelem, jednotlivé texty upozorňující na chyby vstupů či úspěšné dokončení procesů by dle něj bylo vhodné zvýraznit. Přidal by také upozornění na automatické přepsání zadaného počtu účastníků individuálních lekcí na jednoho.

Jako možný problém se ukázalo také, že někdy se po rozkliknutí objeví nová část stránky v jiném místě, nežli původní nabídka, zejména u zobrazení rozhraní pro úpravu kurzů přímo z vyhledaných kurzů v nabídce (situace, kdy lektor rozklikne v nabídce svůj vlastní kurz). Jako řešení by mohla v tomto případě posloužit změna struktury webové stránky, anebo použití odkazů na identifikátory oddílů v rámci stránky.

Častým předmětem námětů ke změně se stala také *přehlednost kalendáře.* Matoucími se ukázala tlačítka pro přesun mezi měsíci, v současném designu některým dotázaným připadala jako součást nadpisu kalendáře, bylo by tak dobré je znatelněji odlišit. Současné zobrazení událostí má také dle některých testujících nedostatečně dořešené zobrazení událostí, objevil se mimo jiné návrh umožnit zobrazení jednotlivých událostí již prostřednictvím políček znázorňujících dny a ne až po kliknutí na ně. Někteří uživatelé by také ocenili zvýraznění vybraného dne v rámci kalendáře a úpravu formátu zobrazení data.

# 8 Analýza nákladů a branding aplikace

Tato kapitola se zaměří na ekonomické a marketingové aspekty vývoje navrhované aplikace, konkrétně na náklady spojené s jejím vývojem a volbu jejího názvu a loga.

## 8.1 Náklady na vývoj

V této podkapitole budou řešeny náklady na vývoj ukázkového modulu aplikace (viz kapitola 4.4). Pro jejich rozbor a znázornění byla použita metoda *WBS*. Proces vývoje byl pro účely této analýzy rozdělen do čtyřech na sebe navazujících segmentů.

Pro účely výpočtu nákladů na mzdy za odvedenou práci byla zvolena taxa 300 korun za člověkohodinu, odvody sociálního a zdravotního pojištění za zaměstnavatele představují v našem schématu 34 % mzdy. Předpokládáme, že vývojář má všechno potřebné vybavení již k dispozici a pro vývoj a návrh aplikace používá bezplatně dostupný software. Opotřebení vybavení je zahrnuto jako součást režijních nákladů. Jeden člověkoden čítá osm hodin.

První částí procesu výroby ukázkového modulu aplikace je *Analýza use-cases a návrh databáze*, v průběhu které dochází k rozboru aktivit uživatelů v prostředí aplikace a návrhu pro její fungování potřebných datových struktur. Jako potřebný čas prací pro tyto úkony bylo stanoveno pět člověkohodin. Náklady na mzdy tak čítají 1 500 *Kč*, povinné odvody ze strany zaměstnavatele by pak čítaly 510 *Kč.* Celkové náklady na tento segment pak představují částku 2 010 *Kč.*

Nejrozsáhlejší částí vývoje jsou práce na *back-endu* aplikace. Jejich rozsah byl odhadnut na sedm člověkodní (56 člověkohodin). Mzdové náklady tohoto segmentu tak čítají 16 800 *Kč*. Povinné odvody pak představuje částka 5 712 *Kč*. Čas potřebný pro vývoj *front-endu* byl odhadnut jako jeden člověkoden, náklady na něj tak představuje částka 2 400 *Kč* s povinnými odvody 816 *Kč*.

Testování funkčnosti aplikace byla přisouzena potřeba využití třech člověkohodin. Náklady na mzdy pro tento obnos práce čítají 900 *Kč*. Povinné odvody za zaměstnavatele jsou vypočítány jako 306 *Kč*.

*Tabulka 5: Analýza nákladů jednotlivých částí projektu metodou WBS*

| Vývoj aplikace | Peněžní prostředky |
|---|---|
| Analýza use-cases a návrh databáze | |
| Vývojář – 5 h | 1 500,00 Kč |
| | 510,00 Kč |
| Vývoj back-endu | |
| Vývojář – 7 MD (56 h) | 16 800,00 Kč |
| | 5 712,00 Kč |
| Vývoj front-endu | |
| Vývojář – 1 MD | 2 400,00 Kč |
| | 816,00 Kč |
| Testování aplikace | |
| Tester – 3 h | 900,00 Kč |
| | 306,00 Kč |
| **CELKEM :** | **28 944,00 Kč** |

Po celkovém sečtení tak *osobní náklady* čítají 28 944 *Kč*, z toho 21 600 *Kč* představují mzdové náklady a 7 344 *Kč* povinné odvody za zaměstnavatele. Tyto náklady jsou jedinými přímými náklady vývoje ukázkového modelu aplikace, jelikož v rámci projektu nebyla identifikována žádná potřeba nakupování *materiálu*, nového *hmotného majetku* či *subdodávek*.

Pro řešení režijních nákladů byl stanoven paušální poměr 34 % z částky přímých nákladů. Nepřímé náklady v podobě režie tak představují 9 840,96 *Kč*. Celkové náklady na projekt tak představují 38 784,96 *Kč*.

*Tabulka 6: Souhrn celkových nákladů na vývoj ukázkové aplikace*

| Náklady | |
|---|---|
| **Přímé náklady** | |
| **Osobní náklady** | 28 944,00 Kč |
| Mzdy | 21 600,00 Kč |
| Odvody | 7 344,00 Kč |
| **Materiál** | 0,00 Kč |
| **Subdodávky** | 0,00 Kč |
| **Hmotný majetek** | 0,00 Kč |
| **Nepřímé náklady** | |
| **Režie (34%)** | 9 840,96 Kč |
| **CELKEM** | 38 784,96 Kč |

## 8.2 Logo a název aplikace

Jak už bylo zmíněno v úvodu kapitoly 6, pracovní název vyvíjené aplikace zní *Haste* (spěch). Tento název byl vybrán, jelikož hlavním cílem vyvíjené aplikace je zajistit rychlou a efektivní komunikaci mezi nabídkou vzdělávání a poptávkou po něm. Tento název nicméně na první pohled neevokuje vzdělávání, a tak pro produkční verzi aplikace by bylo nutné jej pozměnit či doplnit, aby více evokoval myšlenku efektivity v on-line vzdělávání.

*Obrázek 18: Logo pracovní verze aplikace Haste (Icons made by Flat-Icons, 2021)*

Logo zvolené pro současnou verzi aplikace taktéž zrcadlí myšlenku rychlosti a efektivity a má podobu nakročené tenisky s křídly. Stejně jako u názvu aplikace, i v rámci loga aplikace by bylo nutné při přechodu do produkce uskutečnit změny, které by zdůraznily aspekt efektivity v on-line výuce.

Pro změnu jména aplikace je dalším důvodem také skutečnost, že pod jménem Haste již na trhu vystupuje firma vyrábějící software orientovaný na vylepšení stability a rychlosti internetového připojení pro využití hráči on-line počítačových her.

# Závěr

V současné složité době je při množství on-line práce, jejíž nároky jsou kladeny na většinu pracovníků ve vzdělávání, nutno hledat efektivnější cesty komunikace. Jednou z nich by mohla být i v této práci navržená aplikace.

Výstupy teoretické části nám ukazují, že podobné řešení by mohlo být realizováno s napojením na větší množství aplikací pro videohovory, což významně rozšiřuje možnosti jeho budoucího využití. V praktické části byl upřednostněn pro propojení s vyvíjenou aplikací nástroj *Google Meet*.

Teoretická část práce se dále věnovala také problematice on-line výuky, jejího historického vývoje a role v současnosti. Jedním z témat bylo i vymezení pojmu webové aplikace. Obsah daných kapitol může pomoci zasadit záměr vývoje navržené aplikace do širšího kontextu a zorientovat se v jejích komponentech.

V praktické části navržená aplikace spolu s vyvinutým ukázkovým modulem mohou sloužit jako výchozí bod pro širokou škálu řešení uplatnitelných pro time-management v rámci jak vzdělávání, tak podnikové praxe.

Aplikace s pracovním názvem *Haste* by zřejmě našla největší uplatnění jako modul pro zajišťování kurzů v rámci nějaké sítě (chcete-li *on-line tržiště*) sdružující dohromady soukromé lektory a zájemce o volnočasové aktivity. Tato síť by lektorům mohla posloužit jako místo efektivního prodeje on-line kurzů. Vhodné však může být i využití aplikace v e-shopech institucí nabízejících vzdělávání za úplatu či jako nástroj pro efektivní sjednávání on-line schůzek využívaný společně se zavedeným *LMS*.

Ukázkový modul, který byl vyvinut, zobrazuje kompletní proces realizace individuálně vyučovaných kurzů v rámci navrhované aplikace. Pokud bychom se rozhodli stavět na tomto modulu se záměrem vytvořit systém sdružující soukromé lektory a zájemce o jejich nabídku, museli bychom se kromě zprovoznění modulu pro skupinové kurzy zaměřit také na integraci nutných prostředků prodeje kurzů (evidence cen kurzů, systém uživatelského kreditu, hodnocení lektorů, integrace platební brány…) i na prvky uživatelské přívětivosti (*UX*), pro její zlepšení bychom mimo jiné mohli implementovat návrhy, které vzešly z měření zpětné vazby uživatelů, kteří se zúčastnili demonstrace užívání ukázkového modulu.

Díky využití *Google API* bylo možné pro vývoj aplikace využít široké množství zdrojů aplikací dostupných skrze toto rozhraní. Autentizaci skrze účet *Google* se toto řešení zpřístupňuje široké škále uživatel. V této práci vyvinutý modul aplikace může sloužit také jako reference pro budoucí vývojáře řešení spolupracujících s nástrojem *Google Meet*.

# Seznam použité literatury

*Businesses at Work: powered by Okta*, 2021. Okta [online]. San Francisco: Okta [cit. 2021-03-04]. Dostupné z: https://www.okta.com/businesses-at-work/2021/#2020-most-popular-apps

*Ceny Služby Google Meet*, 2021. Google Meet [online]. Dublin: Google [cit. 2021-02-25]. Dostupné z: https://apps.google.com/intl/cs/meet/pricing/

*Co je to e-learning a jaká je jeho historie*, 2016. Skolenibozp.cz [online]. Praha: CRDR spol. s r.o. [cit. 2021-02-21]. Dostupné z: https://www.skolenibozp.cz/aktuality/co-je-to-elearning-a-jaka-je-jeho-historie/

*Developer Guide (Web) · Jitsi Meet Handbook*, 2021. Github [online]. San Francisco: GitHub [cit. 2021-03-01]. Dostupné z: https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-web

*Difference between Website and Web Application*, 2021. Guru99 [online]. Ahmedabad: Guru99 [cit. 2021-03-07]. Dostupné z: https://www.guru99.com/difference-web-application-website.html

DOLEJŠ, Jan, 2019. *Co to je Microsoft Teams?*. Honza Dolejš [online]. Praha: Blog [cit. 2021-02-28]. Dostupné z: https://honzadolejs.cz/blog/co-to-je-microsoft-teams/

*Doporučené postupy pro školy v období vzdělávání na dálku*, 2020. Msmt.cz [online]. Praha: Ministerstvo školství, mládeže a tělovýchovy [cit. 2021-02-16]. Dostupné z: https://www.msmt.cz/doporucene-postupy-pro-skoly-v-obdobi-vzdelavani-na-dalku

*Education Plan - Zoom*, 2021. Zoom [online]. San Jose: Zoom Video Communications [cit. 2021-03-04]. Dostupné z: https://explore.zoom.us/education

*Events | Calendar API | Google Developers*, 2021. Google Calendar API [online]. Mountain View, Kalifornie: Google [cit. 2021-02-25]. Dostupné z: https://developers.google.com/calendar/v3/reference/events

GALLAGHER, Ryan, 2020. *Zoom for Online Learning Updates: Expanded Access for Schools*. Zoom|Blog [online]. San Jose: Zoom Video Communications [cit. 2021-03-04]. Dostupné z: https://blog.zoom.us/how-to-use-zoom-for-online-learning/

*Google Workspace for Education editions*, 2021. Google for Education [online]. Dublin: Google [cit. 2021-02-25]. Dostupné z: https://edu.google.com/products/workspace-for-education/editions/

HUGHES, John, 2020. Zoom vs Microsoft Teams vs Google Meet: Top Video Conferencing Apps Compared. Codeinwp [online]. Bukurešť: CodeInWP [cit. 2021-02-28]. Dostupné z: https://www.codeinwp.com/blog/zoom-vs-microsoft-teams-vs-google-meet/

*Icons made by Flat-Icons*, 2021. Flaticon [online]. Malaga: freepik company [cit. 2021-4-25]. Dostupné z: https://www.flaticon.com/authors/flat-icons

*Introduction to Zoom API*, 2021. Zoom|Developer [online]. San Jose: Zoom Video Communications [cit. 2021-03-04]. Dostupné z: https://marketplace.zoom.us/docs/api-reference/introduction

IRWIN, Luke, 2020. *Is Zoom safe to use?*. IT Governance Blog [online]. Ely: IT Governance [cit. 2021-03-04]. Dostupné z: https://www.itgovernance.co.uk/blog/is-zoom-safe-to-use

*Jitsi as a Service*, 2020. 8x8 [online]. Campbell: 8x8 [cit. 2021-03-01]. Dostupné z: https://jaas.8x8.vc/

*Jitsi Meet*: *Bezpečné a kvalitní setkávání*, 2020. Jitsi [online]. Campbell: 8x8 [cit. 2021-03-01]. Dostupné z: https://meet.jit.si/

KOLOWICH COX, Lindsay, 2020. *Web Design 101: How HTML, CSS, and JavaScript Work*. Hubspot [online]. Cambridge: Hubspot [cit. 2021-03-08]. Dostupné z: https://blog.hubspot.com/marketing/web-design-html-css-javascript

*Microsoft Teams*, 2021. Microsoft [online]. Redmond: Microsoft [cit. 2021-02-28]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-teams/group-chat-software

*Microsoft Teams pro vzdělávání*, 2021. Microsoft [online]. Redmond: Microsoft [cit. 2021-02-28]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-teams/education

MILLER, Georgie, 2014. *History of Distance Learning*. WorldWideLearn [online]. Hoboken, NJ: EducationDynamics [cit. 2021-02-18]. Dostupné z: http://www.worldwidelearn.com/education-articles/history-of-distance-learning.html

*MVC Components*, 2021. Tutorialspoint.com [online]. Haidarabád: Tutorials Point [cit. 2021-4-25]. Dostupné z: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

*PHP vs JavaScript: Must Know Differences*, 2021. Guru99 [online]. Ahmedabad: Guru99 [cit. 2021-03-08]. Dostupné z: https://www.guru99.com/php-vs-javascript.html

PILKOVÁ, Petra, 2016. *Online výuka (nejen) individuálně vzdělávaných žáků: ve Škole na konci světa*. Brno. Závěrečná práce. Masarykova Univerzita, Pedagogická fakulta. Vedoucí práce Doc. PaedDr. Hana Horká, CSc.

PRŮCHA, Jan, Eliška WALTEROVÁ a Jiří MAREŠ, 2013. *Pedagogický slovník*. 7., aktualiz. a rozš. vyd. Praha: Portál. ISBN 978-80-262-0403-9.

*Specifika online komunikace: Online disinhibition effect a flame wars*, 2019. EdTech KISK [online]. Brno: KISK FF MU [cit. 2021-02-22]. Dostupné z: https://medium.com/edtech-kisk/specifika-online-komunikace-a5646048cd02

*Srovnání online možností Microsoft Teams | Microsoft Teams*, 2021. Microsoft [online]. Redmond: Microsoft [cit. 2021-02-28]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-teams/compare-microsoft-teams-options

*Synchronous vs. Asynchronous*, 2015. Pierce College eCampus [online]. Los Angeles: Pierce College Los Angeles [cit. 2021-02-22]. Dostupné z: https://pierce.instructure.com/courses/983325/pages/synchronous-vs-asynchronous?module_item_id=12922163

*Three-Tier Architecture*, 2020. IBM Cloud Learn Hub [online]. New York: IBM [cit. 2021-03-07]. Dostupné z: https://www.ibm.com/cloud/learn/three-tier-architecture

*Use the Microsoft Graph API to work with Microsoft Teams - Microsoft Graph v1.0 | Microsoft Docs*, 2020. Microsoft Docs [online]. Redmond: Microsoft [cit. 2021-02-28]. Dostupné z: https://docs.microsoft.com/en-us/graph/api/resources/teams-api-overview?view=graph-rest-1.0

VODNIK, Sasha, 2017. *HTML for web development: Building the bones of your website*. General Assembly Blog [online]. New York: General Assembly [cit. 2021-03-08]. Dostupné z: https://generalassemb.ly/blog/html-web-development-building-bones-website/

*Voice, Video, Chat, Contact Center | 8x8*, 2020. 8x8 [online]. Campbell: 8x8 [cit. 2021-03-01]. Dostupné z: https://www.8x8.com/

*Web application (Web app)*, 2019. Search Software Quality [online]. Atlanta: Techtarget [cit. 2021-03-04]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app

*Web Server vs. Application Server*, 2020. IBM Cloud Learn Hub [online]. New York: IBM [cit. 2021-03-07]. Dostupné z: https://www.ibm.com/cloud/learn/web-server-vs-application-server

*What end-to-end encryption is, and why you need it*, 2021. Kaspersky Daily [online]. Moskva: Kaspersky Lab [cit. 2021-03-01]. Dostupné z: https://www.kaspersky.com/blog/what-is-end-to-end-encryption/37011/

*What Is a Database?*, 2021. Oracle [online]. Austin: Oracle [cit. 2021-03-07]. Dostupné z: https://www.oracle.com/database/what-is-database/

*What is online education?*, 2021. India Education [online]. Pune: India Education [cit. 2021-02-16]. Dostupné z: https://www.indiaeducation.net/online-education/articles/what-is-online-education.html

*Zákon č. 561/2004 Sb.*, o předškolním, základním, středním, vyšším odborném a jiném vzdělávání (školský zákon), ve znění pozdějších předpisů. In: *Sbírka zákonů*. 24.9.2004. ISSN 1211-1244.

*Zákon č. 111/1998 Sb.*, o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů. In: *Sbírka zákonů*. 22.4.1998. ISSN 1211-1244.

*Zoom Video Conferencing Plans & Pricing*, 2021. Zoom [online]. San Jose: Zoom Video Communications [cit. 2021-03-04]. Dostupné z: https://zoom.us/pricing

# Seznam příloh

*Příloha A* Plná velikost vybraných diagramů zobrazených v diplomové práce

1. Diagram aktivit pro proces smluvení termínu individuální lekce
2. Entity Relationship Diagram (ERD) navrhované databáze
3. ERD upravené do podoby odrážející strukturu tabulek v databázi

*Příloha B* DVD se zdrojovými soubory aplikace Haste a výstupy protokolu zpětné vazby - *Adresářová struktura:*

- *./Haste* Aplikace Haste
- *./Readme.txt* Pokyny ke zprovoznění ukázkové aplikace na lokálním serveru
- *./Feedback. xlsx* Kompletní soubor odpovědí sebraných při testování zpětné vazby