

Univerzita Jana Evangelisty Purkyně  
v Ústí nad Labem  
Přírodovědecká fakulta



Vývoj softwarové platformy pro plánování  
výuky

BAKALÁŘSKÁ PRÁCE

**Vypracoval:** Matěj Kaška

**Vedoucí práce:** Ing. Mgr. Pavel Beránek

**Studijní program:** Aplikovaná informatika

**Studijní obor:** Aplikovaná informatika

ÚSTÍ NAD LABEM 2025



Namísto žlutých stránek vložte digitálně podepsané zadání kvalifikační práce poskytnuté vedoucím katedry.

Zadání musí zaujímat právě dvě strany.

Zadání je nutno vložit jako PDF pomocí některého nástroje, který umožňuje editaci dokumentů (se zachováním elektronického podpisu).

V Linuxe lze například použít příkaz pdftk.



## **Prohlášení**

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a použil jen pramenů, které cituji a uvádím v přiloženém seznamu literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., ve znění zákona č. 81/2005 Sb., autorský zákon, zejména se skutečností, že Univerzita Jana Evangelisty Purkyně v Ústí nad Labem má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Jana Evangelisty Purkyně v Ústí nad Labem oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladu, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

V Ústí nad Labem dne 17. dubna 2025

Podpis: .....



Děkuji vedoucímu práce Ing. Mgr. Pavlu Beránkovi  
za neocenitelné rady a pomoc při tvorbě bakalářské práce.



---

## VÝVOJ SOFTWAROVÉ PLATFORMY PRO PLÁNOVÁNÍ VÝUKY

### **Abstrakt:**

Abstract práce uvádí základní téma práce a především její výstupy. Rozsah abstraktu by měl být alespoň osmdesát slov (abstrakty a klíčová slova se však musí vejít na stránku).

**Klíčová slova:** tři až pět klíčových slov resp. termínů, která usnadní případné vyhledávání závěrečné práce, v pořadí od obecnějších ke konkrétnějším

## DEVELOPMENT OF A SOFTWARE PLATFORM FOR LESSON PLANNING

### **Abstract:**

The abstract states the main topic of the thesis and mainly its outcomes. The length of the abstract should be at least eighty words (however, abstracts and keywords must fit on a page)

**Keywords:** three to five key words or terms to facilitate a possible search of the thesis, in order from more general to more specific



# Obsah

<b>1. Úvod a cíle závěrečné práce</b>	<b>13</b>
<b>2. Přehled současného stavu problematiky a existujících aplikací</b>	<b>15</b>
2.1. Planbook . . . . .	15
2.2. Common Curriculum . . . . .	17
2.3. Lesson Planner PH . . . . .	19
2.4. Slides Go AI Lesson Plan Generator . . . . .	20
<b>3. Teoretická východiska</b>	<b>21</b>
3.1. Výukový proces . . . . .	21
3.2. Výukové aktivity . . . . .	23
3.3. Životní cyklus softwaru . . . . .	32
<b>4. Praktická část</b>	<b>39</b>
4.1. Návrh architektury, komponent a schématu databáze platformy . . . . .	39
4.2. Odůvodnění výběru technického zásobníku . . . . .	54
4.3. Implementace softwaru pro plánování výuky . . . . .	59
4.4. Implementace webového agregátoru výukových plánů . . . . .	80
4.5. Uživatelský manuál . . . . .	88
<b>5. Testování softwaru uživateli</b>	<b>99</b>
<b>6. Diskuse a výsledky</b>	<b>101</b>
<b>7. Závěr</b>	<b>103</b>
<b>8. Externí přílohy</b>	<b>105</b>
<b>9. Základní členění závěrečné práce</b>	<b>107</b>
9.1. Kapitoly . . . . .	107
9.2. Sekce resp. podkapitoly . . . . .	107
9.3. Rozsah závěrečné práce . . . . .	108
<b>10. Typografie</b>	<b>109</b>
10.1. Písmo . . . . .	109
10.2. Základní typografická pravidla . . . . .	110

10.3. Znaky, které nenajdete na klávesnici . . . . .	110
<b>11. Grafika</b>	<b>113</b>
<b>12. Sazba ukázek kódu</b>	<b>115</b>
<b>13. Citace</b>	<b>117</b>
13.1. Označování citací . . . . .	118
13.2. Bibliografický záznam . . . . .	119
13.3. Často kladené otázky . . . . .	123
<b>14. Zhodnocení</b>	<b>127</b>
<b>15. Závěr</b>	<b>129</b>
<b>A. Externí přílohy</b>	<b>131</b>
<b>B. Další přílohy</b>	<b>133</b>

# 1. Úvod a cíle závěrečné práce

Tato závěrečná práce vznikla jako součást rozsáhléjšího výzkumného a vývojového projektu zaměřeného na vytvoření moderního softwarového nástroje pro plánování výuky. K myšlence vývoje samotné aplikace *EDUBO* (software pro plánování výuky) a *Eduklub – plány výuky* (webový agregátor sdílených výukových plánů) jsem se dostal díky nabídce pedagoga Pavla Beránka, který mě přizval do projektu na pozici backend vývojáře. Ačkoliv jsem nikdy nepůsobil v pedagogickém prostředí a neřešil osobně problémy s přípravou výuky, zaujala mě absence robustního digitálního nástroje, který by tuto oblast kvalitně pokrýval.

Na trhu v době zahájení vývoje chyběl software, který by v sobě spojoval vlastní vizuální editor výukového plánu, možnost přímé práce s edukačním obsahem a zároveň podporoval sdílení výukových plánů mezi pedagogy. Hlavní inovací *EDUBO* je především vlastní editor s *drag and drop* interakcí, v němž lze sestavovat hodiny pomocí výukových aktivit vytvořených pedagogy z Pedagogické fakulty Univerzity Karlovy. Tyto aktivity lze dále upravovat podle potřeby a přizpůsobit konkrétní výuce. Dalším důležitým prvkem je tzv. *studentský pohled*, který žákům umožňuje nahlédnout na proběhlé, aktuální nebo budoucí hodiny a tím podpořit jejich přípravu, orientaci i reflexi.

Primární cílovou skupinou aplikace jsou začínající učitelé, kteří se často potýkají s nedostatkem metodické opory při plánování výuky. *EDUBO* jim nabízí přístup ke sdíleným plánům, které lze jednoduše převzít a upravit, čímž výrazně šetří čas a zvyšují kvalitu výuky. Zároveň se však jedná o nástroj využitelný i zkušenými pedagogy, pro které může být *EDUBO* efektivní pomůckou při tvorbě i revizi výukových materiálů.

Aplikace byla vyvíjena ve spolupráci s Pedagogickou fakultou Univerzity Karlovy, jejíž zástupci se podíleli na návrhu výukových aktivit, přípravě metodických návodů a průběžném testování nástroje. Díky jejich zpětné vazbě bylo možné aplikaci přizpůsobovat skutečným potřebám pedagogů.

Vývoj byl součástí několika výzkumných záměrů, mimo jiné v rámci projektů *Vývoj specifického SW nástroje pro plánování výuky* (DataPLEX Consulting, s.r.o.), *Inovativní software pro rozvoj moderní výuky na základních a středních školách* (Databig s.r.o.) a *Inovativní SW nástroj pro podporu moderní výuky* (Mr. Cloud s.r.o.).

Cílem závěrečné práce je především popsat metodiku vývoje tohoto nástroje, včetně technického návrhu, spolupráce s pedagogy a ověření funkčnosti aplikace v praxi. Hlavním výstupem je funkční aplikace *EDUBO*, která byla testována uživateli a jejímž prostřednictvím vzniklo přes 200 výuko-

vých plánů vytvořených pedagogy z Pedagogické fakulty Univerzity Karlovy a přes 250 výuko-vých plánů vytvořených studenty z Pedagogické fakulty Univerzity Jana Evangelisty Purkyně.

## **2. Přehled současného stavu problematiky a existujících aplikací**

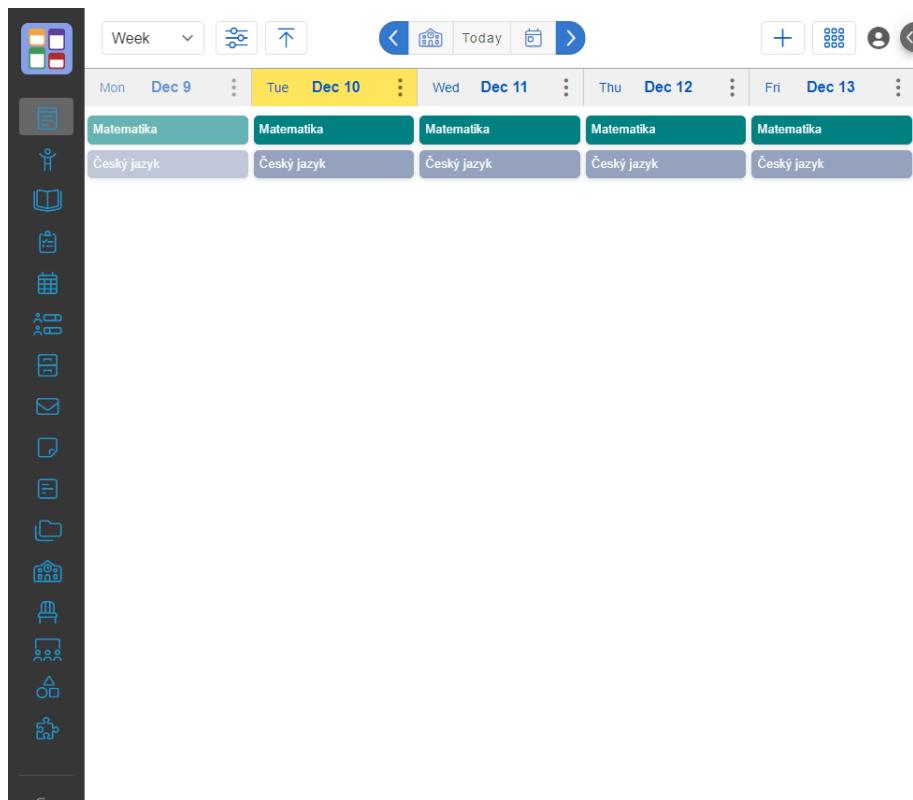
Plánování výuky je klíčovou činností učitelů, která ovlivňuje efektivitu vzdělávacího procesu. V digitální éře se pro tento účel stále více využívají specializované softwarové nástroje, které zjednodušují tvorbu a sdílení výukových plánů, správu tříd nebo interakci se studenty. Tyto aplikace učitelům umožňují šetřit čas, lépe organizovat výuku a poskytovat studentům kvalitnější vzdělávací materiály. Různé nástroje nabízejí širokou škálu funkcí, od jednoduchých textových editorů až po školské informační systémy s integrovaným systémem pro plánování výuky a interaktivními síťovými možnostmi.

Tato kapitola se zaměřuje na přehled čtyř významných aplikací pro plánování výuky a jejich srovnání s projektem *EDUBO*. Nejvíce se *EDUBO* přibližují aplikace *Planbook* a *Common Curriculum*, které nabízejí rozšířené možnosti správy výukových plánů. Dále budou zmíněny jednodušší aplikace *Lesson Planner PH* a *Slides Go AI Lesson Plan Generator*, které mají limitovanou funkčnost a odlišnou cílovou skupinu. Pro každou aplikaci bude popsána její funkcionalita, výhody, nevýhody a rozdíly oproti *EDUBO*.

### **2.1. Planbook**

*Planbook* je nástroj zaměřený na digitální plánování výuky, který umožňuje učitelům vytvářet a spravovat plány hodin, domácích úkolů a událostí v rámci jednotlivých předmětů. Aplikace podporuje široké spektrum funkcí, jako je správa docházky, evidence známk a vytváření událostí. *Planbook* je dostupný jako webová i mobilní aplikace, což usnadňuje její používání na různých zařízeních. Učitelé mohou plány sdílet přímo se studenty prostřednictvím účtů, což zajišťuje jejich snadnou dostupnost. Aplikace je však lokalizovaná pouze do angličtiny, což může být překážkou pro uživatele z jiných jazykových oblastí.

## 2. Přehled současného stavu problematiky a existujících aplikací



Obrázek 2.1.: Okno aplikace Planbook s náhledem na kalendář.

A screenshot of the Planbook application's lesson editor for Tuesday, December 10, at 12:00 PM. The title of the lesson is 'Počítání v oboru 1-10'. The editor interface includes fields for 'Title', 'Lesson Time' (set to 12:00 AM to 1:00 PM), and 'Unit'. Below these, there are tabs for 'LESSON', 'HOMEWORK', 'NOTES', 'STANDARDS', and 'ATTACHMENTS'. The 'LESSON' tab is active and contains a rich text editor toolbar. The 'Cíl hodiny:' section describes the goal of the lesson: 'Žáci se naučí sčítat jednoduché příklady v oboru čísel 1–10 pomocí praktických aktivit a vizuálních pomůcek.' The 'Struktura hodiny:' section details the lesson structure:

- 1. Úvod (5 minut):**
  - Učitel na tabuli napiše čísla od 1 do 10. Společně s dětmi je zopakuje a zeptá se: „Kdo mi řekne, co je nejmenší číslo? A co největší?“
  - Motivační otázka:  
„Když mám 3 jablka a dostanu ještě 2, kolik budu mít celkem?“
- 2. Hlavní část (30 minut):**
  - Praktické cvičení (10 minut):**  
Učitel rozdá kartičky s čísly a předměty (např. barevné kostičky).  
**Úkol:** Slož příklad (např. 3 + 2) a ukaž správný výsledek na kostičkách.
  - Práce ve dvojicích (10 minut):**  
Děti si vzájemně zadávají příklady na sčítání, které si ověřují pomocí pomůcek.
  - Hra „Počítací závod“ (10 minut):**  
Učitel předčítá jednoduché příklady (např. 4 + 3), děti musí co nejrychleji říct správnou odpověď.

Obrázek 2.2.: Okno aplikace Planbook s náhledem na editor výukového plánu.

**Hlavní funkce Planbook:**

- více předmětů – učitelé mohou spravovat plány výuky pro různorodé předměty,
- sdílení hodin – plány mohou být sdíleny se studenty prostřednictvím jejich účtů,
- docházka a známky – aplikace umožňuje evidenci docházky a známek, což přidává další vrstvu organizace,
- WYSIWYG (What You See Is What You Get, intuitivní způsob editace textu) editor – pro tvorbu výukových hodin nabízí jednoduchý editor s možností přidávání souborů a poznámek.

*Planbook* nabízí 90denní zkušební dobu. Následně stojí roční předplatné pro jednotlivce 20 dolarů, zatímco školní licence se pohybují mezi 14 až 18 dolary za uživatele.

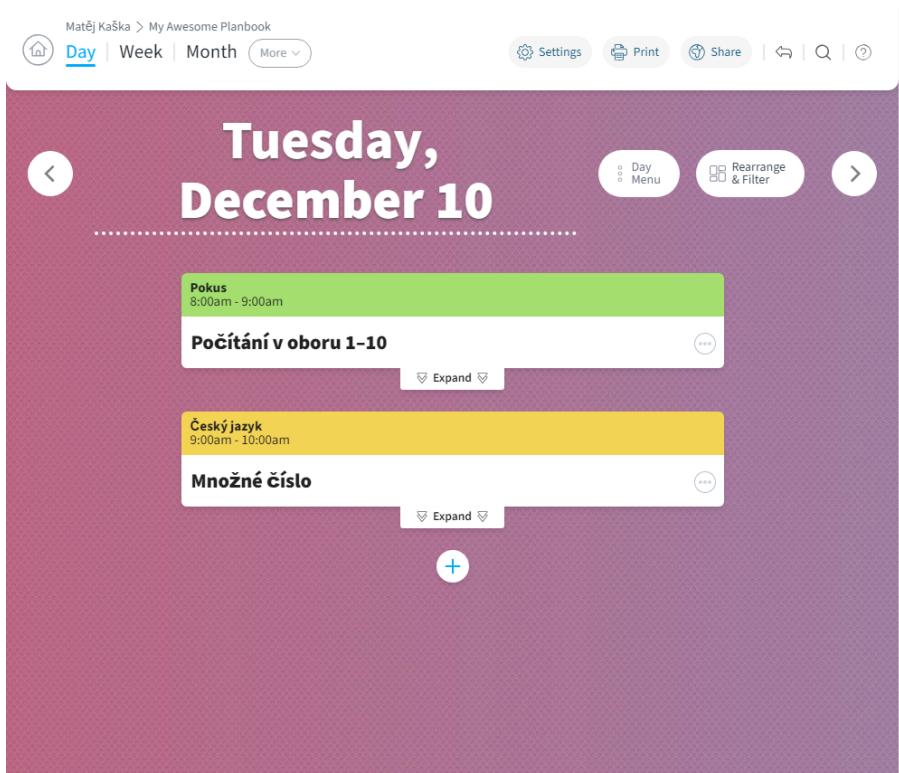
**Porovnání s EDUBO**

*Planbook* i *EDUBO* se zaměřují na digitální plánování výuky, ale jejich přístupy a funkce se liší. Na rozdíl od *Planbooku* nabízí *EDUBO* lokalizaci do češtiny a robustnější editor plánů hodin, který umožňuje detailní strukturování výuky. Zatímco *Planbook* obsahuje pokročilé funkce, jako je docházka a známky, *EDUBO* tyto funkce neřeší, protože se soustředí na samotné plánování a organizaci výuky. Významný rozdíl je také v přístupu k ceně – *EDUBO* je zcela zdarma, zatímco *Planbook* vyžaduje předplatné. Pokud jde o spolupráci mezi učiteli, *Planbook* ji nabízí v placené verzi, kde učitelé mohou sdílet své plány. *EDUBO* však tuto možnost rozšiřuje přes webový agregátor *Eduklub – plány výuky*, který je plně zdarma a podporuje otevřenou spolupráci napříč uživateli.

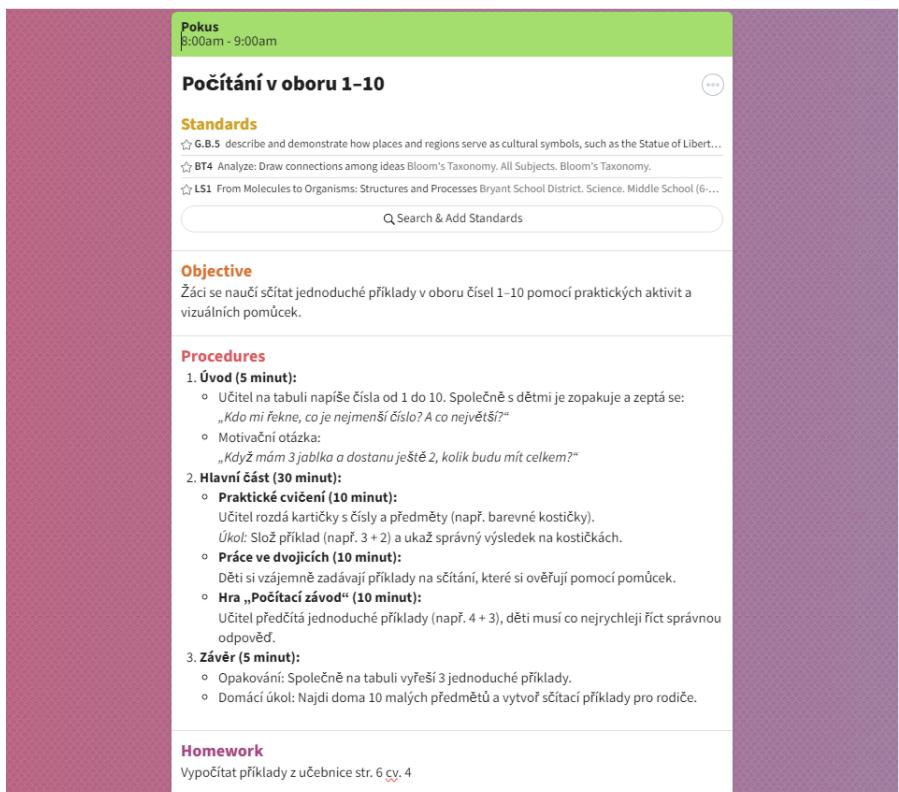
## 2.2. Common Curriculum

*Common Curriculum* je další aplikace zaměřená na plánování výuky, která je populární zejména mezi učiteli ve Spojených státech amerických. Její hlavní předností je integrace amerických vzdělávacích standardů, což umožňuje snadné plánování výuky v souladu s těmito osnovami. Uživatelé mohou vytvářet plány pro různé předměty a sdílet je buď prostřednictvím odkazu nebo ve formátu PDF. Aplikace také umožňuje vytvořit třídní stránku, kde jsou zobrazeny rozvrhy spolu s přehledy výukových plánů.

## 2. Přehled současného stavu problematiky a existujících aplikací



Obrázek 2.3.: Okno aplikace Common Curriculum s náhledem na kalendář.



Obrázek 2.4.: Okno aplikace Common Curriculum s náhledem na editor výukového plánu.

### Hlavní funkce Common Curriculum:

- více předmětů – učitelé mohou spravovat plány pro předměty a organizovat je podle tříd,
- integrace amerických vzdělávacích standardů – uživatelé mohou využívat předpřipravené kurikulární plány,
- možnosti sdílení – plány lze sdílet pomocí odkazu nebo exportovat do PDF,
- třídní stránka – učitelé mohou vytvořit přehled rozvrhů pro žáky a rodiče,
- WYSIWYG editor – jednoduchý editor s možností přidávat přílohy.

Aplikace nabízí bezplatnou verzi s omezenými funkcemi, zatímco plná verze je dostupná za 8 dolarů měsíčně. Školní licence se odvíjejí od počtu učitelů.

### Porovnání s EDUBO

*Common Curriculum* a *EDUBO* sdílejí podobnou vizi v oblasti plánování výuky, ale jejich funkcionality a přístup k uživatelům se značně liší. *Common Curriculum* klade důraz na americký vzdělávací systém, což je pro české prostředí nerelevantní. *EDUBO* se naopak zaměřuje na flexibilní tvorbu plánů přizpůsobených českému školství. Také editor je výrazně odlišný – zatímco *Common Curriculum* nabízí pouze textový editor, *EDUBO* umožňuje rozdělit hodinu do aktivit, přidávat cíle a poznámky. V oblasti spolupráce je *EDUBO* výhodnější díky bezplatnému agregátoru plánů. Výhodu má *EDUBO* rovněž v ceně, jelikož je zcela zdarma.

## 2.3. Lesson Planner PH

*Lesson Planner PH* je jednoduchá aplikace, která využívá umělou inteligenci k automatickému generování plánů výuky a testů. Podporuje pouze angličtinu a filipínštinu a nenabízí žádné další funkcionality kromě základního generování obsahu. Uživatelé mají měsíčně k dispozici 30 bezplatných generování, přičemž další lze dokoupit prostřednictvím kreditového systému (1 generace za přibližně 2 Kč).

Tato aplikace reflektuje současný trend využití umělé inteligence, který přináší jednoduchá řešení pro tvorbu výukových materiálů. Nicméně, vzhledem k absenci dalších funkcí, jako je strukturované plánování nebo správa tříd, není *Lesson Planner PH* příliš vhodný pro učitele, kteří potřebují propracovanější nástroj. *EDUBO* naopak poskytuje širší škálu možností včetně propracovaného editoru a lokalizace, což *Lesson Planner PH* zcela postrádá. Na druhou stranu však *EDUBO* zatím postrádá integraci generativní umělé inteligence, která je jedním z hlavních trendů současných vzdělávacích aplikací. Ačkoliv bylo v plánu obohatit *EDUBO* o tyto funkce pro automatické generování výukových plánů a materiálů, tento rozvojový krok nebyl realizován kvůli nedostatečné podpoře projektu *EDUZA*, jenž měl být zaměřen právě na tento typ funkcionality.

## 2.4. Slides Go AI Lesson Plan Generator

*Slides Go AI Lesson Plan Generator* je nástroj zaměřený na rychlé generování plánů výuky pomocí umělé inteligence. Generované plány lze stáhnout ve formátu PDF, ale aplikace nepodporuje český jazyk. *Slides Go* je zdarma a svou jednoduchostí cílí na učitele hledající rychlá a snadná řešení.

Podobně jako *Lesson Planner PH* odráží tento nástroj rostoucí trend využívání AI v oblasti vzdělávání, ale kvůli omezené funkčnosti nenabízí žádnou skutečnou přidanou hodnotu pro komplexní plánování výuky. *EDUBO* vyniká v této oblasti tím, že umožňuje učitelům nejen detailní plánování, ale také jejich přizpůsobení a sdílení s kolegy a studenty, což *Slides Go* zcela postrádá.

Tyto dvě aplikace ukazují, že současný trend umělé inteligence přináší řadu jednoduchých nástrojů pro tvorbu výukových materiálů. Nicméně jejich omezená funkčnost a absence pokročilých funkcí je činí spíše doplňkem než plnohodnotným řešením pro plánování výuky, jakým je *EDUBO*.

# 3. Teoretická východiska

## 3.1. Výukový proces

Porozumění výukovému procesu je pro každého vyučujícího klíčové, protože právě skrze něj se uskutečňuje efektivní přenos znalostí a dovedností. Výukový proces není pouze o předávání informací, ale o vědomém a promyšleném řízení výuky tak, aby došlo k porozumění, zapamatování a aplikaci učiva. Pro vyučujícího to znamená nejen znalost látky, ale i schopnost zvolit vhodné metody, nástroje a strategie, které podpoří proces učení u každého jednotlivého žáka. Studium výukového procesu tak poskytuje rámec, jak výuku strukturovat, reflektovat a postupně zlepšovat.

Jedním z často citovaných modelů, který pomáhá pochopit strukturu efektivního vyučování, je Gagného devíti krokový model (*Nine Events of Instruction*). Americký psycholog Robert Mills Gagné, A.B., Sc.M., Ph.D., vytvořil tuto strukturu jako vodítko pro učitele k postupnému vedení výuky, která podporuje učení na úrovni kognitivních procesů. Jeho devět kroků lze shrnout následovně:

1. **Získání pozornosti** (*Gaining attention*) – Například pomocí otázky, překvapivého faktu nebo vizuálního stimulu.
2. **Informování o cílech výuky** (*Informing learners of objectives*) – Učící se by měl vědět, co se má naučit.
3. **Navázání na předchozí znalosti** (*Stimulating recall of prior learning*) – Aktivace dřívějších znalostí a zkušeností.
4. **Prezentace obsahu** (*Presenting the stimulus*) – Podání nové látky vhodnou formou.
5. **Poskytnutí vedení** (*Provide learning guidance*) – Poskytnutí vodítek, která pomáhají učíćímu se propojit nové informace do smysluplného celku.
6. **Vybídnutí k praxi** (*Eliciting performance*) – Možnost procvičení nebo vyzkoušení nového učiva.
7. **Poskytnutí zpětné vazby** (*Providing feedback*) – Okamžitá a konkrétní zpětná vazba učíćímu se.
8. **Zhodnocení výstupu** (*Assessing performance*) – Ověření, zda se učící se učivo naučil, například formou testu, diskuse nebo praktického úkolu.
9. **Podpora zapamatování a přenosu nově osvojených znalostí** (*Enhancing retention and transfer*) – Pomoc učíćím se, aby si nové poznatky zapamatovali a uměli je aplikovat v praxi.

Tento model je cenný zejména svou přehledností a univerzálností – lze jej aplikovat jak při přípravě běžné vyučovací hodiny, tak při tvorbě digitálních výukových materiálů.

Pohled na výukový proces rozšířil také Pavel Beránek, který jej ve své přednášce popsal jako interakci dvou subjektů – vyučujícího a učícího se. Výuková látka je vyučujícím nejprve didakticky transformována, tedy přetvořena do podoby, která je vhodná pro předání, a následně předávána učícímu se. Ten ji zpracovává (asimiluje) svým vlastním tempem a způsobem. Cílem práce vyučujícího je optimalizovat celý tento proces tak, aby bylo učení co nejfektivnější.

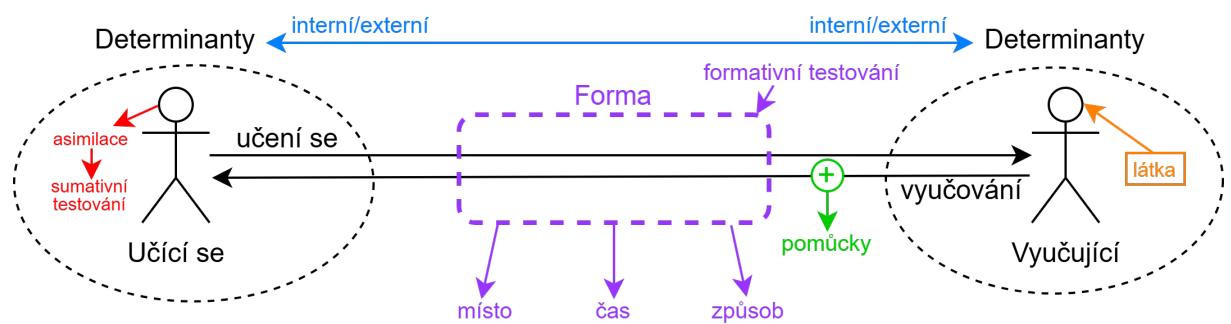
Výuka však vždy probíhá v určité formě, která ji výrazně ovlivňuje. Tato forma zahrnuje například místo výuky, časové zařazení (dopoledne nebo odpoledne) a také způsob výkladu, tedy zda vyučující látku pouze přednáší, vede diskusi nebo nechává učící se pracovat samostatně. Výběr těchto parametrů má zásadní vliv na výkon učících se – například náročné učivo zařazené na konec dne celodenní výuky může být méně efektivní než stejně učivo vyučované na začátku dne.

Výukový proces je zároveň ovlivněn mnoha determinanty, které lze rozdělit na interní a externí. Interní determinanty zahrnují například mentální schopnosti učících se, úroveň porozumění, motivaci či případné specifické poruchy učení. Tyto faktory mohou být částečně ovlivnitelné, avšak mnohdy je nutné je zohlednit při výběru přístupů a metod. Mezi externí determinanty spadá prostředí třídy, hladina hluku, světelné podmínky, čas výuky nebo dostupné pomůcky.

Podstatným prvkem efektivní výuky je také volba vhodných nástrojů a pomůcek. Ty mohou být jak tradiční (tabule, učebnice, abakus), tak moderní (interaktivní tabule, výukové aplikace, experimentální sady). Správně zvolené pomůcky usnadňují pochopení látky a mohou posílit motivaci učících se.

Didaktické strukturování hodiny do výukových aktivit umožňuje lépe řídit průběh výuky a přizpůsobit ji rytmu pozornosti učících se. Délka jednotlivých aktivit, jejich náplň a návaznost mají vliv na to, jak učící se vnímají a zpracovávají učivo. Dobře připravený plán s různorodými aktivitami zvyšuje efektivitu výuky a umožňuje vyučujícímu pružně reagovat na potřeby třídy.

Závěrem je nutné zmínit také testování asimilace, tedy ověření toho, zda se učící se skutečně naučili to, co bylo cílem výuky. Rozlišujeme sumativní testování, které poskytuje celkový přehled o dosažených výsledcích (např. test na konci kapitoly), a formativní hodnocení, které probíhá průběžně a slouží jako zpětná vazba pro vyučujícího i učícího se. Může mít podobu diskuse, otázek během výuky, nebo jednoduchého dotazu při individuálním kontaktu. Tento přístup umožňuje okamžitou korekci a podporuje aktivní zapojení učících se do procesu učení.



Obrázek 3.1.: Diagram znázorňuje výukový proces navržený Pavlem Beránkem.

## 3.2. Výukové aktivity

Výukové aktivity tvoří základní nástroje, pomocí kterých učitel strukturuje a řídí proces učení. Nejde pouze o plnění úkolů, ale o promyšlené činnosti, které naplňují konkrétní výukové cíle, zapojují žáky a pomáhají budovat hlubší porozumění. Jejich správný výběr a zařazení do hodinové struktury přispívá k efektivnějšímu učení. Mezi klíčové typy aktivit patří např. evokace, objevování, procvičování, výklad, shrnutí, hodnocení, práce se zdroji či zpětná vazba.

### Evokace

Evokace představuje úvodní fázi výukové hodiny a její hlavní funkcí je připravit žáky na výuku. Správně zvládnutý začátek hodiny je předpokladem pro úspěšné učení. Učitel v této fázi pracuje s pozorností, motivací, cíli a navázáním na předchozí znalosti žáků. Účelem je zajistit, aby žáci věděli, co se budou učit, proč je to důležité, a zároveň se cítili připraveni na nové učivo.

### Aktivace předchozích znalostí

Jednou z klíčových aktivit v rámci evokace je aktivace vstupních znalostí, kdy učitel zjišťuje, co žáci o tématu už vědí. Může využít brainstorming, otázky, digitální nástroje (např. Mentimeter) nebo krátké úlohy. Výsledky se doporučuje vizualizovat, aby s nimi bylo možné dále pracovat.

### Stanovení cílů

Další důležitou aktivitou v úvodu hodiny je sdělení výukového cíle. Dobře formulovaný cíl by měl být konkrétní, ověřitelný a formulovaný z pohledu žáka – např. „žák na konci hodiny rozliší ...“. Formulace cílů může vycházet z Bloomovy taxonomie nebo její revidované verze, která rozlišuje dimenzi znalostí a kognitivních procesů. Stanovení cíle dává výuce směr a umožňuje učiteli i žákovi reflektovat pokrok.

## Hodnocení

Hodnocení je jednou ze součástí vzdělávacího procesu. Jeho cílem je poskytnout žákům zpětnou vazbu a podporu pro jejich další učení a rozvoj. Mělo by být konstruktivní, pozitivní a objektivní. Hodnocení neslouží jen ke klasifikaci, ale i k identifikaci oblastí, na kterých je třeba dále pracovat. Důležité je vytvořit podpůrné a bezpečné prostředí, kde stres z výsledků není překážkou v učení. Hodnocení tak může být nástrojem rozvoje, ne trestu.

### Kontrolní test

Kontrolní test může mít písemnou nebo praktickou podobu. Slouží k ověření znalostí a dovedností v rámci aktuálně probírané oblasti. V praktické podobě může žák například něco vytvořit, opravit či navrhnout. Testy mohou zahrnovat různé typy otázek – otevřené, uzavřené, shody nebo doplňování vět.

### Souhrnný test

Souhrnný test ověřuje zvládnutí učiva za delší časové období, např. na konci tematického celku, čtvrtletí nebo pololetí. Mívá širší záběr než běžné testy a může mít písemnou nebo praktickou formu. Záměrem není jen ověřit paměťové zvládnutí faktů, ale také schopnost propojit klíčové koncepty a dovednosti.

### Ústní zkoušení

Ústní zkoušení může být pro některé žáky stresující. Je důležité žáky na jeho průběh připravit, poskytnout jim jasné informace o jeho cíli a průběhu. Důraz by měl být kladen na porozumění, nikoli na formu. Alternativou může být projekt, prezentace nebo týmová práce.

### On-line testování

On-line testování poskytuje okamžitou zpětnou vazbu, umožňuje personalizaci testu a efektivní sledování pokroku žáků. Díky automatickému hodnocení je časově efektivní a umožňuje rychlé vyhodnocení odpovědí. Test může být proveden na různých zařízeních – mobilních telefonech, tabletech i počítačích.

### Skupinové testování

Skupinové testování je forma hodnocení, která rozvíjí nejen znalosti a dovednosti, ale také schopnost komunikace a spolupráce. Test není soutěž, ale příležitost pro vzájemné učení. Žáci by měli být předem seznámeni s hodnoticími kritérii. Správně vedené skupinové testování pomáhá posilovat sociální dovednosti i kritické myšlení.

## Objevování

Objevování neboli „umění objevovat“ označuje metodologický přístup k poznání založený na uvědomění si existence určitých pravidelností, které dosud nebyly ověřeny. Žáci nejsou pasivními příjemci informací, ale sami přicházejí na nové poznatky prostřednictvím vlastního bádání, srovnávání, kladení otázek a vytváření hypotéz. Didaktickým cílem objevování je zapojit žáky do procesu samostatného řešení problémů a podpořit jejich tvůrčí myšlení.

Objevování může mít podobu heuristického rozhovoru nebo výkladu, přičemž žáci pracují s otázkami, modely, schématy či analogiemi. Výuka založená na heuristických funkcích stimuluje žáky k aktivnímu učení, hlubšímu porozumění a integraci znalostí. Učitel může zvolit „malou“ nebo „velkou“ heuristickou metodu – podle míry samostatnosti žáka při zpracování učiva.

## Práce se zdroji, příklady a situacemi

Žáci se učí využívat dostupné informační zdroje, pracují s texty, obrázky, animacemi nebo simulacními programy. Výuka může být zaměřena na vyhledávání informací, jejich prezentaci, práci s výukovým softwarem nebo tvorbu vlastních výstupů. Učitel by měl nabídnout vhodné okruhy nebo odkazy a důsledně promyslet činnosti žáků po stránce metodické i obsahové.

## Pracovní listy

Pracovní listy vedou žáky k aktivnímu zkoumání a kritickému myšlení. Mohou obsahovat otázky, úkoly nebo úlohy k analýze konkrétního problému. Pracovní listy lze využít při individuální, párové i skupinové práci.

## Bádání

Badatelsky orientované učení podporuje koncepční porozumění a proces osvojování znalostí skrze vlastní aktivitu žáka. Žáci formulují problémy, navrhují způsoby řešení, analyzují a ověřují data. Podle míry vedení učitelem lze rozlišit čtyři úrovně bádání – potvrzující, strukturované, nasměrované a otevřené.

## Experiment

Experiment slouží k ověřování hypotéz a poznání za řízených podmínek. Může mít formu kvalitativní, kvantitativní nebo myšlenkovou. Během experimentu žáci pozorují, měří, analyzují data a vyvozují závěry. Významnou roli hraje i plánování a zpracování výsledků.

## On-line experiment

On-line experimenty zpřístupňují vzdálená měření nebo přístroje, které by jinak byly žákům nedostupné. Žáci mohou získávat data z reálných měření například prostřednictvím meteorologických stanic nebo vzdálených laboratoří.

## Simulace

Simulace představují využití virtuálního prostředí k modelování jevů, které jsou nebezpečné, obtížně přístupné nebo abstraktní. Ve výuce se využívají například simulace měření, modelování molekul nebo virtuální laboratoře. Tyto nástroje pomáhají vizualizovat procesy a zprostředkovat poznání jinou formou než tradiční experiment.

## Zdroje

Práce se zdroji představuje důležitou výukovou aktivitu, která rozvíjí kritické myšlení, informační gramotnost a schopnost samostatného učení. Žáci se neučí pouze pasivně přijímat předložené informace, ale také sami vyhledávat relevantní a důvěryhodné zdroje, posuzovat jejich kvalitu a ověřovat fakta. Učitelé je přitom vedou k rozpoznání spolehlivosti informací, zohlednění různých pohledů na téma a etickému zacházení s informacemi. Klíčem je podpora samostatnosti a schopnosti žáků aktivně se zapojit do vzdělávacího procesu prostřednictvím informací z různých typů zdrojů.

## Sběr dat

Sběr dat je aktivitou, při které žáci sami získávají informace prostřednictvím vlastních měření, dotazníků nebo pozorování. Tento způsob práce vede k praktické zkušenosti, rozvoji kritického myšlení i zapojení do místně zakotveného učení. Žáci se učí třídit a interpretovat data, která mají pro ně osobní význam, a zároveň mohou rozvíjet i komunikační dovednosti.

## On-line zdroje

On-line prostředí nabízí široké spektrum informačních materiálů – od odborných databází po běžně navštěvované portály. Práce s těmito zdroji rozvíjí nejen digitální kompetence, ale především schopnost hodnotit důvěryhodnost, relevanci a účel obsahu. Učitelé vedou žáky k tomu, aby při vyhledávání informací zvažovali nejen co, ale i proč a od koho čerpají.

## Práce s literaturou

Tištěné materiály, jako slovníky, tabulky, grafy nebo mapy, umožňují žákům porozumět obsahu do hloubky a trénovat dovednosti spojené s vyhledáváním a interpretací dat. Práce s knihou podporuje i dovednosti potřebné pro akademickou přípravu.

## Počítačové simulace ve výuce

Simulace umožňují žákům interaktivní a bezpečné zkoumání složitých jevů – například vizualizací pohybu, chemických reakcí nebo sociálních interakcí. Zajišťují okamžitou zpětnou vazbu a podporují aktivní učení, analýzu výsledků a kreativitu. Jejich dostupnost a rozmanitost umožňuje efektivní využití napříč obory a stupni vzdělávání.

## Procvičování

Procvičování je základním stavebním kamenem efektivního vzdělávání. Umožňuje žákům upevnit znalosti, přenést je do dlouhodobé paměti a osvojit si automatizované dovednosti. Klíčem je pravidelnost, rozmanitost a cílené opakování. Kombinace různých typů úloh, praktické aplikace a poskytnutí zpětné vazby podporují hlubší porozumění a motivují k dalšímu učení. Významnou roli zde sehrává i technologie, která umožňuje individualizaci výuky a okamžitou odezvu na výkon žáků.

## Receptivní a produktivní jazykové dovednosti

Procvičování zahrnuje aktivity zaměřené na všechny složky jazykového projevu. Čtení a poslech rozvíjejí schopnost porozumět tištěným i audiovizuálním materiálům. Mluvení a psaní vedou k aktivní produkci, kde žáci uplatňují získané poznatky ve vlastním projevu. Všechny tyto dovednosti se uplatňují v komunikaci, často i formou interakce mezi spolužáky, která přirozeně propojuje jazyk s reálnými situacemi.

## Procvičování s využitím ICT

Digitální nástroje jako Google Forms, Kahoot! nebo Quizlet umožňují rychlé a variabilní procvičování s okamžitou zpětnou vazbou. Podporují motivaci žáků, ulehčují sledování pokroku a umožňují efektivní diferenciaci výuky.

## Mediace

Mediace představuje schopnost zprostředkovat informaci dalším osobám – shrnout, přeložit, vysvětlit či přetvořit původní sdělení tak, aby bylo srozumitelné. Umožňuje žákům propojit porozumění a produkci, rozvíjí samostatnost a schopnost přizpůsobit jazyk různým situacím a adresátům.

Mediace se uplatňuje napříč předměty i v běžných komunikačních situacích. Učitel může zařadit aktivity, při kterých žáci např. převyprávějí text spolužákovi, vytvoří shrnutí videa, přeloží návod z cizího jazyka nebo vysvětlí spolužákovi úlohu jinými slovy. Klíčové je, že cílem není reprodukce obsahu, ale jeho přeformulování a přizpůsobení konkrétnímu posluchači nebo čtenáři.

Mediace podporuje nejen jazykový rozvoj, ale také kritické myšlení, empatii a vnímání různých perspektiv. Aktivní začlenění tohoto přístupu do výuky vede žáky k větší odpovědnosti za porozumění obsahu a za komunikaci s ostatními.

#### **Manipulativní činnosti**

Práce s konkrétními objekty pomáhá rozvíjet jemnou motoriku, prostorové vnímání, ale i schopnost spolupráce a řešení problémů. Rýsování, stavění modelů nebo praktická měření přinášejí aktivní zapojení žáků a podporují jejich učení prostřednictvím více smyslů.

Manipulativní činnosti umožňují lepší pochopení abstraktních pojmu, protože propojují myšlení s fyzickou činností. Uplatnění nachází především ve výuce matematiky, přírodovědných předmětů, ale i například v jazykovém vzdělávání formou třídění kartiček nebo při tvorbě příběhů. Učitel by měl volit pomůcky přiměřené věku a kognitivní úrovni žáků.

#### **Herní aktivity**

Hra vytváří přirozené a motivující prostředí pro učení. Využívá soutěživost, ale i spolupráci. Jazyk a poznatky jsou aplikovány nenásilně, což napomáhá jejich osvojení. Didaktické hry umožňují procvičování i evaluaci bez stresu z formálního hodnocení.

Herní aktivity mohou mít podobu deskových, digitálních i pohybových her. Mohou být použity k motivaci na začátku hodiny, k procvičování učiva nebo k opakování na konci tematického celku. Kromě kognitivních dovedností rozvíjejí i sociální interakce, kreativitu a řešení problémů. Důležité je, aby pravidla hry byla jasná a spravedlivá a aby reflektovala vzdělávací cíle.

#### **Shrnutí**

Závěrečné shrnutí je důležitou součástí hodiny. Nejde jen o zopakování, co se v hodině dělo, ale hlavně o zdůraznění podstatného a propojení na další téma. Shrnutí pomáhá utvrdit učivo, dává žákům přehled, kontroluje porozumění a podporuje aktivní učení. Má také pozitivní vliv na motivaci a přípravu na domácí úkoly nebo zkoušení.

#### **Had**

Had je jednoduchá, ale efektivní metoda shrnutí učiva, při které každý žák stručně sdělí, co si z hodiny odnáší. Postupuje se po třídě, jeden po druhém – vzniká tak pomyslný „had“. Vyučující může žáky omezit na jedno či dvě slova, případně zadání ztížit tím, že každý musí říct něco, co ještě nezaznělo. Aktivita posiluje soustředění, sebereflexi a dává učiteli rychlou zpětnou vazbu o tom, co žáci vnímají jako důležité.

## Propustky

Propustky jsou shrnovací technika, při které žáci na konci hodiny vyplní malý lístek s odpovědí na zadanou otázku – např. „Co si z hodiny odnášíš?“, případně „Čemu jsi nerozuměl?“. Mohou být anonymní, což podporuje upřímnost a pocit bezpečí. Učitel získává rychlý přehled o pochopení učiva i tématech, ke kterým je potřeba se vrátit. Propustky jsou nenáročné na přípravu a dají se snadno začlenit do jakékoli výuky.

## Myšlenkové mapy

Myšlenkové mapy slouží jako vizuální nástroj pro shrnutí učiva, který pomáhá zapamatování, třídění informací a rozvíjí kreativitu. Lze je využít při prezentacích, brainstormingu, plánování, řešení problémů i alternativních zápisích.

## Otázky

Otázky jsou klíčovým nástrojem zpětné vazby. Pomáhají učiteli zjistit míru porozumění a žákům třídit myšlenky. Měly by být promyšlené, navázané na výukové cíle a přizpůsobené úrovni žáků. Kromě faktických dotazů je vhodné používat otázky vyšší kognitivní úrovni (např. podle Bloomovy taxonomie), otevřené otázky nebo techniky jako *Think – Pair – Share (Mysli – Diskutuj ve dvojici – Sdílej)*.

## Výklad

Výklad je metoda, při které učitel slovně předává nové informace a vysvětluje význam učiva. Patří mezi nejběžnější formy výuky, umožňuje rychle a efektivně seznámit žáky s novým tématem. Aby byl výklad co nejúčinnější, měl by být srozumitelný, strukturovaný, doplněný vizuální oporou a proložený aktivizačními prvky.

## Struktura učiva

Seznámení žáků s cíli a strukturou učiva na začátku tématu podporuje jejich motivaci, orientaci a pochopení. Jasný rámec výuky pomáhá lépe plánovat další aktivity a rozvíjí i kritické myšlení žáků.

## Vizualizace

Teorie duálního kódování vysvětluje, proč si žáci lépe zapamatují informaci, pokud ji slyší a zároveň vidí. Mozek totiž zpracovává informace pomocí dvou oddělených kanálů: verbálního (slova) a vizuálního (obrazy). Když tyto kanály propojujeme, snižujeme zatížení pracovní paměti a zvyšujeme šanci na hlubší učení, zejména u dětí starších sedmi let. Výzkumy ukazují, že např. při

učení slovíček pomáhá jejich propojení s odpovídajícími obrázky. V hodinách tak dává smysl kombinovat výklad s vhodnými vizuálními prvky.

#### **Praktické příklady**

Použití konkrétních příkladů, modelů nebo běžných životních situací přibližuje učivo realitě a podporuje hlubší porozumění. Žáci si lépe zapamatují to, co si dokáží představit nebo si sami vyzkoušet.

#### **Modely**

Modely chápeme jako názorné učební pomůcky, které pomáhají konkretizovat abstraktní pojmy a objasnit vztahy nebo procesy. Měly by být názorné, věrné a přiměřeně zjednodušené. Důležité je, aby žáci chápali rozdíl mezi modelem a skutečností.

#### **Zpětná vazba**

Zpětná vazba je klíčovou součástí formativního hodnocení. Než ji poskytneme, musíme mít jasné cíle a kritéria hodnocení. Efektivní zpětná vazba popisuje, co se žákovi podařilo, a místo přímé kritiky klade otázky, které vedou k zamýšlení. Správně podaná zpětná vazba by měla žáka motivovat a vést ke zlepšení výkonu, nikoliv ho demotivovat nebo hodnotit jako osobnost.

#### **Signál**

Signál je jednoduchá technika pro okamžité zjištění, zda žáci porozuměli učivu. Pomocí palců, symbolů nebo kartiček dávají žáci najevo, jestli učivo chápou, mají pochybnosti nebo nerozumí. Výhodou je rychlá vizuální zpětná vazba, kterou učitel získá od celé třídy najednou. Tato metoda posiluje participaci a učí žáky reflektovat své vlastní porozumění.

#### **Teplovýměná**

Teplovýměná slouží k vizuálnímu vyjádření pocitu porozumění nebo nálady žáků na škále. Umožňuje jim sledovat svůj pokrok nebo míru pochopení v čase. Pomocí procent nebo symbolů si žáci sami označují, jak se cítí nebo jak rozumí probíranému tématu. Tato metoda podporuje sebereflexi a umožňuje učiteli lépe přizpůsobit výuku.

#### **Škála**

Škála je nástroj, pomocí kterého mohou žáci vyjádřit, jak dobře se cítí ve vztahu k probírané látce nebo aktivitě. Často se využívá čtyř až desetistupňové hodnocení. Učitel získává subjektivní, ale užitečný pohled na vnímání výuky ze strany žáků. Škála se může týkat nejen porozumění, ale i emocí a spolupráce.

## Semafor

Semafor je oblíbená technika, při níž žáci pomocí tří barev vyjadřují úroveň porozumění. Zelená značí úplné porozumění, oranžová nejistotu a červená neporozumění. Učitel okamžitě získává přehled o stavu třídy a může reagovat. Žáci se zároveň učí hodnotit sami sebe a nebát se přiznat problém.

## Mazací tabulky

Mazací tabulky žáci využívají k zaznamenání krátkých odpovědí, výpočtů, náčrtků, grafů nebo jazykových jevů, které následně na pokyn učitele ukazují. Tím poskytuje rychlou zpětnou vazbu o své úrovni porozumění, aniž by museli vystupovat veřejně. Tato technika rozvíjí schopnost přesné formulace myšlenek a je snadno použitelná v různých předmětech.

## Losovátka a špachtlíčky

Losovátka přináší do výuky prvek náhody a spravedlnosti. Učitel si připraví např. dřevěné špachtle, na které napíše jména všech žáků, a vloží je do kelímku. Po položení otázky učitel jednu ze špachtlí náhodně vytáhne a ta určí, kdo bude odpovídat. Tímto způsobem se udržuje pozornost celé třídy a aktivizují se i ti, kteří se běžně nehlásí. Tento způsob zajišťuje spravedlnost a udržuje všechny ve středu.

## Think – Pair – Share

*Think – Pair – Share (Mysli – Diskutuj ve dvojici – Sdílej)* dává žákům čas na přemýšlení, diskuzi a následné sdílení. Učitel nejprve položí otázku, pak žáci diskutují ve dvojicích, a nakonec jeden z nich odpovídá. Tato metoda podporuje hlubší zamýšlení a spolupráci. Introverti mají šanci připravit si odpověď, extroverti se učí strukturovat své myšlenky.

## Barevné kelímky

Barevné kelímky pomáhají žákům vizuálně vyjádřit, v jaké fázi práce se právě nachází. Každý má na lavici sadu kelímků různých barev – např. bílý „pracuji“, zelený „mám hotovo a mohu pomoci“, žlutý „potřebuji pomoc od spolužáka“ a červený „nerozumím a potřebuji pomoc učitele“. Žák jednoduše postaví na lavici kelímek dané barvy podle své situace. Učitel tak získává okamžitý přehled o třídě a může efektivně reagovat. Zároveň se tím podporuje vzájemná spolupráce a vrstevnická výpomoc, která rozvíjí jak žáky, kteří pomoc poskytují, tak ty, kteří ji přijímají.

## Nedokončené věty

Nedokončené věty slouží žákům k tomu, aby na konci hodiny reflektovali svůj učební proces. Učitel jim nabídne začátek věty, jako např. „Zlepšil jsem se v …“ nebo „Nejvíce mě zaujalo…“,

a žáci ji dokončí podle svých zkušeností z hodiny. Tato technika rozvíjí schopnost sebereflexe, podporuje vědomé učení a pomáhá žákům lépe formulovat, co potřebují ke zlepšení. Výhodou je, že se aktivně zapojují i ti, kteří se v hodině běžně tak neprojevují.

#### Otevřené otázky

Otevřené otázky rozvíjejí kritické myšlení a samostatné vyjadřování žáků. Učitel se skrze ně dozvídá více než jen fakta – například i o emocích nebo způsobu práce žáka. Je třeba klást jasné formulované otázky bez náznaku správné odpovědi. Tato technika je vhodná pro hlubší porozumění a plánování další výuky.

#### Učební úloha

Učební úloha slouží k tomu, aby se žák něco naučil – na rozdíl od testové, která jen ověřuje. Žáci mohou pracovat samostatně, ve dvojicích nebo se zdroji, pokud se úloha zaměřuje na vyšší kognitivní náročnost. Učební úlohy často fungují jako diagnostický nástroj na konci hodiny.

#### Komplexní způsob hodnocení

Komplexní hodnocení propojuje formativní a sumativní přístupy a vedle známek zahrnuje i slovní zpětnou vazbu, reflexi a povzbuzení. Klade důraz na proces učení, nikoliv jen na jeho výsledek. Učitel spolu se žákem sleduje, kde se žák nachází, a společně hledají cesty ke zlepšení. Žáci se učí hodnotit nejen práci druhých, ale i sami sebe, což rozvíjí jejich metakognitivní dovednosti a zvyšuje zodpovědnost za vlastní učení. Smyslem tohoto přístupu je nejen zlepšit výsledky, ale vést žáky k hlubšímu porozumění a dlouhodobému růstu.

### 3.3. Životní cyklus softwaru

#### Charakteristika životního cyklu

Životní cyklus softwaru představuje chronologicky propojenou soustavu stavů, kterými software postupně prochází od počátečního návrhu až po finální údržbu. Tento cyklus zahrnuje všechny kroky nutné k vytvoření kvalitního produktu, který odpovídá potřebám uživatelů a zadavatelů. Životní cyklus je přizpůsobitelný různým typům projektů a metodikám, což z něj činí univerzální rámec vhodný pro malé i rozsáhlé projekty. Jeho správné řízení zajišťuje efektivitu vývoje a umožňuje reagovat na změny v průběhu celého projektu. Životní cyklus softwaru je řízen procesem vývoje softwaru.

Proces vývoje softwaru je systematický a strukturovaný přístup, jehož cílem je vytvořit spolehlivý a funkční produkt. Tento proces minimalizuje rizika a chyby, optimalizuje využití zdrojů a zajišťuje splnění jak funkčních, tak mimofunkčních požadavků. Každá fáze vývoje, od analýzy a návrhu

přes implementaci až po testování a nasazení, má jasně stanovené cíle a výstupy, které tvoří základ pro další kroky. Strukturovaný proces navíc podporuje týmovou spolupráci a usnadňuje komunikaci mezi členy týmu, vedením projektu a uživateli, čímž přispívá k hladkému průběhu vývoje.

## Fáze životního cyklu softwaru

Životní cyklus softwaru se skládá z několika základních fází, z nichž každá plní specifickou funkci a přispívá k dosažení cílového produktu. Jednotlivé fáze na sebe logicky navazují, přičemž jejich průběh může být lineární (např. vodopádový model) nebo iterativní (např. agilní přístupy).

### Analýza

Fáze analýzy je klíčovým počátečním krokem životního cyklu. Během této fáze se sbírají a analyzují požadavky uživatelů a stakeholderů (zainteresovaných stran). Hlavním cílem analýzy je pochopit, jaké funkce a vlastnosti má výsledný software mít, a vytvořit tak pevný základ pro následný návrh a vývoj.

V této fázi dochází k definování:

- funkčních požadavků (co má software dělat, např. možnost registrace uživatele),
- mimofunkčních požadavků (např. výkon, bezpečnost, škálovatelnost),
- základních cílů projektu (časový harmonogram, rozpočet, klíčové milníky).

V rámci analýzy požadavků se požadavky často rozdělují do tří kategorií:

- **normální požadavky (normal)** – standardní funkce, které uživatelé očekávají, např. možnost přihlášení,
- **očekávané požadavky (expected)** – základní vlastnosti, které musí software splňovat, jako je stabilita nebo bezpečnost,
- **vzrušující požadavky (exciting)** – nadstandardní funkce, které uživatele příjemně překvapí a odlišují software od konkurence, například personalizované doporučování obsahu.

Výstupem analýzy je dokumentace požadavků, často ve formě specifikace, uživatelských příběhů nebo případů užití, které slouží jako referenční bod pro další fáze.

### Návrh

Na základě výsledků analýzy se v této fázi vytváří technická architektura systému a plán jeho implementace. Návrh specifikuje, jak budou jednotlivé části softwaru spolupracovat a zohledňuje funkční i mimofunkční požadavky definované v předchozí fázi.

Mezi hlavní činnosti v této fázi patří:

- návrh softwarové architektury (např. výběr mezi monolitickou architekturou nebo architekturou mikroslužeb),
- datové modelování (např. návrh entitně-vztahového modelu pro databázi),
- vytvoření prototypů uživatelského rozhraní (např. drátěný model),
- definice API a způsobu integrace s dalšími systémy.

Výstupem této fáze jsou technické dokumenty, diagramy (např. UML diagramy), specifikace rozhraní a někdy i první prototypy, které umožňují ověřit správnost návrhu před zahájením vývoje.

## Vývoj

Fáze vývoje představuje proces implementace návrhu systému do podoby funkčního softwaru. V této fázi vývojáři píší kód podle specifikací vytvořených v předchozích fázích. Kromě programování probíhá také integrace jednotlivých komponent a jejich příprava pro testování.

Hlavní činnosti zahrnují:

- **Implementace funkcionalit.** Vývojáři vytvářejí kód pro jednotlivé moduly aplikace (frontend i backend).
- **Integrace komponent.** Jednotlivé části systému (např. databáze, API, uživatelské rozhraní) se propojují tak, aby systém fungoval jako celek.
- **Kontrola kvality kódu.** Průběžná kontrola kódu, jeho struktury a konzistence přispívá k udržení vysoké kvality softwaru (např. psaní dokumentace nebo optimalizace kódu).

Vývoj probíhá iterativně nebo v předem definovaných krocích, v závislosti na použitém modelu procesu vývoje softwaru (např. *Scrum* sprinty v agilním vývoji). Výstupem této fáze je plně funkční software připravený k testování.

## Testing

Fáze testování je zásadní částí životního cyklu softwaru, která zajišťuje, že vytvořený produkt splňuje požadavky uživatelů a je bez kritických chyb. Testování probíhá na různých úrovních a zahrnuje manuální i automatizované procesy, které ověřují funkčnost, výkonnost a bezpečnost systému.

## Typy testování:

- **Jednotkové testování:** Testování jednotlivých částí kódu, jako jsou funkce nebo metody, aby se ověřila jejich správnost.
- **Integrační testy:** Zajišťuje, že různé moduly systému spolu správně komunikují.
- **Systémové testování:** Komplexní test, který ověřuje funkčnost celého systému podle specifikací.
- **Akceptační testování:** Provádí uživatelé, aby zjistili, zda produkt splňuje jejich očekávání a je připraven pro nasazení.
- **Výkonnostní testování:** Ověřuje, zda software zvládá zátěž a funguje efektivně i při vysokém zatížení.
- **Regresní testování:** Kontrola, že nové změny v kódu nezpůsobily problémy ve stávající funkcionalitě.

Moderní vývoj softwaru často zahrnuje automatizované testování, které umožňuje rychlé a opakované provádění testů, čímž se zvyšuje efektivita celého procesu. Automatizace je často součástí *CI pipeline* (Continuous Integration), která umožňuje, aby každá změna v kódu byla automaticky zkонтrolována, otestována a integrována do hlavní větve projektu. Díky tomu mohou vývojáři rychle odhalit chyby, minimalizovat rizika při slučování kódu a zefektivnit spolupráci v týmu.

Výstupem této fáze je produkt, který je připraven pro uživatelské testování nebo nasazení do produkčního prostředí.

## Údržba

Fáze údržby softwaru hraje klíčovou roli v jeho životním cyklu, neboť zajišťuje, že systém zůstává funkční, bezpečný a přizpůsobený měnícím se požadavkům. Tato fáze zahrnuje opravy chyb, přidávání nových funkcí a přizpůsobení softwaru novým technologiím nebo legislativním změnám. Proces údržby lze dobře vysvětlit na základě Lehmanových zákonů vývoje softwaru, které popisují přirozenou evoluci softwarových systémů.

### Lehmanovy zákony v kontextu údržby:

- **Zákon trvalé proměny:** Software používaný v reálném prostředí se musí neustále měnit, aby odpovídal novým požadavkům, dokud není ekonomičtější jej restrukturalizovat nebo nahradit.
- **Zákon rostoucí složitosti:** Každá změna zvyšuje složitost softwaru, kterou je nutné pravidelně redukovat refaktorizací a optimalizací, jinak klesá jeho udržovatelnost.
- **Zákon vývoje programu:** Tempo změn se může krátkodobě jevit jako náhodné, ale dlouhodobě sleduje samoregulační proces, který lze statisticky předvídat.
- **Zákon invariantní spotřeby práce:** Celkový pokrok při vývoji softwaru je stabilní a nezávisí přímo na množství vynaložených zdrojů.

- **Zákon omezené velikosti přírůstku:** Překročení limitu množství změn v jedné verzi může vést k vážným problémům s kvalitou a stabilitou systému.

Údržba je často nejdelší a nákladově nejnáročnější částí životního cyklu, protože zahrnuje průběžnou práci na již nasazeném softwaru.

## Práce v týmu a verzování

Efektivní práce v týmu a správné řízení projektů jsou klíčovými faktory úspěchu při vývoji softwaru. Různé metodiky jako je *Scrum*, *Kanban* nebo *Waterfall* určují, jakým způsobem týmy plánují, sledují a realizují své úkoly. Správa verzí kódu, zajištěná například systémem *Git*, zase umožňuje efektivní spolupráci při vývoji.

### Projektové metodiky

- **Scrum:** Agilní přístup zaměřený na iterativní vývoj v rámci krátkých časových úseků, tzv. sprintů (obvykle 1–4 týdny). Týmy se pravidelně setkávají na stand-upech, kde sdílejí pokrok a překážky. Scrum role zahrnují:
  - **Product owner:** Zodpovídá za správu produktového backlogu, stanovuje priority úkolů a zajišťuje soulad s požadavky stakeholderů.
  - **Scrum master:** Koordinuje tým, zajišťuje dodržování Scrum metodiky, pomáhá odstraňovat překážky a spolupracuje s product ownerem.
  - **Tým vývojářů:** Realizuje zadané úkoly, vytváří funkční řešení a dodává výsledky v rámci sprintů.
- **Waterfall (vodopád):** Tradiční lineární model, kde jednotlivé fáze (analýza, návrh, vývoj, testování, nasazení) probíhají sekvenčně. Každá fáze musí být dokončena, než se přejde k další. Vhodné pro projekty s pevně definovanými požadavky.
- **Kanban:** Vizualizační metoda řízení úkolů, která umožňuje flexibilní plánování práce bez striktních iterací. Úkoly se přesouvají mezi stavy jako *To Do*, *In Progress* a *Done*.
- **Agilní přístupy obecně:** Metodiky jako *Extrémní programování* nebo *Lean development* kladou důraz na adaptivní plánování, průběžné zlepšování a rychlé dodání funkčního softwaru.
- **Nástroje pro spolupráci:** Týmy často využívají nástroje jako *Slack*, *Discord* nebo *Jira* pro komunikaci a sledování úkolů.

### Verzování kódu

*Git* je dnes nejpoužívanějším nástrojem pro správu verzí. Umožňuje sledovat změny kódu, pracovat paralelně na různých větvích a řešit konflikty.

### **Pracovní postup v Gitu:**

- **Hlavní větve:** Typicky se používají větve jako `main`, `development` a další specifické větve s prefixem `feature` pro nové funkce.
- **Pull requests a code review:** Každá změna prochází kontrolou kvality a schválením před sloučením do hlavní větve.
- **Řešení konfliktů:** Konflikty při slučování větví jsou řešeny týmem a vyžadují komunikaci a koordinaci.

Práce v týmu a správná správa verzí jsou základními stavebními kameny moderního vývoje softwaru. Díky těmto nástrojům a postupům mohou týmy spolupracovat efektivně, rychle reagovat na změny a udržovat vysokou kvalitu produktu.

## **DevOps a CI/CD pipeline**

*DevOps* je moderní přístup k vývoji softwaru, který propojuje vývojové a provozní týmy, aby zefektivnil a automatizoval procesy od psaní kódu až po nasazení aplikace. Klíčovým prvkem DevOps je *CI/CD pipeline* (Continuous Integration / Continuous Deployment), která umožňuje průběžnou integraci, testování a nasazení softwaru.

### **Continuous Integration (CI)**

CI zajišťuje, že všechny změny v kódu jsou průběžně integrovány do hlavní větve projektu a automaticky testovány.

#### **CI zahrnuje:**

- automatické spuštění testů (unit, integration, regression),
- kontrolu kvality kódu pomocí *lintování* nebo statické analýzy,
- rychlé odhalování chyb a minimalizaci konfliktů při slučování kódu.

### **Continuous Deployment (CD)**

CD umožňuje automatizované nasazení softwaru do produkčního prostředí a zkracuje dobu mezi vývojem a nasazením.

#### **CD zahrnuje:**

- spuštění pipeline pro build a přípravu kódu,
- automatické nasazení do testovacího nebo produkčního prostředí,
- možnost rollbacku v případě problémů.

### Výhody DevOps a CI/CD:

- zrychlení vývoje a nasazení díky automatizaci,
- vyšší kvalita kódu a stabilnější verze softwaru,
- možnost kontinuálních aktualizací a rychlá reakce na změny.

Mezi nejčastěji používané nástroje patří *GitHub Actions*, *GitLab CI/CD* nebo *Jenkins*. Nasazení CI/CD pipeline je klíčovým krokem k zajištění efektivity, kvality a flexibility moderního vývoje.

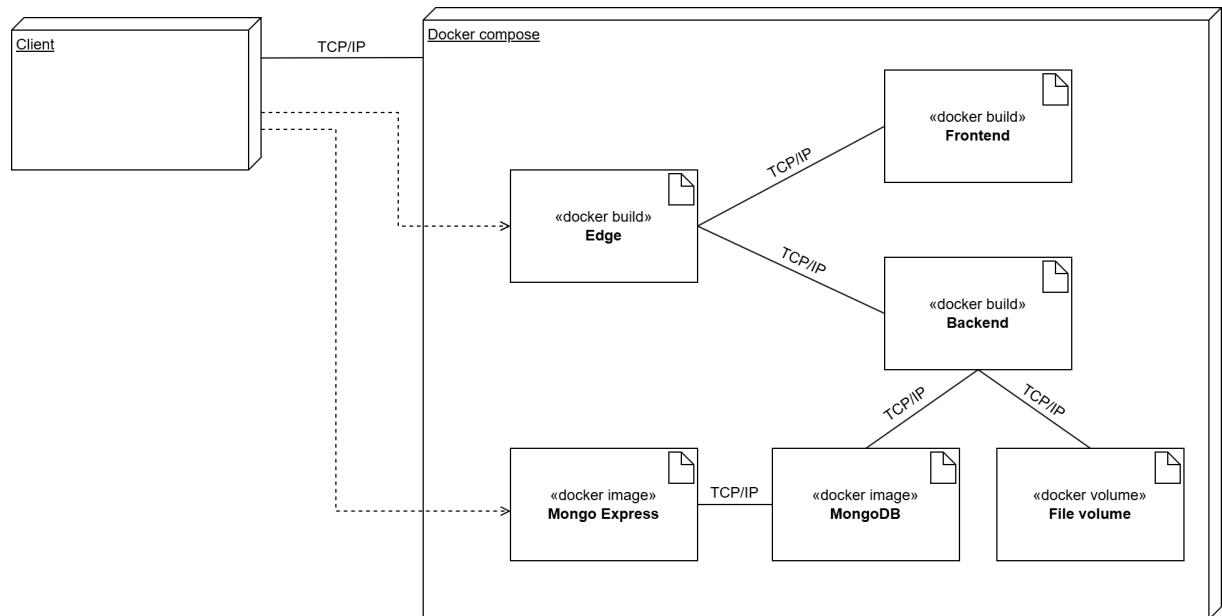
# 4. Praktická část

## 4.1. Návrh architektury, komponent a schématu databáze platformy

Tato kapitola se zaměřuje na návrh architektury, komponent a databázového schématu platformy pro plánování výuky a sdílení výukových plánů. Prostřednictvím diagramů nasazení je znázorněno nasazení dvou hlavních aplikací – *EDUBO* (editor výukových plánů) a *Eduklub – plány výuky* (webový agregátor sdílených výukových plánů). Obě aplikace jsou nasazeny pomocí Docker Compose, což zajišťuje modularitu a snadnou správu jednotlivých komponent.

Dále jsou v kapitole rozebrány struktury složek a databázové modely, které popisují organizaci backendu a frontendu. Entitně-vztahové diagramy a databázová schémata pak detailně znázorňují vztahy mezi hlavními entitami systému.

### Nasazení aplikace EDUBO



Obrázek 4.1.: Diagram nasazení znázorňující komunikaci přes TCP/IP protokol aplikace *EDUBO*.

Tento diagram nasazení znázorňuje strukturu nasazení aplikace pro plánování výuky v prostředí Docker Compose, kde jsou jednotlivé komponenty aplikace rozděleny do kontejnerů. Kontejnery

spolu komunikují pomocí TCP/IP protokolu na vybraných portech.

Klient představuje zařízení uživatele (např. počítač nebo mobil), který přistupuje k aplikaci. Komunikuje s **Edge** kontejnerem (NGINX) přes protokol TCP/IP, který zajišťuje směrování požadavků do backendu nebo frontendu. Správce komunikuje přímo s kontejnerem **Mongo Express**, který slouží k správě databáze MongoDB.

Celý systém je nasazen v rámci prostředí Docker Compose, které umožňuje spravovat více kontejnerů najednou. Následuje přehled jednotlivých komponent:

### **NGINX**

NGINX je nakonfigurován jako reverzní proxy server. Jeho úkolem je přijímat požadavky od klienta a směrovat je na příslušné komponenty v rámci aplikace, například do frontendu nebo backendu. Tento kontejner je sestaven pomocí **Dockerfile** (v diagramu vyjádřeno jako stereotyp «*docker build*»), což umožňuje přizpůsobit konfiguraci NGINX podle specifických potřeb aplikace (např. využití Brotli komprese). Komunikuje s frontendem a backendem přes TCP/IP a také zprostředkovává připojení klienta k Mongo Express a dalším službám.

### **Frontend**

Tato komponenta je zodpovědná za zobrazení uživatelského rozhraní. Obsahuje kód frontendu aplikace (např. HTML, CSS, TypeScript) a je vystavena prostřednictvím NGINX. Tento kontejner je sestaven pomocí **Dockerfile**, aby zahrnoval specifické konfigurace a balíčky potřebné pro provoz uživatelského rozhraní. Komunikuje s NGINX přes TCP/IP protokol, který přijímá požadavky od klienta a přesměrovává je do frontendu.

### **Backend**

Backend obsahuje aplikační logiku, která zpracovává požadavky z frontendu. Je vystaven prostřednictvím NGINX a komunikuje s dalšími službami (např. MongoDB). Tento kontejner je sestaven pomocí **Dockerfile**, což umožňuje nastavit backend tak, aby odpovídal specifickým požadavkům aplikace, včetně požadovaných knihoven a závislostí. Zajišťuje interakce mezi uživatelskými požadavky (přijatými přes frontend) a databází, případně dalšími zdroji.

### **MongoDB**

MongoDB je dokumentová databáze používaná k ukládání dat aplikace. Tento kontejner je vytvořen pomocí obrazu z DockerHubu (v diagramu vyjádřeno jako stereotyp «*docker image*»), což zjednodušuje nasazení databáze bez nutnosti vlastní konfigurace. Komunikuje s backendem přes TCP/IP protokol a poskytuje úložiště pro data, která jsou uložena ve formátu Binary JSON (BSON).

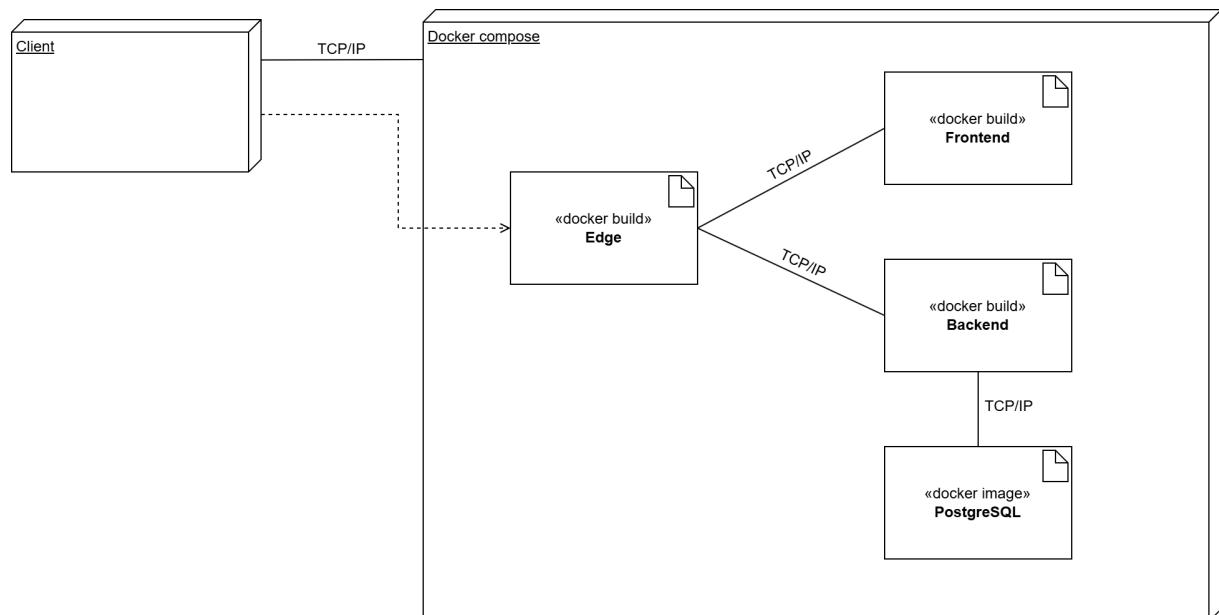
## Mongo Express

Mongo Express je webové rozhraní pro správu MongoDB, které umožňuje vizuální správu databáze. Běží jako samostatný kontejner a poskytuje přístup k MongoDB pro administrativní účely. Tento kontejner je vytvořen pomocí obrazu z DockerHubu, což poskytuje rychlý a snadný způsob pro správu MongoDB. Komunikuje přímo s MongoDB pomocí TCP/IP protokolu a umožňuje správci snadno procházet a manipulovat s daty uloženými v databázi.

## File Volume

File Volume je úložný prostor pro soubory, které jsou nahrány do hodin. Tento objem zajišťuje, že soubory a data budou uchovány i při restartu kontejnerů a slouží jako externí úložiště.

## Nasazení aplikace Eduklub – plány výuky

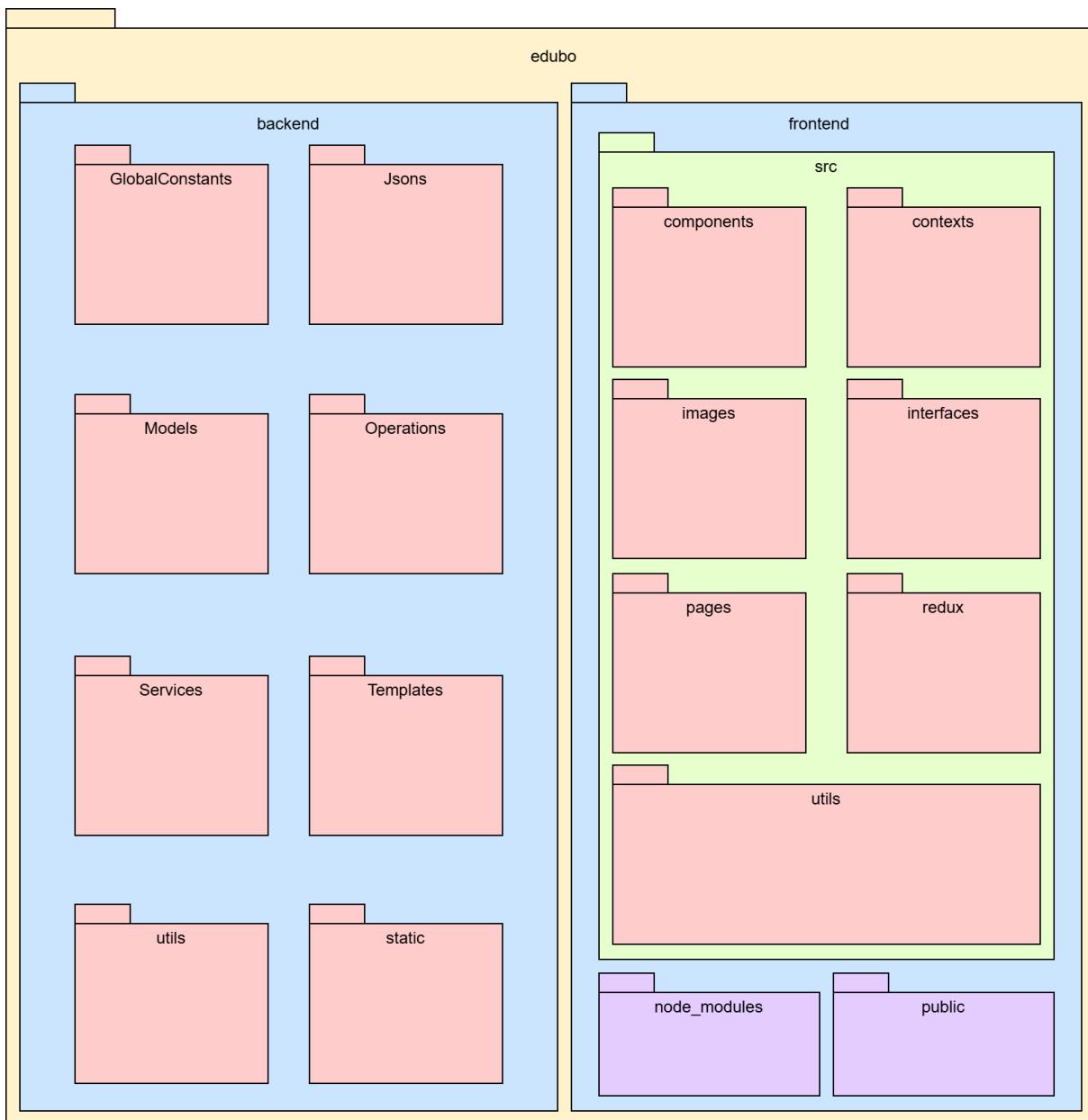


Obrázek 4.2.: Diagram nasazení znázorňující komunikaci přes TCP/IP protokol aplikace *Eduklub – plány výuky*.

Tento diagram nasazení znázorňuje nasazení webového agregátoru výukových plánů. Architektura aplikace zůstává téměř shodná s nasazením *EDUBO*, přičemž jednotlivé komponenty běží v kontejnerech řízených pomocí Docker Compose. Klient komunikuje s aplikací přes reverzní proxy server NGINX, který směruje požadavky mezi frontendem a backendem.

Hlavní změnou oproti *EDUBO* je nahrazení dokumentové databáze MongoDB relační databází PostgreSQL. Tato změna byla provedena s cílem efektivnější správy strukturovaných dat a umožnění komplexnějších dotazů nad výukovými plány, například pro filtrování mezi sdílenými plány podle různých kritérií, jako je předmět, kategorie nebo klíčová slova. Ostatní komponenty, včetně frontendu, backendu a jejich vzájemné komunikace, zůstávají stejné.

## Struktura složek aplikace EDUBO



Obrázek 4.3.: Diagram balíčků znázorňující strukturu složek aplikace *EDUBO*.

Tento diagram balíčků zobrazuje strukturu složek aplikace *EDUBO* postavené na backendu pomocí Flask (Python) a frontendu pomocí React (TypeScript). Aplikace je rozdělena do dvou hlavních částí: **backend** a **frontend**, přičemž každá část obsahuje své specifické složky a komponenty, které zajišťují určité funkcionality.

### Backend

- **GlobalConstants:** Tato složka obsahuje globální konstanty, které jsou využívány v celém backendu. Tyto konstanty zahrnují nastavení aplikace, konfigurace nebo klíčové proměnné, které jsou sdíleny mezi jednotlivými moduly.

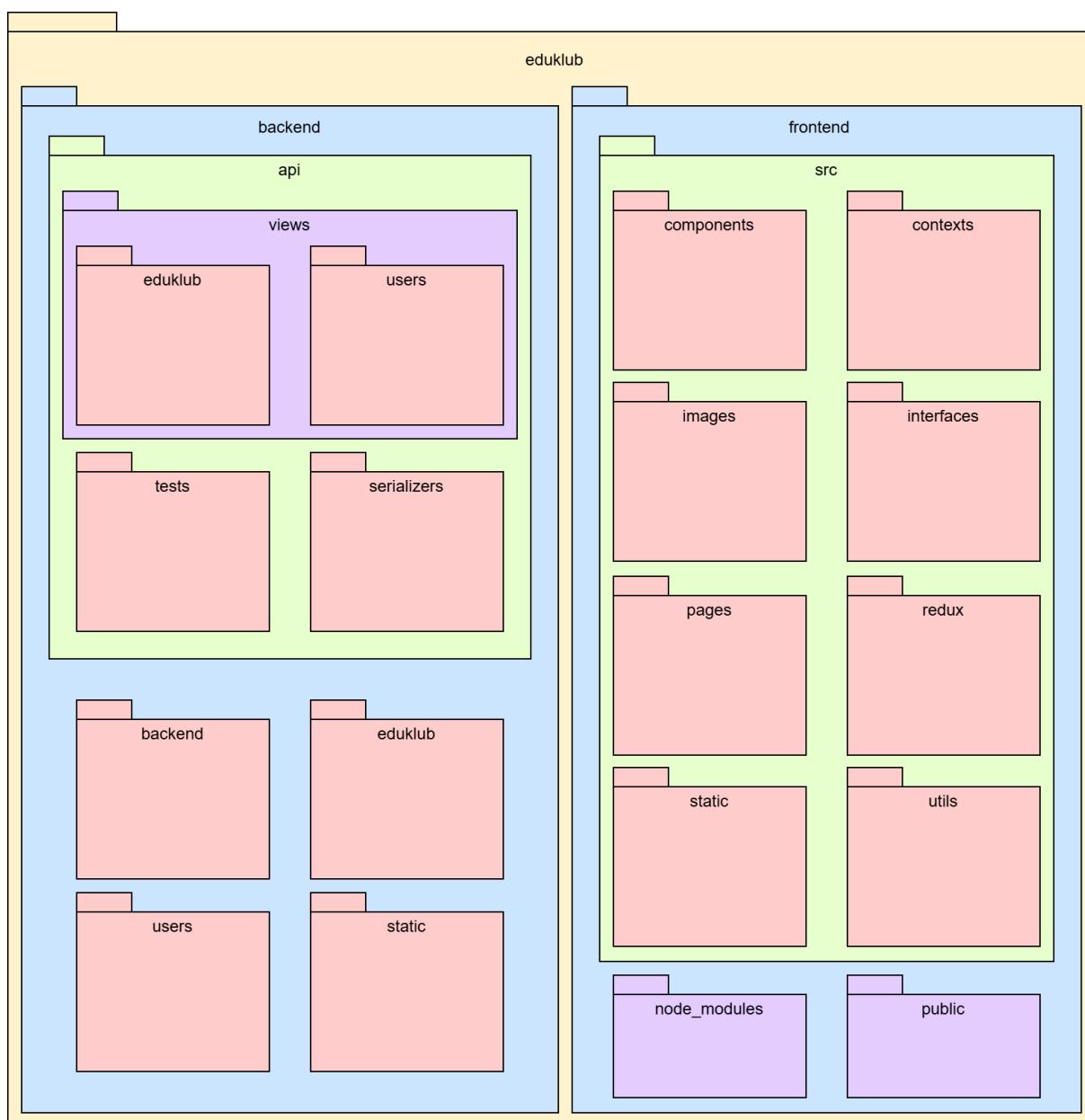
- **Jsons:** Složka obsahuje příkladné JSON soubory, které ukazují, jak vypadají datové struktury typů aktivit, výukového plánu, kurikulárních výstupů a uživatelských dat.
- **Models:** Obsahuje Pydantic modely a schémata, které definují strukturu a validaci dat.
- **Operations:** Tato složka zahrnuje databázové operace, tedy CRUD (Create, Read, Update, Delete) operace pro MongoDB. Každá operace je implementována jako funkce, která interaguje s databází a umožňuje manipulaci s uloženými daty.
- **Services:** Obsahuje API endpointy, které jsou definovány pomocí Flask blueprintů. Tyto služby fungují podobně jako views v Djangu, kde každý service odpovídá určitému API endpointu a zpracovává konkrétní typ požadavku (např. GET, POST). Blueprinty v této složce rozdělují logiku aplikace na menší, modulární části, což usnadňuje správu a rozšiřování kódu.
- **Templates:** Složka pro šablony určené k vykreslení PDF dokumentů, konkrétně plánů výuky. Obsahuje Jinja2 kód, který se generuje jako HTML výstup, který si mohou uživatelé stáhnout nebo vytisknout ve formě PDF.
- **utils:** Obsahuje pomocné utility, což mohou být například funkce nebo skripty, které usnadňují opakované operace napříč aplikací. Tyto utility zahrnují formátování dat, konverze nebo jiné malé pomocné operace.
- **static:** Slouží k ukládání statických souborů, jako jsou obrázky, styly (CSS) nebo JavaScript, které jsou nezávislé na dynamicky generovaném obsahu aplikace.

## Frontend

- **src:**
  - **components:** Obsahuje React komponenty, které tvoří jednotlivé části uživatelského rozhraní. Tyto komponenty slouží v systému jako znovupoužitelné prvky, například tlačítka, formuláře nebo karty, které se skládají dohromady, aby vytvořily plné stránky aplikace.
  - **contexts:** Obsahuje kontexty pro React, což jsou data, která lze sdílet napříč komponentami.
  - **images:** Složka pro ukládání obrázků používaných v uživatelském rozhraní, jako jsou loga, ikony nebo ilustrace.
  - **interfaces:** Závazky pro TypeScript, které definují struktury dat používané v aplikaci. Používají se k zajištění typové bezpečnosti při práci s daty.
  - **pages:** Obsahuje jednotlivé stránky aplikace (např. Přihlášení, Registrace, Editor). Každá stránka je tvořena kombinací komponent a zobrazuje konkrétní obsah aplikace.
  - **redux:** Obsahuje konfigurace a reducery pro Redux, což je knihovna pro správu stavu aplikace.

- **utils:** Stejně jako na backendu, tato složka obsahuje pomocné funkce, které se používají napříč frontendem. Jsou zde například funkce pro formátování dat.
- **node\_modules:** Tato složka obsahuje všechny závislosti a knihovny nainstalované pomocí Node.js. Je automaticky generována při instalaci balíčků a obsahuje knihovny třetích stran.
- **public:** Složka pro veřejné soubory, které jsou přístupné přímo z prohlížeče (např. favicon, index.html). Tyto soubory jsou statické a nemění se během běhu aplikace.

## Struktura složek Eduklub – plány výuky



Obrázek 4.4.: Diagram balíčků znázorňující strukturu složek aplikace *Eduklub – plány výuky*.

Tento diagram balíčků zobrazuje strukturu složek webové agregátoru výukových plánů, kde je backend vytvořen pomocí Django (Python) a frontend pomocí React (TypeScript). Diagram ukazuje hlavní složky a jejich účel, což poskytuje přehled o architektuře aplikace. Struktura frontendových složek je v package diagramu shodná se strukturou použitou v *EDUBO*.

## Backend

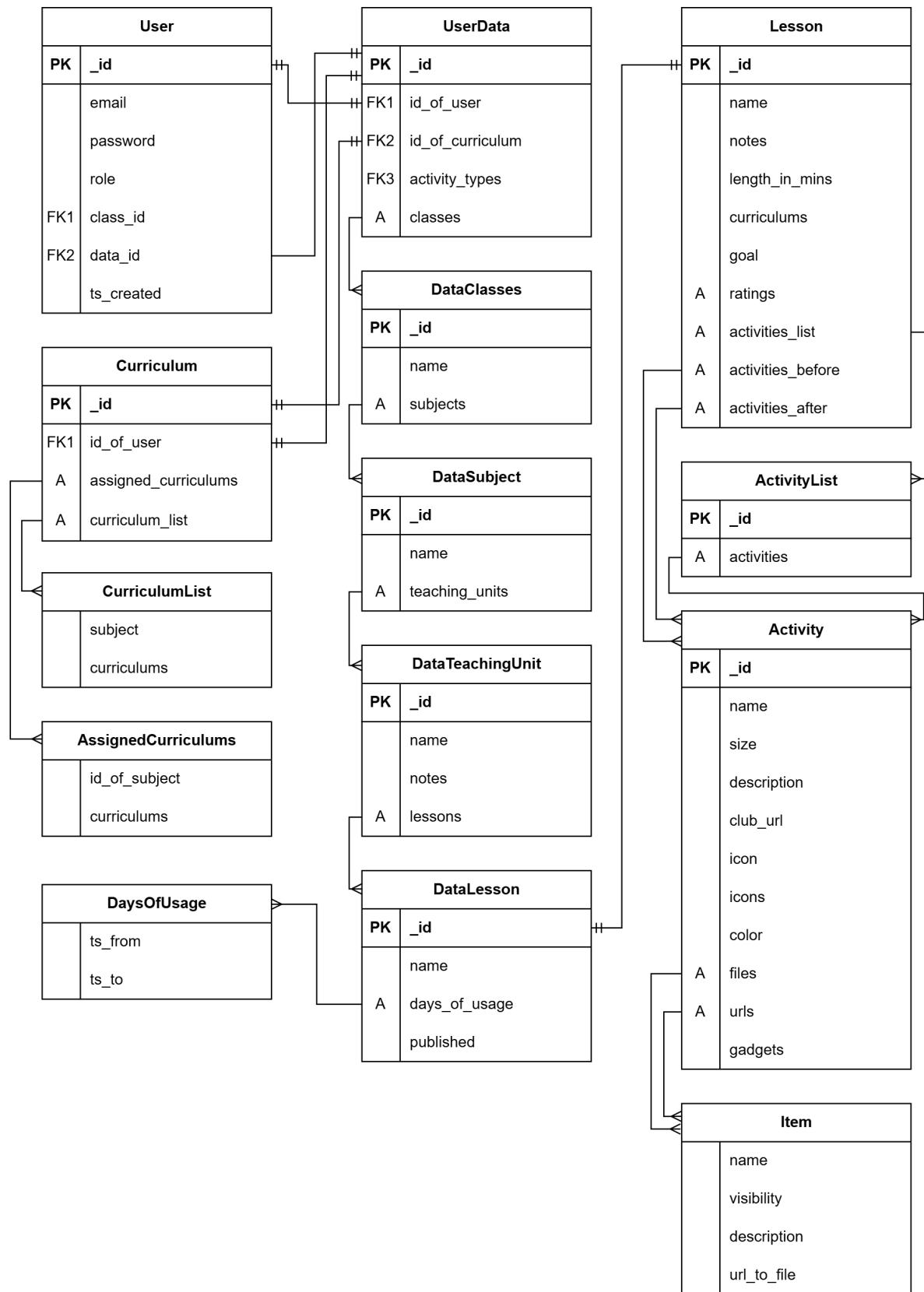
- **api:** Obsahuje strukturu pro API aplikace, která je rozdělena na jednotlivé části zajišťující různé funkce aplikace.
- **views:**
  - **eduklub:** Obsahuje views pro hlavní funkce aplikace, které zajišťují zpracování požadavků týkajících se obsahu a funkcí platformy.
  - **users:** Obsahuje views specifické pro správu uživatelů, například autentizaci, registraci a správu uživatelských profilů.
- **serializers:** Obsahuje serializéry, které převádějí data mezi Django modely a JSON formátem.
- **tests:** Obsahuje testy pro jednotlivé části API, které slouží k ověření funkčnosti a správnosti implementace.
- **backend:** Obsahuje hlavní konfigurační soubory a nastavení specifické pro backend aplikace. Může obsahovat například nastavení databáze, middleware nebo bezpečnostní konfigurace.
- **eduklub:** Tato složka obsahuje modely všech objektů kromě uživatelů.
- **users:** Složka pro správu uživatelských účtů a uživatelských modelů. Obsahuje funkce pro autentizaci, autorizaci a správu profilů.
- **static:** Obsahuje statické soubory potřebné pro backend, například styly, JavaScript nebo obrázky, které jsou potřebné pro generování určitých šablon nebo statických zobrazení v Djangovém prostředí.

## Databázové schéma aplikace EDUBO

V systému se nachází následující entity:

- Uživatel – osoba, která pracuje s aplikací. Může to být učitel, žák nebo rodič.
- Třída – organizační jednotka, ke které jsou přiřazeni žáci a učitelé.
- Předmět – oblast výuky (např. matematika, čeština), k níž jsou přiřazeny výukové celky.
- Výukový celek – sada tematicky souvisejících výukových plánů.
- Výukový plán – konkrétní plán jedné vyučovací jednotky.

- Aktivita – základní stavební prvek výukového plánu, který definuje konkrétní činnost (např. diskuse, práce s textem, test).



Obrázek 4.5.: Entitně vztahový diagram znázorňující databázovou strukturu aplikace EDUBO.

Tento entitně vztahový diagram popisuje databázovou strukturu pro aplikaci *EDUBO*. Diagram je navržen specificky pro dokumentovou databázi, kde některé tabulky fungují jako objekty s poli dalších podobjektů. Šipky s vidlicí vedoucí přímo do názvu tabulky označují podtabulky (pole objektů), což znamená, že pokud má záznam v prvním sloupci „A“, odkazuje na pole těchto objektů. Tento diagram není kompletní a chybí zde šest tabulek (**Class**, **ClassKey**, **ParentKey**, **ActivityTypes**, **ActivityTypeList**, **Rating**) a to kvůli čitelnosti. Kompletní diagram se nachází v externí příloze. Diagram zajišťuje organizaci dat týkajících se uživatelů, dat učitelů a výukových hodin.

### **Popis důležitých objektů:**

- **User:** Obsahuje základní informace o uživatelích systému.

Atribut **role** definuje roli uživatele v systému (student, teacher nebo parent). Atribut **data\_id** odkazuje na objekt v tabulce **UserData**, který obsahuje podrobné informace o uživateli.

- **Curriculum:** Tabulka, která obsahuje přiřazené kurikulární výstupy k předmětům.

Obsahuje dvě hlavní pole:

- **assigned\_curriculums** – výčet výstupů, které učitel přiřadil k danému předmětu. Každý záznam obsahuje **id\_of\_subject** (identifikátor předmětu) a **curriculum** (pole výstupů).
- **curriculum\_list** – výchozí a vlastní nahrané výstupy, které může uživatel přiřadit ke svým předmětům. Každý záznam obsahuje název předmětu a pole výstupů.

- **UserData:** Obsahuje podrobné informace o uživateli.

Obsahuje např. pole **classes**, které odkazuje na jednotlivé třídy. Každá třída obsahuje pole **subjects**, ve kterém se nachází název předmětu a přiřazené výukové celky. Každý výukový celek obsahuje poznámky a seznam výukových hodin (**lessons**). Každá hodina má příznak **published** a pole **days\_of\_usage**, které definuje, kdy je hodina použita.

- **Lesson:** Tabulka obsahující informace o jednotlivých výukových hodinách.

Obsahuje název, poznámky, délku v minutách, přiřazené kurikulární výstupy, cíl a pole paralelních aktivit **activities\_list**. Každý prvek v **activities\_list** obsahuje pole aktivit, které odkazují na tabulku **Activity**.

Dále obsahuje volitelné bloky **activities\_before** a **activities\_after**, které mohou obsahovat pole aktivit s následujícími atributy: název, velikost, popis, odkaz na článek na Eduklubu (**club\_url**), pomůcky (**gadgets**), soubory a URL odkazy. Každý soubor nebo odkaz obsahuje název, viditelnost, popis a odkaz na soubor.

```

1 class ExtendedItems(Schema):
2     name = fields.String(validate=validate.Length(max=1023))
3     visibility = fields.String(validate=validate.OneOf(["student", "teacher", "after-lesson"]))
4     description = fields.String(validate=validate.Length(max=1023))
5     url_to_file = fields.String()
6
7 class ActivitySchema(Schema):
8     _id = fields.String(validate=validate.Length(equal=36))
9     name = fields.String(validate=validate.Length(max=127))
10    size = fields.Integer(validate=validate.Range(min=1))
11    description = fields.String(validate=validate.Length(max=8191))
12    club_url = fields.String()
13    icon = fields.String(validate=validate.OneOf(["xxx"]))
14    icons = fields.List(fields.String(validate=validate.OneOf(["xxx"])))
15    color = fields.String(validate=validate.Length(equal=7))
16    files = fields.List(fields.Nested(ExtendedItems))
17    urls = fields.List(fields.Nested(ExtendedItems))
18    gadgets = fields.List(fields.String(validate=validate.Length(max=255)))
19        )
20
21 class ActivityList(Schema):
22     _id = fields.String(validate=validate.Length(equal=36))
23     activities = fields.List(fields.Nested(ActivitySchema))
24
25 class Rating(Schema):
26     id_of_user = fields.String()
27     rating = fields.String(validate=validate.OneOf(["downvote", "upvote", "neutral", "unused"]))
28
29 class LessonDatabaseSchema(Schema):
30     _id = fields.String(validate=validate.Length(equal=36))
31     ts_created = fields.String()
32     ts_updated = fields.String()
33     name = fields.String(validate=validate.Length(max=68))
34     notes = fields.String(validate=validate.Length(max=4095))
35     length_in_mins = fields.Integer(validate=validate.Range(min=5, max=720))
36     curriculums = fields.List(fields.String())
37     goal = fields.String(validate=validate.Length(max=511))
38     ratings = fields.List(fields.Nested(Rating))
39     activities_list = fields.List(fields.Nested(ActivityList))

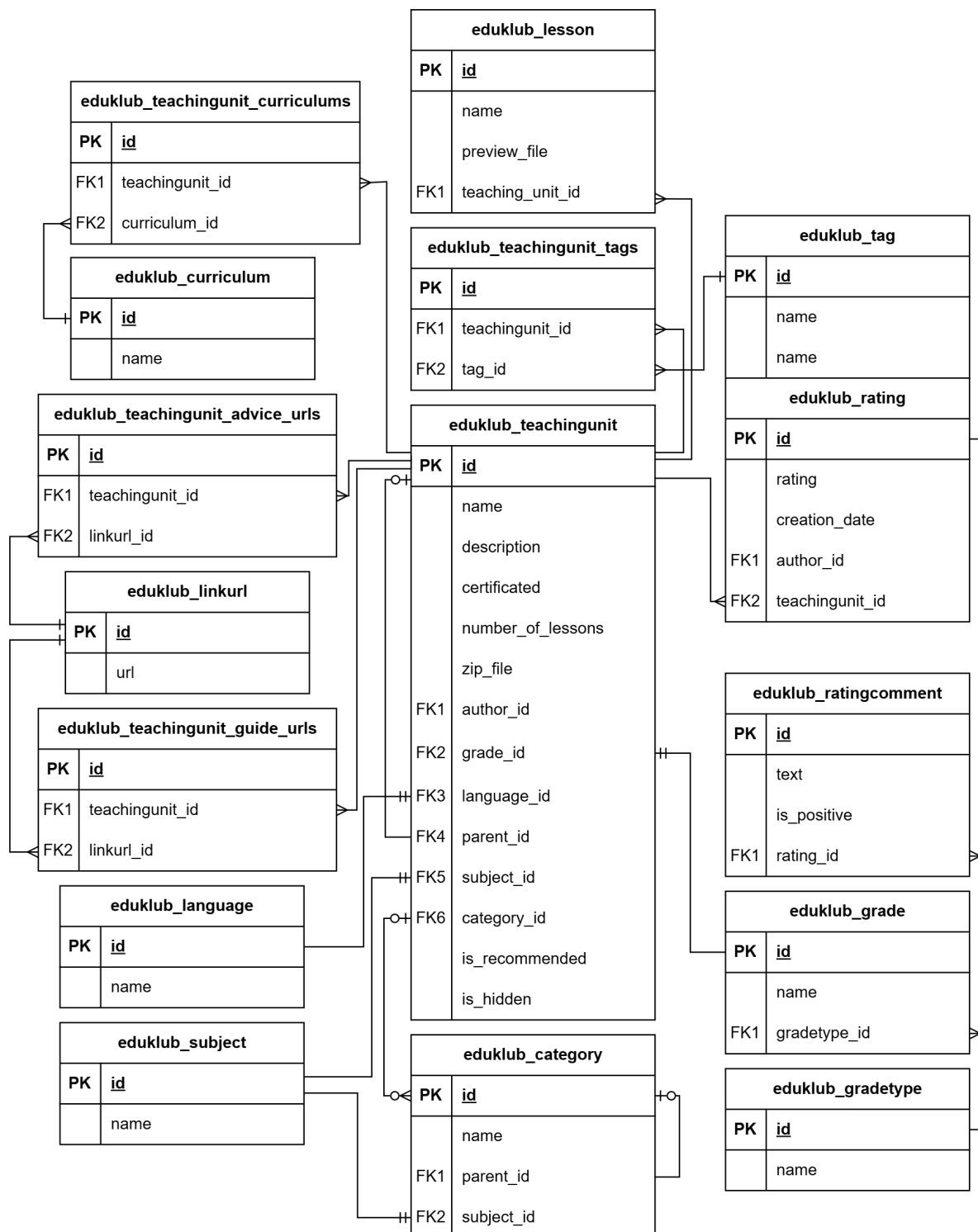
```

```
39 activities_before = fields.List(fields.Nested(ActivitySchema))
40 activities_after = fields.List(fields.Nested(ActivitySchema))
```

Toto schéma definuje strukturu výukového plánu pomocí ORM (Objektově-relační mapování) knihovny Marshmallow v jazyce Python. Každá lekce je reprezentována třídou `LessonDatabaseSchema`, která obsahuje klíčové informace, jako je název, čas vytvoření, poznámky a délka lekce. Lekce může mít přiřazené kurikulární výstupy a definovaný cíl hodiny. Klíčovou součástí schématu je seznam aktivit, který je strukturován do tří hlavních částí – běžné aktivity v hodině (`activities_list`), aktivity před hodinou (`activities_before`) a aktivity po hodině (`activities_after`). Každá aktivita (`ActivitySchema`) má své unikátní ID, název, popis, velikost a další atributy, jako je barva, soubory, URL odkazy a pomůcky.

Další částí modelu je hodnocení výukových plánů, které je reprezentováno třídou `Rating`. Každý žák může označit lekci jako pozitivní, negativní, neutrální nebo ji neohodnotit. Pro správu souborů a odkazů byla vytvořena samostatná struktura `ExtendedItems`, která určuje viditelnost těchto položek (pro studenty, učitele nebo po hodině).

## Databázové schéma aplikace Eduklub – plány výuky



Obrázek 4.6.: Entitně vztahový diagram znázorňující databázovou strukturu aplikace *Eduklub – plány výuky*.

Tento entitně vztahový diagram popisuje databázovou strukturu pro webový agregátor výukových plánů. Diagram není kompletní a chybí zde tři tabulky (`favoritelist_teaching_units`, `user`, `favoritelist`) z důvodu úspory místa v textu této závěrečné práce. Kompletní diagram se nachází v externí příloze.

Popis důležitých tabulek:

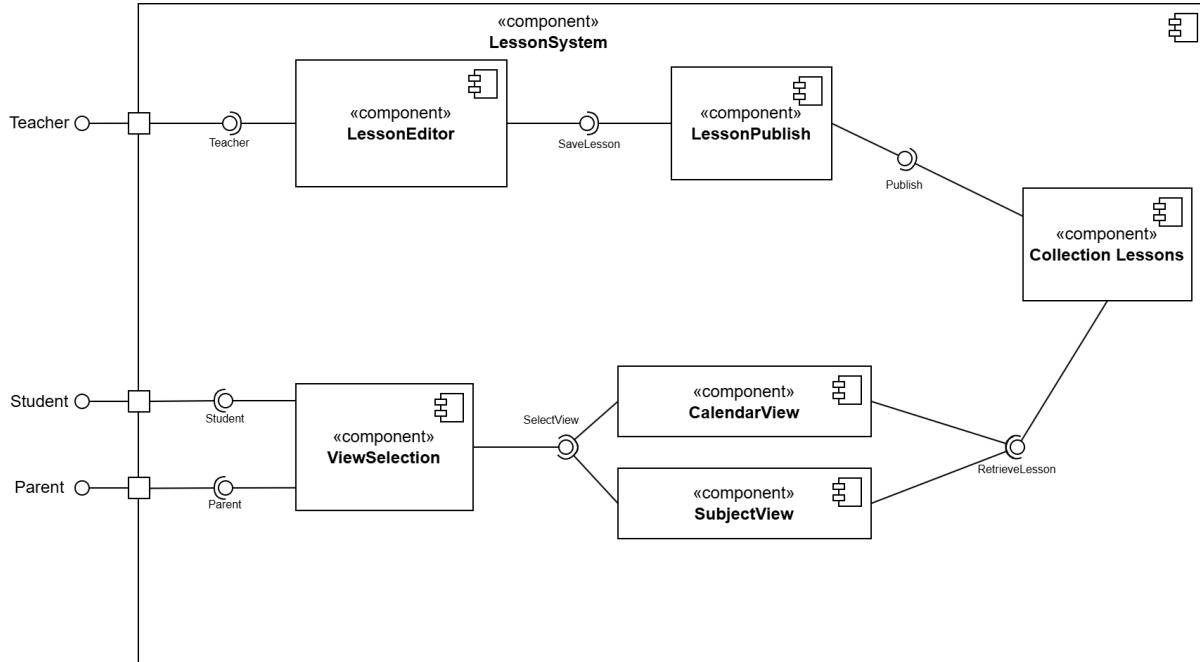
- **eduklub\_teachingunit:** Centrální tabulka pro výukové celky v aplikaci.
  - **name, description:** Název a popis výukového celku.
  - **certificated:** Příznak určující, zda je plán certifikovaný.
  - **number\_of\_lessons:** Počet výukových hodin ve výukovém celku.
  - **author\_id:** Odkaz na uživatele, který jednotku vytvořil.
  - **grade\_id:** Ročník, do kterého výukový celek spadá.
  - **parent\_id:** Odkaz na rodičovský celek, pokud se jedná o mutaci jiného celku.
  - **subject\_id:** Předmět, do kterého výukový celek spadá.
  - **category\_id:** Kategorie, do kterého výukový celek spadá. Tabulka **eduklub\_category** obsahuje **subject\_id**. Tento vztah umožňuje, aby kategorie existovaly samostatně, i bez přiřazení konkrétního výukového celku.
- **eduklub\_lesson:** Tabulka pro jednotlivé výukové hodiny, které jsou součástí výukových celků.
  - **name:** Název výukové hodiny.
  - **preview\_file:** Náhledový HTML soubor výukové hodiny.
  - **teachingunit\_id:** Odkaz na výukový celek, ke kterému výuková hodina patří.
- **eduklub\_category:** Tabulka pro kategorie výukových celků.
  - **name:** Název kategorie.
  - **parent\_id:** Možnost hierarchického uspořádání kategorií.
- **eduklub\_tag:** Tabulka pro značky, které lze přiřadit výukovým celkům.
- **eduklub\_rating:** Tabulka pro hodnocení výukových celků.
  - **rating:** Hodnota hodnocení.
  - **creation\_date, author\_id:** Datum vytvoření a odkaz na uživatele, který hodnocení vytvořil.
- **eduklub\_ratingcomment:** Obsahuje komentáře k hodnocením.
  - **id, text:** Primární klíč a text komentáře.
  - **is\_positive:** Příznak indikující, zda je komentář pozitivní.
  - **rating\_id:** Odkaz na hodnocení, ke kterému komentář patří.
- **eduklub\_gradetype:** Typ vzdělávacího stupně, který poskytuje specifikaci pro jednotlivé ročníky.
  - **id, name:** Primární klíč a název typu ročníku.

- **eduklub\_grade**: Obsahuje ročníky ze vzdělávacích stupňů, pro které je výukový celek určen.
    - name, grade\_type\_id: Název ročníku a odkaz na typ ročníku.

```
1 class TeachingUnit(models.Model):
2     name = models.CharField(max_length=63)
3     description = models.TextField(null=True, blank=True)
4     certificated = models.BooleanField(default=False)
5     is_recommended = models.BooleanField(default=False)
6     creation_date = models.DateField(auto_now_add=True)
7     is_hidden = models.BooleanField(default=False)
8     number_of_lessons = models.IntegerField()
9     number_of_downloads = models.IntegerField(default=0)
10    zip_file = models.FileField(upload_to='teaching_units/zip_files')
11    language = models.ForeignKey(Language, on_delete=models.PROTECT)
12    author = models.ForeignKey(User, on_delete=models.PROTECT, null=True
13        )
14    tags = models.ManyToManyField(Tag, related_name='teaching_units',
15        blank=True)
16    parent = models.ForeignKey('self', on_delete=models.PROTECT, null=
17        True, blank=True)
18    alternatives = models.ManyToManyField('self', blank=False)
19    advice_urls = models.ManyToManyField(LinkURL, related_name='
20        advice_urls', blank=True)
21    guide_urls = models.ManyToManyField(LinkURL, related_name='
22        guide_urls', blank=True)
23    subject = models.ForeignKey(Subject, on_delete=models.PROTECT)
24    grade = models.ForeignKey(Grade, on_delete=models.PROTECT)
25    curriculums = models.ManyToManyField(Curriculum, related_name='
26        teaching_units', blank=True)
27    category = models.ForeignKey(Category, on_delete=models.PROTECT,
28        null=True, blank=True)
29
30    class Lesson(models.Model):
31        name = models.CharField(max_length=63)
32        preview_file = models.FileField(upload_to='lessons/preview_files')
33        teaching_unit = models.ForeignKey(TeachingUnit, on_delete=models.
34            CASCADE, related_name='lessons')
35
36    class Category(models.Model):
37        name = models.CharField(max_length=63)
38        parent = models.ForeignKey('self', on_delete=models.PROTECT, null=
39            True, blank=True, related_name='children')
40        subject = models.ForeignKey(Subject, on_delete=models.PROTECT,
```

```
blank=True, null=True)
```

Toto schéma definuje strukturu výukového celku a jeho jednotlivých lekcí pomocí ORM Django. **TeachingUnit** reprezentuje výukový celek s atributy jako název, popis, počet lekcí a soubor ke stažení. Každý výukový celek může mít rodičovský celek (**parent**), což umožňuje vytváření mutací. **Lesson** představuje jednotlivé lekce a obsahuje soubor náhledu ve formátu **.html**. **Category** pak slouží k tematickému zařazení výukových celků. Kategorie mohou být hierarchicky uspořádané pomocí vazby **parent**, což umožňuje vytvářet nadřazené a podřazené skupiny.



Obrázek 4.7.: Komponentní diagram znázorňující tok dat mezi hlavními částmi systému pro práci s výukovými plány.

Tento komponentní diagram znázorňuje hlavní funkční části systému pro práci s výukovými plány a jejich vzájemné propojení. Učitel využívá **LessonEditor** výukových plánů k tvorbě a úpravě hodin. Po dokončení může plán uložit, což umožní jeho pozdější úpravy nebo jej pomocí komponenty **PublishLesson** publikovat. Publikované hodiny se ukládají do **CollectionLessons**, což je hlavní úložiště všech dostupných výukových hodin.

Studenti a rodiče mají přístup k výukovým hodinám prostřednictvím komponenty **ViewSelection**, která jim umožňuje volit mezi dvěma režimy zobrazení – **CalendarView** pro přehledný kalendářní pohled na hodiny podle data nebo **SubjectView**, kde jsou hodiny organizovány podle předmětů. Oba způsoby zobrazují relevantní výukové hodiny uložené v **CollectionLessons**.

## 4.2. Odůvodnění výběru technického zásobníku

Výběr správného technického zásobníku je zásadním krokem při návrhu softwaru, který ovlivňuje celý životní cyklus projektu, od vývoje až po nasazení a údržbu. Zvolená architektura a technologie přímo ovlivňují flexibilitu, škálovatelnost, výkon a také efektivitu práce vývojového týmu. Důležité je zohlednit nejen technické požadavky, ale také zkušenosti týmu a dostupné zdroje. Tato kapitola se zaměřuje na zdůvodnění volby konkrétního technického zásobníku pro systémy *EDUBO* a *Eduklub – plány výuky*.

### Výběr architektury

Při návrhu architektury softwaru byly zvažovány různé přístupy, včetně monolitické a mikroservisní architektury. Pro oba projekty, *EDUBO* i *Eduklub – plány výuky*, byla zvolena monolitická architektura ve spojení s Docker Compose. Tato volba byla ovlivněna jednoduchostí vývoje, nasazení a správy systému, což bylo klíčové vzhledem k velikosti projektů a zkušenostem týmu.

Monolitická architektura umožnila integrovat všechny části aplikace do jednoho repozitáře, což zjednodušilo správu kódu i týmovou spolupráci. Všechny komponenty systému běží společně, což minimalizuje složitost při komunikaci mezi jednotlivými částmi aplikace, jako je frontend, backend a databáze.

#### Frontend

Pro oba projekty *EDUBO* a *Eduklub – plány výuky*, byl jako frontendová technologie zvolen React, což je populární framework pro tvorbu uživatelských rozhraní. Tento výběr byl založen především na zkušenostech týmu a požadavcích na rychlé a dynamické uživatelské rozhraní.

React umožňuje vytvářet komponentově orientované aplikace, což znamená, že jednotlivé části uživatelského rozhraní (například tlačítka, formuláře nebo karty) jsou vytvořeny jako samostatné komponenty, které lze znova použít v různých částech aplikace. Tato modularita usnadňuje nejen vývoj, ale také údržbu a rozšiřování systému.

Důvody výběru Reactu:

1. Zkušenosti týmu: Většina vývojového týmu měla zkušenosti s Reactem, což umožnilo rychlejší začátek vývoje a minimalizovalo potřebu doučování se nových technologií.
2. Podpora mobilního zobrazení: Ačkoliv byla zvažována alternativa React Native pro mobilní aplikace, zvolený přístup s Reactem a responsivním designem byl dostatečný pro zajištění kompatibility na mobilních zařízeních.
3. Komunitní podpora a ekosystém: React je široce podporován vývojářskou komunitou, což poskytlo přístup k rozsáhlé dokumentaci, nástrojům a knihovnám (např. React Router, Redux nebo react-beautiful-dnd).

Pro sestavení frontendového kódu byly použity různé sestavovací nástroje. *EDUBO* bylo původně vytvořeno pomocí Create React App, což je jednoduchý a dobře známý nástroj pro rychlé vytvoření React aplikací. Naopak *Eduklub – plány výuky* byl vyvíjen s použitím moderního sestavovacího nástroje Vite, který nabízí rychlejší a efektivnější vývoj, díky okamžitému znovu načtení modifikací kódu a optimalizovanému balíčkování.

Během vývoje se ukázalo, že Vite výrazně urychluje práci s kódem, což vedlo k rozhodnutí reinženýrovat *EDUBO* na stejný sestavovací nástroj. Tato změna se uskutečnila až ve fázi údržby, ale přinesla mi rychlejší zpětnou vazbu při opravách chyb v aplikaci.

React byl nasazen jako build (sestavená aplikace), který poskytuje optimalizovaný a produkční kód pro rychlé načítání aplikace v prohlížeči. Tento build byl dále servírován pomocí reverzního proxy serveru NGINX, což zajišťovalo rychlou a bezpečnou distribuci frontendového kódu uživatelům.

## Backend

Backendová část aplikací *EDUBO* a *Eduklub – plány výuky* byla navržena s ohledem na rozdílné požadavky obou projektů, což vedlo k použití různých frameworků. Zatímco *EDUBO* bylo vytvořeno pomocí mikro-frameworku Flask, pro *Eduklub – plány výuky* bylo zvoleno robustnější Django. Tento rozdíl reflektuje specifické potřeby každého projektu a zkušenosti získané během vývoje.

### EDUBO: Flask

Flask byl vybrán pro *EDUBO* především díky své jednoduchosti, rychlému nastavení a zkušenostem týmu. Flask je mikro-framework v Pythonu, který umožňuje vývojářům flexibilně přizpůsobit architekturu projektu jejich potřebám.

Mezi klíčové výhody Flasku patří:

- **Flexibilita:** Flask poskytuje svobodu v návrhu projektu, protože nevyžaduje pevnou strukturu.
- **Nízká složitost:** Ideální pro menší aplikace nebo prototypy, kde není potřeba složitá logika nebo vestavěné funkce.
- **Kompatibilita s rozšířeními:** Umožňuje snadno integrovat nástroje a knihovny třetích stran podle aktuálních potřeb projektu.

Nicméně během vývoje se objevily i významné nevýhody:

- **Chybějící migrace:** Nutnost manuálních úprav databázové struktury zpomalovala vývoj a zvyšovala pravděpodobnost chyb.
- **Základní správa uživatelů:** Omezené možnosti nativní autentizace a správy uživatelských práv si vyžádaly dodatečnou práci.
- **Absence jasné struktury:** Flexibilita může být nevýhodou u větších projektů, kde chybí standardizovaný přístup k organizaci kódu.

Flask byl vhodnou volbou pro rychlý začátek vývoje, avšak při nárůstu komplexity projektu se ukázalo, že by robustnější framework zjednodušil dlouhodobou údržbu.

### Eduklub – plány výuky: Django

Pro *Eduklub – plány výuky* bylo zvoleno Django, což je plnohodnotný framework v Pythonu, který nabízí rozsáhlé vestavěné funkce a jasnou strukturu. Django se osvědčilo zejména u projektů, které vyžadují vyšší míru komplexnosti a robustní řešení.

Mezi klíčové výhody patřily:

- **Komplexní nástroje:** Vestavěná podpora pro migrace, autentizace, správu uživatelských práv a ORM výrazně zjednodušila vývoj.
- **Standardizace:** Jasná a přehledná struktura projektu usnadnila spolupráci v týmu a přehlednost kódu.
- **Rychlosť vývoje:** Vestavěné administrativní rozhraní a šablonovací systém umožnily rychlé nasazení nových funkcí.

Na druhou stranu, Django má i své nevýhody:

- **Nižší flexibilita:** Pevně daná struktura a pravidla mohou být omezením pro projekty s netypickými požadavky.
- **Komplexnost:** U menších projektů může být Django zbytečně složité, což může zpomalit vývoj při nízkých náročích.
- **Vyšší počáteční nároky:** Vzhledem k většímu množství funkcí a možností vyžaduje Django delší dobu na naučení a nastavení.

Django bylo jasnou volbou pro tuto platformu, díky své schopnosti efektivně řešit komplexní potřeby projektu, což by s Flaskem nebylo možné bez výrazně vyššího úsilí.

### Srovnání Flaska a Djanga

Zatímco Flask je ideální pro menší aplikace, kde je potřeba rychle začít vývoj a přizpůsobit strukturu projektu, Django vyniká u větších projektů svou robustností a bohatou sadou vestavěných nástrojů.

Obě technologie přinášejí výhody v závislosti na konkrétních požadavcích projektu a zkušenostech vývojového týmu.

Oba backendy byly nasazeny jako samostatné služby v kontejnerech pomocí Dockeru. Tyto kontejnery byly spravovány pomocí Docker Compose, což umožnilo snadnou integraci s dalšími částmi systému, jako je databáze a frontend.

## Databáze

Pro databázovou vrstvu byly v projektech *EDUBO* a *Eduklub – plány výuky* použity různé přístupy reflektující jejich specifické požadavky a potřeby. V systému *EDUBO* byla zvolena MongoDB, zatímco ve webovém agregátoru byla použita relační databáze PostgreSQL.

### **EDUBO: MongoDB**

MongoDB byla původně vybrána díky své flexibilitě při ukládání nestrukturovaných dat. Její schopnost pracovat s dokumenty ve formátu JSON, které jsou interně ukládány ve formátu BSON (binární JSON), umožnila snadné ukládání nestrukturovaných dat z frontendu a efektivnější práci s daty díky optimalizovanému ukládání a rychlému přístupu.

Hlavní výhody MongoDB zahrnují:

- **Flexibilní datový model:** Umožnil snadné ukládání různorodých dat bez nutnosti předem definovaných schémat.
- **Škálovatelnost:** MongoDB nabízí jednoduché horizontální škálování.

Nicméně během vývoje se ukázaly i významné nevýhody:

- **Chybějící relace:** Komplikovalo to práci s datovými vazbami, což vedlo k potřebě složitějších dotazů a manuálního zpracování referencí.
- **Manuální úpravy struktury:** Kvůli absenci migrací bylo nutné měnit databázovou strukturu ručně při každé aktualizaci, což zvyšovalo riziko chyb.

### **Eduklub – plány výuky: PostgreSQL**

Pro *Eduklub – plány výuky* byla zvolena PostgreSQL relační databáze, která lépe odpovídala potřebám rozsáhlejší aplikace a zjednodušila práci s datovými vztahy.

Mezi její hlavní výhody patří:

- **Podpora relací:** Relace mezi tabulkami umožnily efektivní a přehledné ukládání datových vazeb.
- **Stabilita a výkon:** PostgreSQL je známá svou spolehlivostí a schopností pracovat s velkými objemy dat.
- **Vestavěná podpora pro migrace:** Díky integraci s Djangem byla práce s databází výrazně efektivnější.

## NGINX a Docker

Pro správu a nasazení obou systémů byly využity nástroje NGINX jako reverzní proxy server a Docker pro kontejnerizaci všech aplikací a jejich komponent.

### NGINX

NGINX byl použit k optimalizaci síťové komunikace mezi frontendem, backendem a uživateli.

Hlavní úloha NGINX spočívala v:

- **Směrování požadavků:** NGINX přesměrovává API požadavky na backend a ostatní požadavky na frontend, čímž zajišťuje efektivní distribuci zatížení.
- **Bezpečnostní filtrování:** Chrání aplikace před nežádoucími požadavky a útoky.
- **Zrychlení načítání dat:** Ve fázi údržby byl přidán kompresní nástroj Brotli, který umožnil odesílání statických souborů (např. JavaScript, CSS nebo obrázky) v co nejmenší velikosti. Statické soubory byly předem komprimovány na nejvyšší možnou úroveň, což výrazně snížilo množství přenášených dat a zlepšilo uživatelský zážitek díky rychlejšímu načítání.

### Docker

Docker je klíčovým nástrojem pro kontejnerizaci všech komponent aplikace, včetně frontendu, backendu, databáze a NGINX.

Hlavní přínosy Dockeru zahrnují:

- **Izolace prostředí:** Každá část aplikace běží v odděleném kontejneru, což eliminuje konflikty závislostí mezi různými komponentami.
- **Snadné nasazení:** Docker Compose umožňuje definovat konfiguraci všech kontejnerů v jednom souboru, což zjednodušilo jejich spuštění na různých prostředích.
- **Přenositelnost:** Docker zajistil, že aplikace může být spuštěna na libovolném serveru s minimální konfigurací.

Technický zásobník použitý v aplikacích *EDUBO* a *Eduklub – plány výuky* byl vybrán s ohledem na specifické požadavky obou projektů, zkušenosti týmu a potřebu efektivního vývoje. Monolitická architektura a kontejnerizace pomocí Dockeru zajistily jednoduché nasazení a správu aplikací, zatímco React pro frontend poskytl moderní a dynamické uživatelské rozhraní. Ačkoliv původní volba Flasku a MongoDB pro *EDUBO* nebyla ideální a přinesla komplikace spojené s absencí robustních nástrojů a relací, zkušenosti získané během vývoje vedly k použití Django a PostgreSQL v *Eduklub – plány výuky*, což se ukázalo jako efektivnější řešení pro komplexnější aplikace.

## 4.3. Implementace softwaru pro plánování výuky

Během vývoje *EDUBO* bylo nutné navrhnout a implementovat robustní systém pro tvorbu a organizaci výukových plánů. Celý proces byl rozdělen do dvou iterací, přičemž každá z nich se zaměřovala na konkrétní funkcionality systému. Mezi hlavní cíle patřilo vytvoření editoru výukových plánů, který umožní učitelům intuitivně sestavovat své plány, dále vyvinout systém pro publikaci výukových hodin, díky kterému budou mít studenti snadný přístup ke svým rozvrhům a možnosti exportu výukových plánů do PDF. Vývoj probíhal ve spolupráci se společnostmi DataPLEX Consulting, s.r.o. a Databig s.r.o., které poskytly odborné konzultace a zpětnou vazbu k implementovaným řešením.

### Vývojový proces a týmová spolupráce

V první iteraci byl vytvořen editor výukových plánů, který poskytuje učitelům nástroje pro snadné sestavování výukových plánů. Druhá iterace se soustředila na zpřístupnění hodin studentům, což zahrnovalo jejich publikaci a správu v uživatelském rozhraní.

Na vývoji se podílel tým studentů Univerzity Jana Evangelisty Purkyně a Gymnázia Josefa Jungmanna v Litoměřicích, přičemž role jednotlivých členů se postupně měnily podle aktuálních potřeb projektu. Projekt byl řízen agilní metodikou Scrum, která umožnila iterativní vývoj, pravidelné testování a rychlou reakci na zpětnou vazbu.

- **Pavel Beránek** byl projektovým manažerem v první iteraci a zajišťoval řízení projektu, komunikaci se stakeholders a organizaci vývojových aktivit. Ve druhé iteraci tuto roli převzal **Lukáš Polidar**.
- **Matěj Kaška (já)** byl hlavním backend programátorem v první iteraci a od druhé iterace převzal i roli hlavního programátora a Scrum mastera, zajišťujícího vedení týmu a organizaci sprintů.
- Na frontendu pracoval v první iteraci **Jan Plechatý**, který byl hlavním frontend programátorem. Spolu s ním se na vývoji podíleli **Vojtěch Kunc** a **Vlasta Michalcová**, která byla součástí týmu během všech tří iterací. Od druhé iterace převzal roli hlavního frontend programátora **Lukáš Priban**.

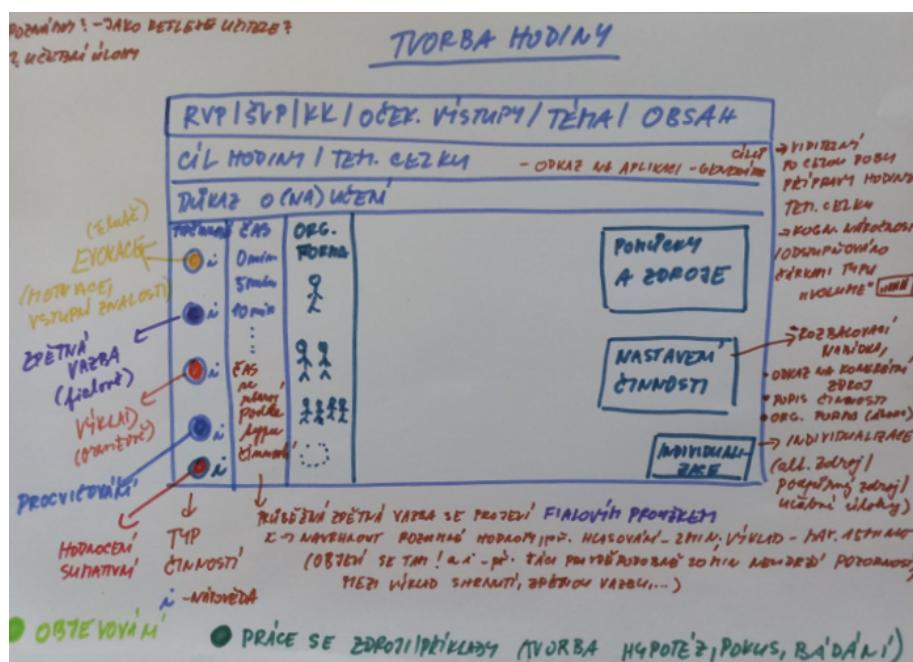
- **Andrej Dunda** a **Tomáš Svoboda** působili jako juniorní programátoři, přičemž Andrej Dunda se zaměřoval na frontend a Tomáš Svoboda měl na starosti testování a vývoj backendu i frontendu.
  - Návrh uživatelského rozhraní a UX designu v prvních dvou iteracích měla na starosti **Kateřina Pazderová**.
  - Na backendu pracoval kromě mě také **Lukáš Jurásek**, kterého ve druhé iteraci nahradil **Jakub Kopecký** jako hlavní backend programátor.

Komunikace v týmu probíhala prostřednictvím **Discordu**, kde se konaly pravidelné **stand-up meetingy**, během nichž se hodnotil pokrok a plánovala se následující iterace. Pro řízení úkolů byla využívána **Kanban tabule v ClickUpu**, což umožňovalo efektivní organizaci práce.

Pro verzování kódu byl využíván software **Git** ve spojení s **GitHubem**, kde jsme dodržovali standardizované pojmenovávání větví a workflow zahrnující pull requesty a issues. Každá nová funkčionalita byla vyvíjena v samostatné větvi (např. `feature/xxx`, `fix/xxx`), která byla následně schválena pomocí **code review** před sloučením do hlavní vývojové větve.

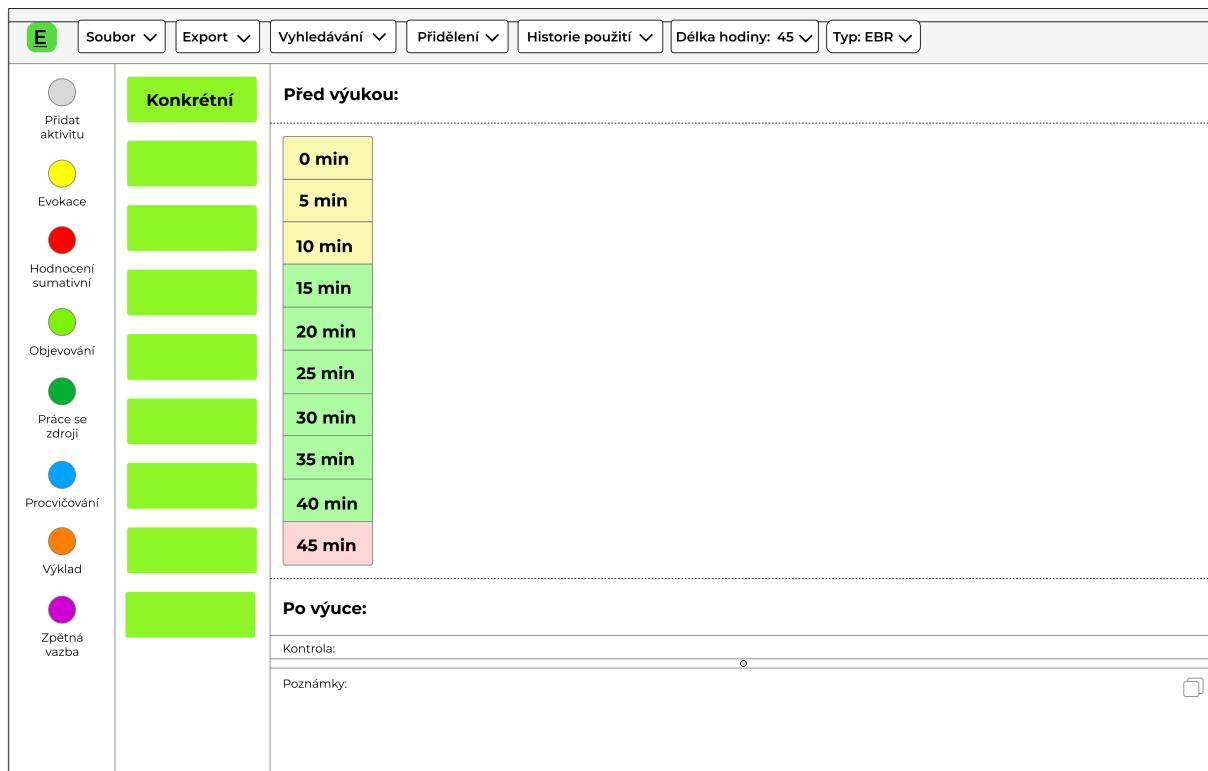
# Návrh uživatelského rozhraní ve Figma

Vývoj uživatelského rozhraní aplikace začal ručně kresleným wireframem (drátěný model), který vytvořili pedagogové Univerzity Karlovy. Tento návrh na papíře sloužil jako prvotní koncept aplikace, ve kterém se objevily základní požadavky na funkčnost editoru výukových plánů. I když měl tento návrh jasně definovanou strukturu, většina komponent byla v průběhu dalšího vývoje odstraněna nebo přeupravována. Zachovalo se však základní rozvržení plátna.



Obrázek 4.8.: Prvotní ručně kreslený návrh uživatelského rozhraní editoru aplikace EDUBO.

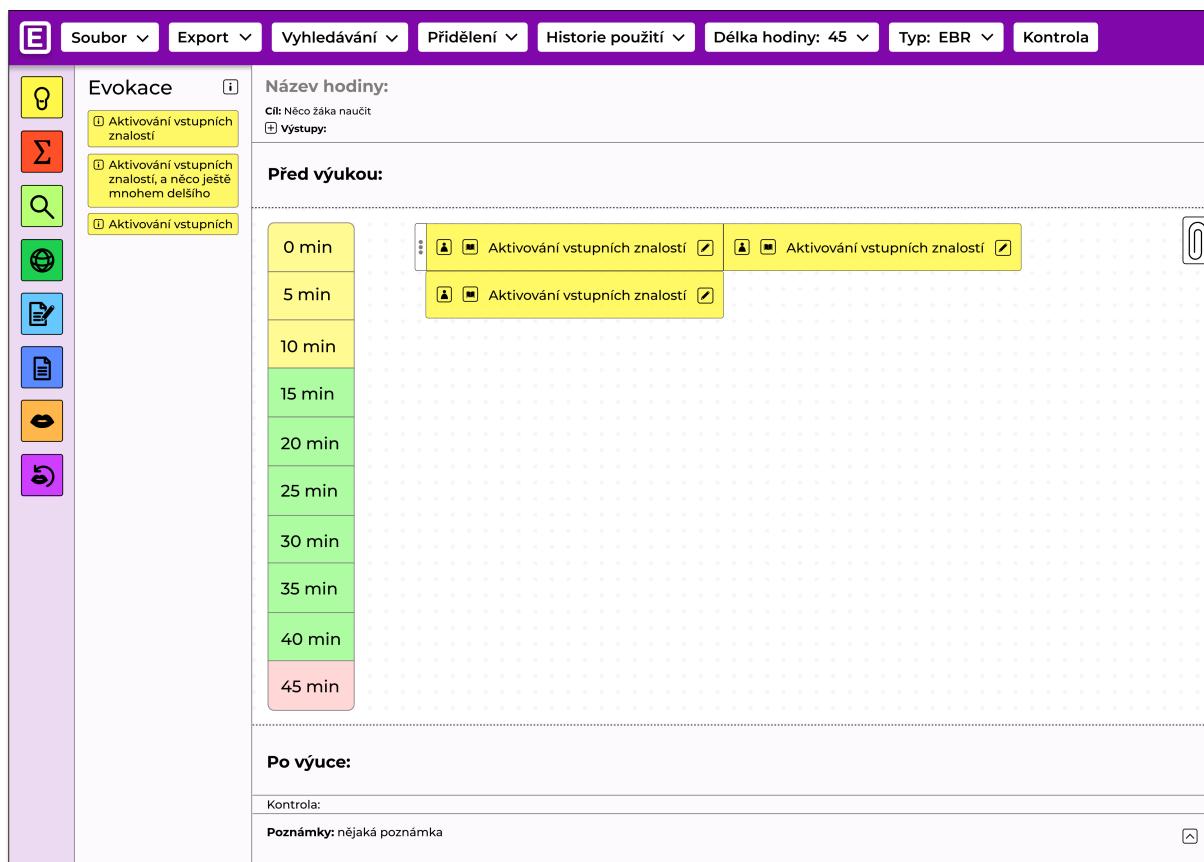
Na základě tohoto návrhu vznikla první digitální verze v aplikaci **Figma**, kterou vytvořila **Kateřina Pazderová**. Tento návrh se již více podobal finálnímu vzhledu aplikace, ale stále zde chyběla definitivní vizuální identita – například nebyla určena primární barva uživatelského rozhraní. V tomto návrhu měly typy aktivit kruhový design a zahrnoval několik funkcí, které nakonec nebyly součástí editoru, ale některé z nich našly své uplatnění v jiných částech aplikace. Mezi tyto komponenty patřilo například vyhledávání, přidělování hodin, historie použití a funkce pro kontrolu plánu a typ plánu.



Obrázek 4.9.: Druhý návrh uživatelského rozhraní editoru aplikace *EDUBO*.

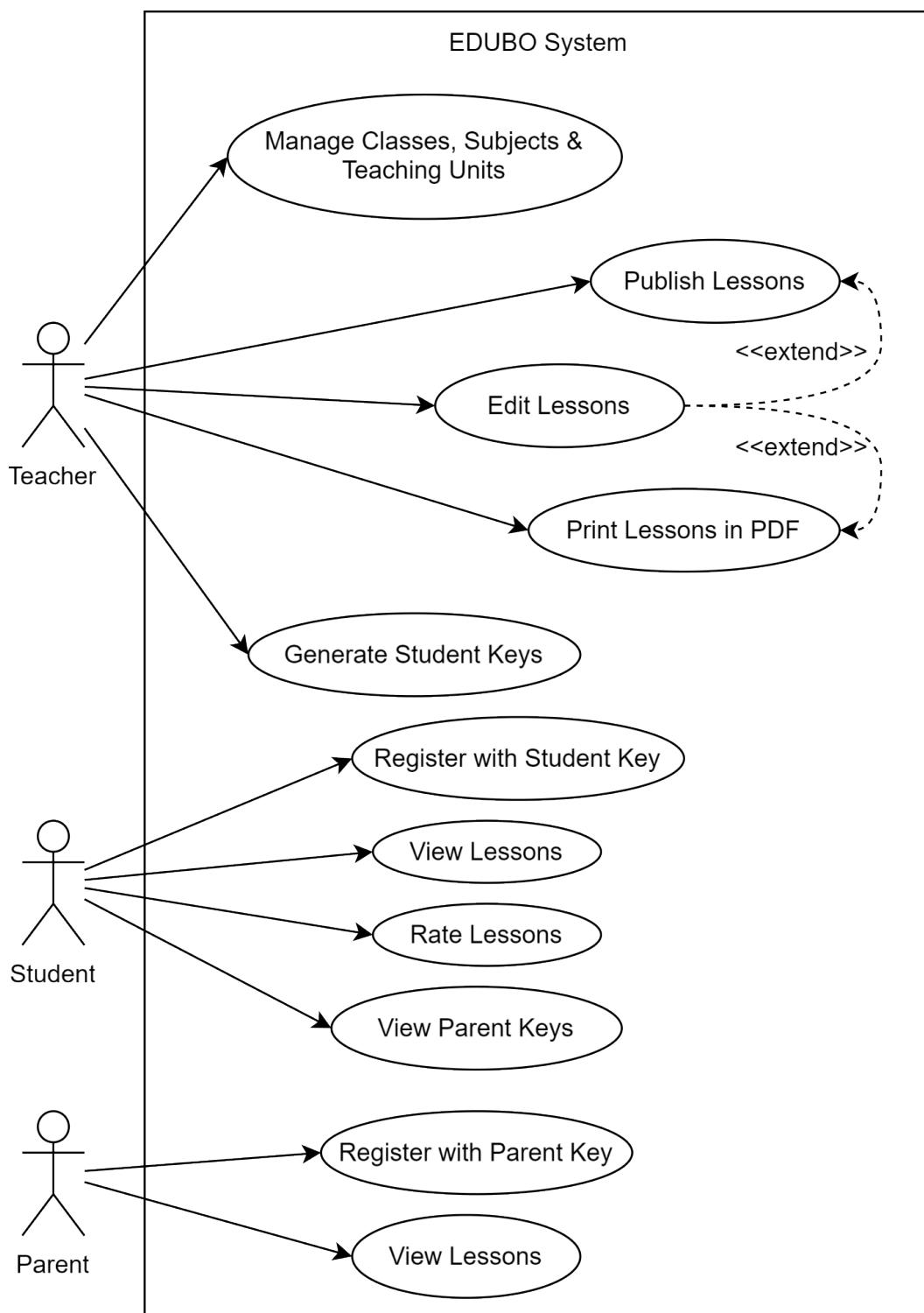
Třetí návrh ve Figma již téměř odpovídal finální podobě aplikace. V této fázi byla určena hlavní barva uživatelského rozhraní a vznikl přesný návrh plátna editoru. Tento návrh sloužil jako základ pro implementaci grafického rozhraní v samotné aplikaci.

#### 4. Praktická část



Obrázek 4.10.: Finální návrh uživatelského rozhraní editoru aplikace EDUBO.

## Modelování případů užití a procesů



Obrázek 4.11.: Diagram případů užití ilustrující základní scénáře interakce uživatelů se systémem.

Tento *Use case diagram* popisuje hlavní případy užití (*use cases*) pro tři různé typy uživatelů systému: učitele, studenta a rodiče. Diagram zobrazuje hlavní způsoby, jak bude systém využíván uživateli. Tyto případy zahrnují správu tříd, výukových jednotek, registraci uživatelů pomocí klíčů a možnost hodnocení hodin.

#### Teacher (Učitel):

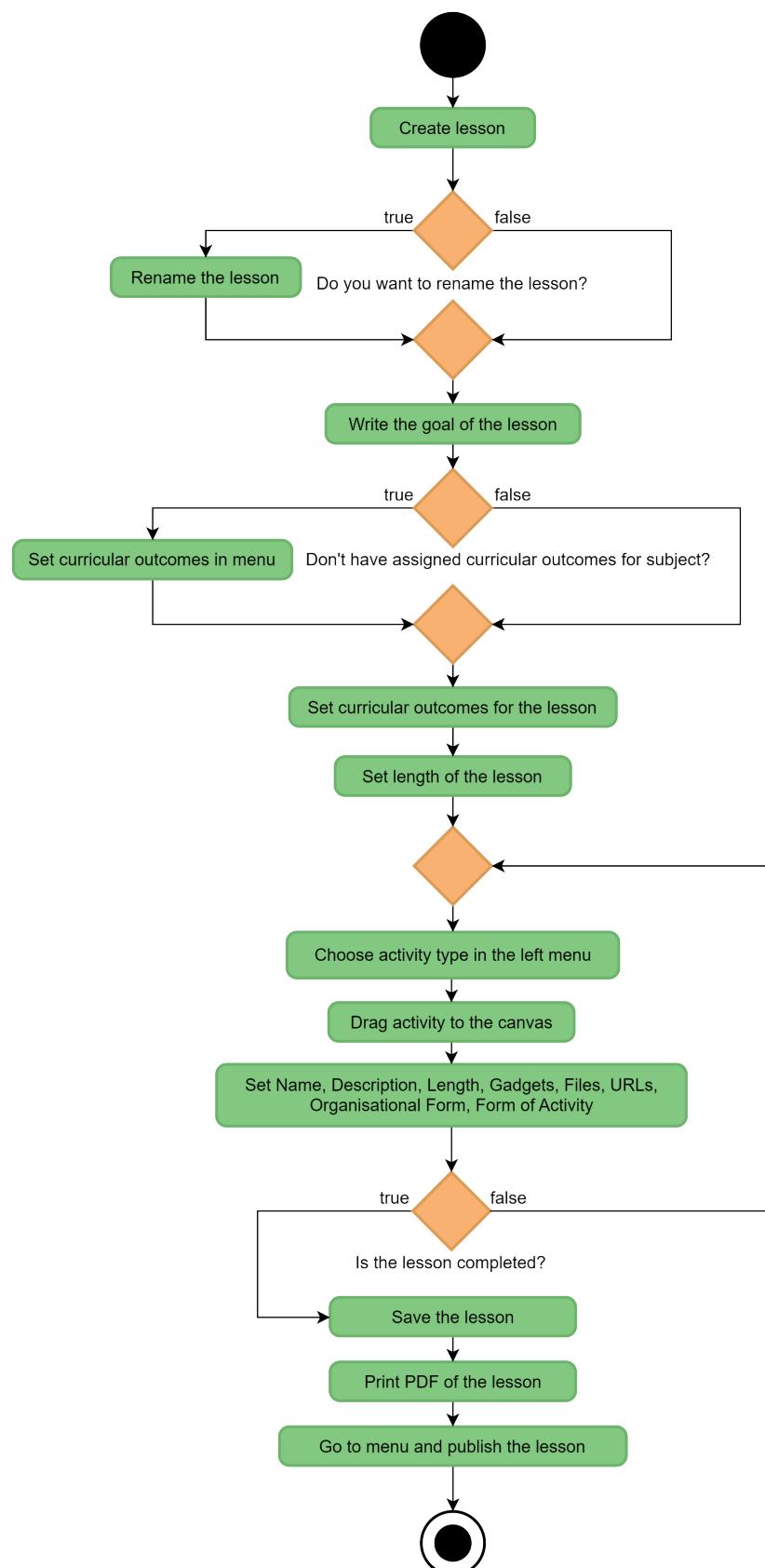
- **Manage Classes, Subjects & Teaching Units:** Učitel může spravovat třídy, předměty a výukové jednotky v systému.
- **Edit Lessons:** Učitel má možnost upravovat hodiny. Tento případ užití je propojen s dalšími dvěma funkcemi:
  - **Publish Lessons:** Učitel může publikovat hodiny jako volitelné rozšíření úprav hodin.
  - **Print Lessons in PDF:** Učitel má možnost vytisknout hodiny ve formátu PDF jako volitelné rozšíření úprav hodin.
- **Generate Student Keys:** Učitel může generovat registrační klíče pro studenty, které umožní studentům přihlášení do systému.

#### Student (Student):

- **Register with Student Key:** Student se může zaregistrovat do systému pomocí registračního klíče, který obdrží od učitele.
- **View Lessons:** Student má přístup k prohlížení publikovaných hodin.
- **Rate Lessons:** Student může hodnotit hodiny.
- **View Parent Keys:** Student může získat klíče pro rodiče, což jim umožní přístup k zobrazení hodin.

#### Parent (Rodič):

- **Register with Parent Key:** Rodič se může zaregistrovat do systému pomocí klíče, který obdržel od studenta.
- **View Lessons:** Rodič může zobrazit publikované hodiny, které byly přiřazeny jejich studentovi, což jim umožňuje sledovat průběh výuky.



Obrázek 4.12.: Diagram aktivit znázorňující postup při tvorbě a publikaci výukového plánu v aplikaci *EDUBO*.

Tento *diagram aktivit* ilustruje proces tvorby hodiny v aplikaci *EDUBO*. Diagram zobrazuje jednotlivé kroky od vytvoření hodiny až po publikaci hodiny pro studenty. Proces zahrnuje

nastavení základních parametrů hodiny, přidání aktivit a úpravy jejich detailů, a nakonec uložení a publikaci vytvořené hodiny.

**1. Vytvoření hodiny:**

- Proces začíná akcí *Create lesson*, kdy uživatel vytvoří výukovou hodinu v menu.

**2. Nastavení základních informací o hodině:**

- Uživatel má možnost hodinu přejmenovat *Rename the lesson*.
- Následně může definovat cíl hodiny pomocí akce *Write the goal of the lesson*.
- V případě, že nejsou přiřazeny kurikulární výstupy pro daný předmět, může uživatel výstupy nastavit přes *Set curricular outcomes in menu*.
- Poté se nastaví délka trvání hodiny pomocí akce *Set length of the lesson*.

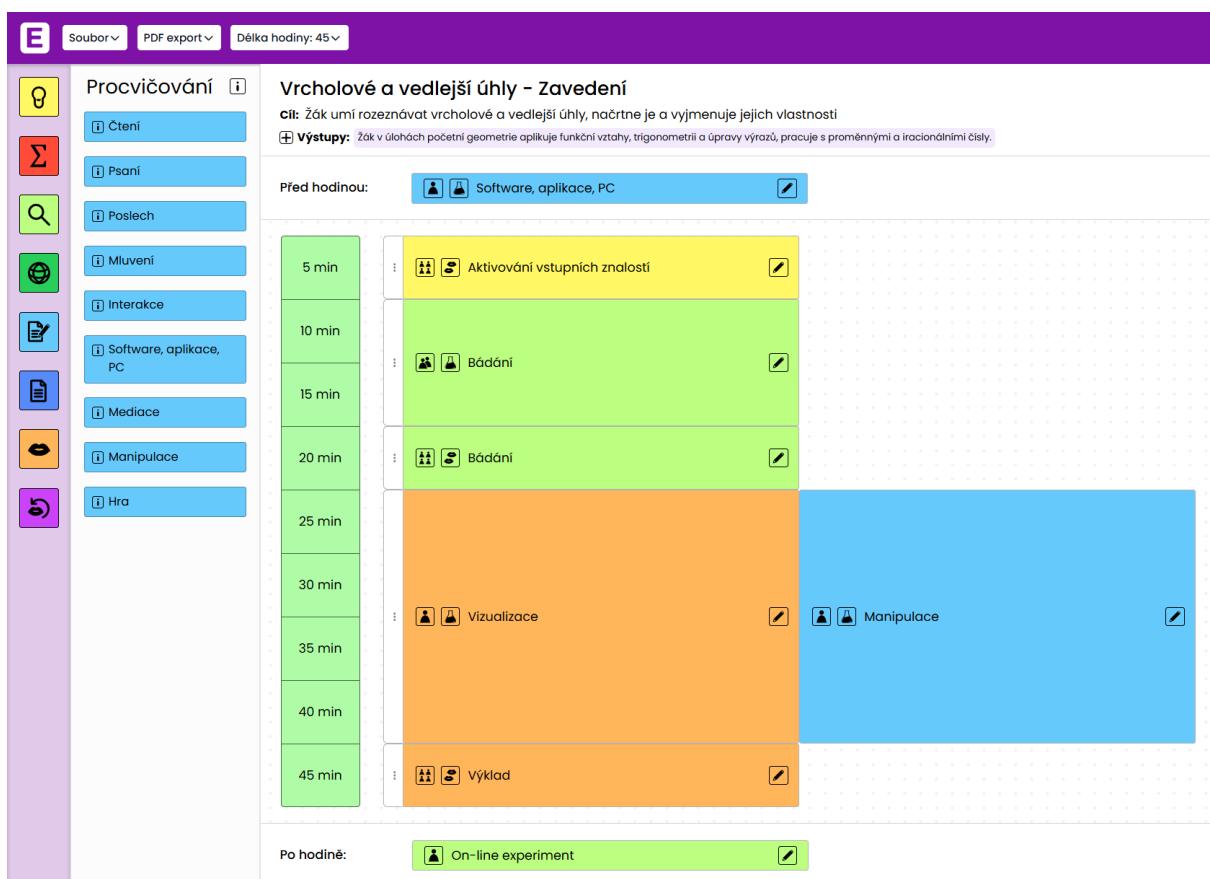
**3. Přidávání aktivit do hodiny:**

- Uživatel vybere typ aktivity z levého menu *Choose activity type in the left menu*.
- Uživatel vybranou aktivitu přetáhne na plátno pomocí drag and drop interakce *Drag activity to the canvas*.
- Uživatel upraví detaily aktivity, jako je název, popis, délka, pomůcky, soubory, odkazy, organizační formy a formy aktivity *Set Name, Description, Length, Gadgets, Files, URLs, Organisational Form, Form of Activity*.
- Tento krok se opakuje, dokud hodina není kompletně zaplněná.

**4. Dokončení a publikace hodiny:**

- Po dokončení všech úprav uživatel hodinu uloží *Save the lesson*.
- Hodina je připravena ve formátu PDF k tisku pomocí akce *Print PDF of the lesson*.
- Nakonec uživatel přejde do menu a publikuje hodinu pro studenty pomocí akce *Go to menu and publish the lesson*.

## Editor výukových plánů



Obrázek 4.13.: Okno aplikace EDUBO s náhledem na editor výukového plánu.

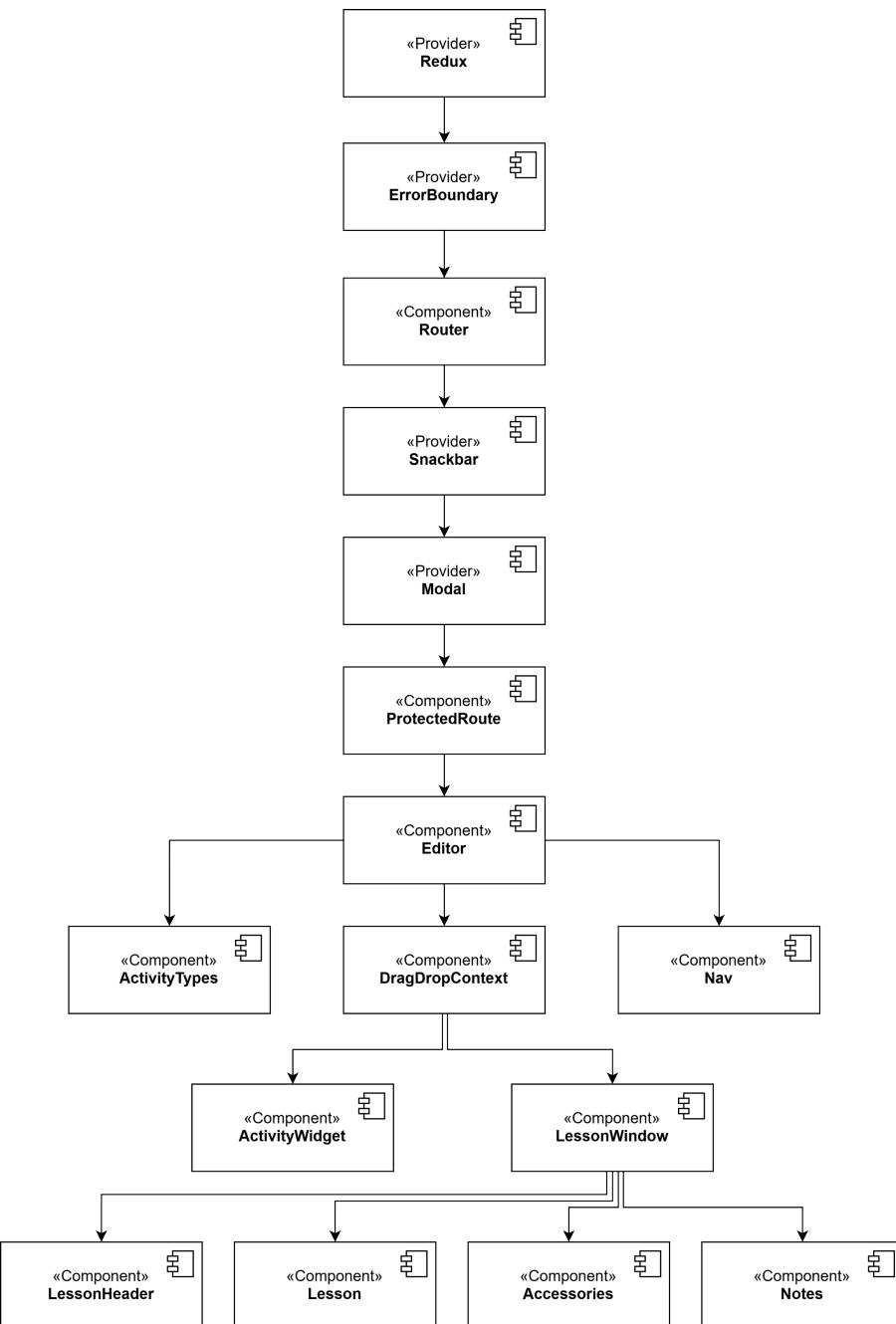
Editor výukových plánů je klíčovým prvkem softwaru pro plánování výuky, který učitelům umožňuje intuitivně sestavit strukturovaný plán hodiny. Uživatelské rozhraní editoru je přehledné a usnadňuje práci s výukovými aktivitami. V horní části se nachází navigační lišta s rozbalovacími seznamy, které umožňují manipulaci s plánem výuky, export do PDF nebo úpravu délky hodiny.

Na levé straně rozhraní jsou k dispozici různé typy aktivit, mezi ně patří evokace, hodnocení, objevování, zdroje, procvičování, shrnutí, výklad a zpětná vazba. Každý z těchto typů obsahuje další specifické aktivity, které byly vytvořeny ve spolupráci s Pedagogickou fakultou Univerzity Karlovy pod vedením proděkana doc. Antonína Jančářka. K jednotlivým aktivitám jsou připojená informační tlačítka, která učitele přesměrují na podrobné metodické materiály s návody na jejich efektivní využití ve výuce.

Střední část editoru tvoří hlavní plátno, na kterém učitel vytváří časovou strukturu výuky. Plátno je rozděleno podle časových bloků, do nichž lze pomocí *drag and drop* interakce přetahovat vybrané aktivity. Každou aktivitu lze následně editovat, upravit její název, popis či délku a přidat k ní potřebné pomůcky, soubory nebo odkazy. Důležitou součástí je také možnost nastavení organizační formy, tedy zda bude aktivita realizována v rámci celé třídy, ve skupinách, ve dvojicích nebo individuálně. Kromě toho je možné určit formu činnosti, například mluvený projev, interakci, čtení, psaní, experiment nebo poslech.

Kromě hlavního plátna obsahuje editor i speciální sekce pro aktivity, které mají proběhnout před výukou nebo po ní. Tento prvek slouží například k evidenci úkolů, které si žáci mají připravit před hodinou nebo k záznamu reflexe po jejím skončení. Editor také podporuje paralelní aktivity, což znamená, že v jednom časovém úseku mohou probíhat různé činnosti současně. Tato funkce je užitečná například při dělení žáků do skupin, kdy každá skupina pracuje na odlišných úkolech.

Další důležitou součástí editoru je sekce věnovaná cílům a výstupům hodiny. Výstupy vycházejí z rámcového vzdělávacího programu (RVP), přičemž software umožňuje nejen jejich přizpůsobení, ale také nahrání vlastních výstupů pomocí CSV souboru.



Obrázek 4.14.: Komponentní diagram znázorňující strukturu React komponent v editoru výukových plánů.

Tento komponentní diagram znázorňuje strukturu hlavních React komponent v Editoru výukových plánů. Nejvyšší úroveň tvoří globální poskytovatelé (Providers), kteří zajišťují sdílení stavů a funkcionalit napříč aplikací. Redux spravuje globální stav aplikace, ErrorBoundary se stará o zachytávání chyb, zatímco Snackbar a Modal poskytují uživatelské notifikace a modální okna. Router řídí navigaci mezi stránkami a ProtectedRoute zajišťuje, že k editoru mají přístup pouze oprávnění uživatelé.

Jádro editoru tvoří komponenta **Editor**, která spravuje interaktivní prvky uživatelského rozhraní. Pro správu typu aktivit je využita komponenta **ActivityTypes**. DragDropContext se stará o přetahování a organizaci prvků. Navigační menu je implementováno v komponentě **Nav**.

Další klíčové komponenty zahrnují **LessonWindow**, které představuje hlavní pracovní prostor editoru. V něm jsou obsaženy komponenty **LessonHeader** (obsahující název hodiny, cíle a výstupy), **Lesson** (hlavní struktura hodiny), **Accessories** (pomocné prvky, jako jsou soubory a pomůcky) a **Notes** (sekce pro dodatečné poznámky učitele). Komponenta **ActivityWidget** obsahuje jednotlivé výukové aktivity, které lze pomocí *drag and drop* interakce přetahovat do plátna editoru, kde jsou následně organizovány podle časové osy výukového plánu.

V dalších diagramových zobrazeních budou společné komponenty, jako jsou **Redux**, **ErrorBoundary**, **Router** a další globální poskytovatelé, vyjmuty, protože tvoří základní infrastrukturu celé aplikace a nejsou specifické pouze pro editor. Důraz bude kladen na detailnější rozdělení jednotlivých částí aplikace dle jejich konkrétní funkcionality.

## Výukový plán v PDF formátu

### EDUBO

Předmět: Matematika

Třída: 6. A

nkaskaj@gmail.com

Výukový celek: Úhly

### Vrcholové a vedlejší úhly - Zavedení

Cíl hodiny: Žák umí rozlišovat vrcholové a vedlejší úhly, načrtne je a vyjmenuje jejich vlastnosti

Výstupy:  Žák v úlohách početní geometrie aplikuje funkční vztahy, trigonometrii a úpravy výrazů, pracuje s proměnnými a iracionálními čísly.



### Pomůcky

Před výukou: GeoGebra

Během výuky: Vystrížené (a nejlépe zalaminované skládačky) na úhly (viz soubor/odkaz), Počítače/Notebooky/Tablety, Program GeoGebra, Kružítko, Pravítko, Úhloměr

Před  
výukou



### Software, aplikace, PC



Délka: 30 min

Popis: Žák se seznámí s programem GeoGebra a vyzkouší si na něm základní matematické nástroje, jako jsou rýsování geometrických útvarů, práce se souřadnicovým systémem, tvorba funkcí a jejich grafů. Naučí se využívat interaktivní prvky programu k objevování matematických vztahů a řešení úloh, což podpoří jeho prostorovou představivost a logické myšlení.

Pomůcky: GeoGebra

Odkazy:

<https://www.geogebra.org/>

5 min



### Aktivování vstupních znalostí



Popis: Co už víme o úhlech? Jaké známe typy úhlů? Jak úhel narýsujeme? Jak úhly graficky sčítáme a odčítáni? A jak je sčítáme, odčítáme, násobíme a dělíme numericky? Můžeme spolu vynásobit dva úhly? Nebo je spolu vydělit? A co se stane, pokud dělíme nulový úhel libovolným číslem? atd.

Obrázek 4.15.: Ukázka výukového plánu v PDF formátu.

Výukový plán lze exportovat do PDF formátu, což umožňuje učitelům mít snadný přístup k materiálům i bez připojení k internetu. Tento export je generován pomocí HTML šablonového systému Jinja2, který převádí data z aplikace do strukturovaného a čitelného formátu. Výstupní dokument obsahuje všechny důležité informace o hodině, včetně jejího cíle, výstupů, potřebných pomůcek a podrobného rozpisu výukových aktivit. Tento formát usnadňuje archivaci a distribuci plánů mezi učiteli i studenty.

```

1  {% if lesson['activities_list'] %}
2  {% set total_time = namespace(value=0) %}
3  {% for list_of_activities in lesson['activities_list'] %}
4  {% if list_of_activities['activities'] %}
5  {% set total_time.value = total_time.value + list_of_activities['
6  activities'][0]['size'] %}
7  {% set parallel_activities_count = list_of_activities['activities'].l
8  ength %}
9  {% endif %}
10 {% if loop.index == last_activity_row_index %}
11   {% set is_last_row = true %}
12 {% endif %}
13 {% for activity in list_of_activities['activities'] %}
14   <section class="activity">
15     <div class="grid grid-cols-24 gap-y-4 mt-12">
16       <div class="col-span-2 text-center text-lg font-semibold">
17         {% if is_last_row %}
18           {% if loop.index == 1 %}
19             <p>{{ total_time.value }} min</p>
20           {% endif %}
21           {% elif loop.index == 1 %}
22             <p>{{ total_time.value }} min</p>
23             <div
24               class="timeline {% if parallel_activities_count > 1 %}double{%
25                 endif %}"
26               style="height: calc(100% + 1.25rem); "></div>
27           {% elif loop.last %}
28             <div
29               class="timeline {% if parallel_activities_count > 1 %}double{%
30                 endif %}"
31               style="height: calc(100% + 3rem); "></div>
32           {% else %}
33             <div
34               class="timeline {% if parallel_activities_count > 1 %}double{%
35                 endif %}"
36               style="height: calc(100% + 3.5rem); "></div>
37             {% endif %}
38           </div>
39         </div>
40       </section>
41     {% endfor %}
42   {% endfor %}

```

38    {%- endif %}

Tento úryvek Jinja2 kódu slouží k vygenerování časové osy v PDF výukového plánu. Prochází seznam aktivit v hodině a postupně sčítá jejich časovou délku. Dynamicky vykresluje minutové značky a určuje, zda jsou aktivity v rámci jedné časové jednotky vykonávány paralelně. Pomocí podmínek nastavuje vzhled časové osy, například přidáním dvojité časové linie pro paralelní aktivity. Výstupem je přehledná vizualizace struktury hodiny se správným časovým rozvržením jednotlivých aktivit.

**40 min** **Vizualizace**

**Popis:** Žáci se přihlásí na počítačích na GeoGebu Geometrii a společně pomocí pokynů učitele provedou konstrukci vrcholových/vedlejších úhlů, tedy narýsuji si dvě různoběžky a zobrazí si hodnoty úhlů, které svírají. Následně je učitel vyzve, aby polohu původních přímek začali měnit (vytvořte si situaci, kterou jste sestavili na skládačce, mohou být všechny úhly stejně velké? A kdy pokud ano?, Co se stane, pokud budou přímky rovnoběžné?...)

**Pomůcky:** Počítače/Notebooky/Tablety, Program GeoGebra

**Manipulace**

**Popis:** Žáci si vezmou rýsovací potřeby (pravítko, úhloměr, kružítko) a na papír narýsuji dvě různoběžky. Pomocí úhloměru změří a zapiší velikosti vrcholových a vedlejších úhlů, které tyto přímky svírají. Učitel je provede diskusí o vzájemných vztazích mezi úhly a vyzve je k experimentování – mohou upravit polohu jedné z přímek a sledovat, jak se mění velikosti úhlů. Žáci si kladou otázky: Může být všech osm úhlů stejně velkých? Pokud ano, za jakých podmínek? Co se stane, pokud se přímky stanou rovnoběžnými? Výsledky své práce porovnají a společně formulují závěry o vlastnostech vrcholových a vedlejších úhlů.

**Pomůcky:** Kružítko, Pravítko, Úhloměr

**45 min** **Výklad**

**Popis:** Učitel formálně zavádí pojmy vrcholové a vedlejší úhly a rekapituluje s žáky, na jaké jejich vlastnosti v průběhu hodiny přišli.

Obrázek 4.16.: Ukázka paralelních aktivit ve výukovém plánu v PDF formátu.

```

1 <div class="col-start-3 col-span-2 flex justify-center">
2   
6 </div>
7 <div class="col-span-20">
8   <h1 class="dynamic-header text-xl font-bold">{{ activity.name }}</h1>
9   <div class="flex items-baseline my-2">
```

```

9  {% if 'icons' in activity %}
10 {% for icon in activity['icons'] %}
11   {% if icon != "" %}
12     
16   {% endif %}
17   {% endfor %}
18 {% endif %}
19 </div>
20 <div class="flex flex-wrap items-baseline space-x-1 mt-1">
21   {% if html_tags_remover(newlines_remover(activity['description'])) !=
22     "" %}
23     <p>
24       <span class="font-bold">Popis: </span>
25       {{ html_tags_remover(html_entity_replacer(activity.description)) }
26     }
27   </p>
28   {% endif %}
29 </div>
30 <div class="flex items-baseline space-x-1 mt-1">
31   {% if activity['gadgets'] and activity['gadgets']|reject('equalto', '')
32     |list %}
33     <span><h1 class="text-base font-semibold">Pomůcky:</h1></span>
34     <p>{{ empty_string_remover(activity['gadgets'])|join(', ') }}</p>
35   {% endif %}
36 </div>
37   {% set visibleurls = visibility_normalizer(activity['urls'], view)
38   %}
39   {% if visibleurls | length > 0 %}
40     <h1 class="text-base font-semibold mt-1">Odkazy:</h1>
41     {% for url in visibleurls %}
42       {% if url != "" %}
43         <p class="text-sm mt-1 underline">{{ url|urlize(target='_blank') }
44         }
45       </p>
46       {% endif %}
47     {% endfor %}
48   {% endif %}
49 </div>

```

Tento Jinja2 kód generuje jednotlivé sekce aktivit ve výukovém plánu PDF. Každá aktivita je vizuálně reprezentována pomocí ikony, názvu, popisu a případných pomůcek. Kód také dynamicky

načítá a zobrazuje seznam připojených odkazů, pokud jsou k dispozici. Všechny textové vstupy jsou ošetřeny proti nežádoucím HTML značkám a speciálním znakům, aby byl výstup čistý a přehledný.

## Uživatelské menu

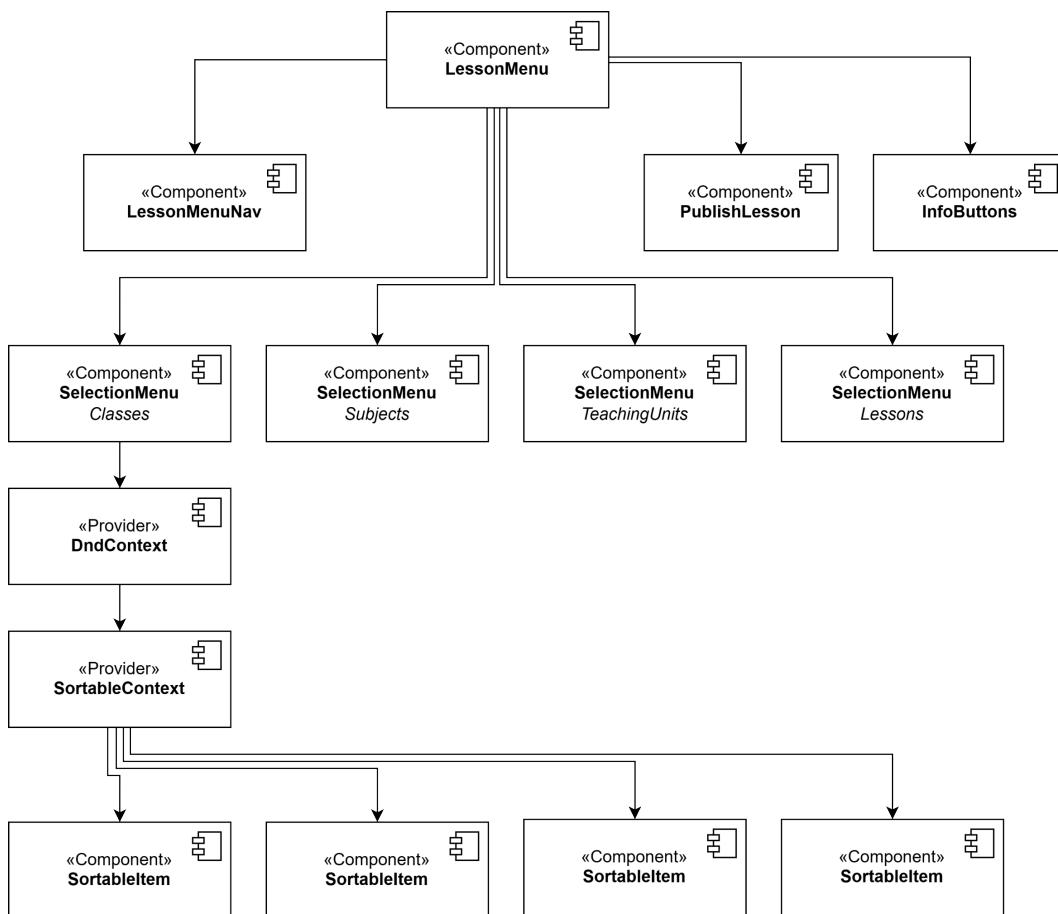
Obrázek 4.17.: Okno aplikace EDUBO s náhledem na uživatelské menu.

Uživatelské menu poskytuje pedagogům správu a publikaci výukových plánů. Uživatel může v navigační liště nahrávat výukové plány, výukové celky nebo soubory výukových celků ve formátu ZIP, přičemž je podporována integrace s webovým agregátorem výukových plánů. Dále je možné nahrávat vlastní kurikulární výstupy ve formátu CSV. Další funkcí je generování registračních klíčů, které studenty automaticky přiřadí do odpovídajících tříd. Menu také umožňuje změnit režim zobrazení – lineární režim, kde uživatel nejprve vybírá třídu a následně předmět, nebo dynamický režim, kde může být volba předmětu provedena před výběrem třídy.

Uživatelé mají přístup k seznamům tříd, předmětů, výukových celků a jednotlivých hodin, přičemž všechny tyto položky mohou přidávat, upravovat a mazat. Pro pohodlnou organizaci obsahu je možné využít drag and drop interakce, který usnadňuje řazení výukových položek. Při vytváření třídy lze vybrat z již existujících tříd a automaticky se tak uživatel přiřadí do dané třídy. V rámci editace předmětu může pedagog přiřadit již přednastavené kurikulární výstupy nebo nahrát vlastní. Výukové celky je možné doplnit o poznámky.

V pravé části menu se nachází informace o konkrétní hodině spolu s možností její publikace.

Uživatel může nastavit čas a datum zveřejnění, po kterém se daná hodina zobrazí v uživatelském rozhraní studentů.



Obrázek 4.18.: Komponentní diagram znázorňující strukturu React komponent v uživatelském menu.

Tento komponentní diagram popisuje strukturu React komponent uživatelského menu v aplikaci pro správu výukových plánů. Hlavní komponentou je `LessonMenu`, která zastřešuje celé rozhraní pro organizaci tříd, předmětů, výukových celků a plánů.

Komponenta `LessonMenuNav` poskytuje navigační prvky pro orientaci v menu. `PublishLesson` umožňuje publikaci výukových plánů a jejich zpřístupnění studentům. `InfoButtons` slouží k zobrazení tlačítek pro exportování nebo úpravu výukových plánů.

Struktura tříd, předmětů, výukových celků a plánů je spravována pomocí komponent `SelectionMenu`, které jsou rozděleny podle jednotlivých kategorií (`Classes`, `Subjects`, `TeachingUnits` a `Lessons`). Tyto komponenty umožňují uživateli vybírat a spravovat jednotlivé položky.

Pro podporu drag and drop interakce byly do menu integrovány providery `DndContext` a `SortableContext`, které umožňují přetahování a řazení prvků. Samotné přetahovatelné položky jsou reprezentovány komponentou `SortableItem`, která se stará o jejich správné zobrazení a interaktivitu v rámci menu.

```

1 const modalAddLessonCallback = async (oldValue: string) => {
2   const value = oldValue.trim();
3   if (!userData) return console.error("UserData are missing!");
4   if (value === "") return openErrorSnackbar("Hodina musí mít název!");
5   if (value.length > 63) return openErrorSnackbar("Název nesmí být delší
6     než 63 znaků!");
7
8
9   const indexes = getActiveItemsIndexes();
10  const lessons = newUserData
11    .classes[indexes.activeClassIndex]
12    .subjects[indexes.activeSubjectIndex]
13    .teaching_units[indexes.activeTeachingUnitIndex]
14    .lessons;
15
16  if (lessons.find((item) => (item.name === value)))
17    return openErrorSnackbar("Hodina se stejným názvem již existuje!");
18
19  const newLesson: iMenuLesson = {
20    _id: uuidv4(),
21    name: value,
22    published: false,
23    days_of_usage: []
24  };
25  lessons.push(newLesson);
26
27  await Promise.all([
28    updateClasses(newUserData.classes),
29    updateLesson(newLesson._id, newLesson.name)
30  ]);
31}

```

Tato funkce je určena k vytvoření nové výukové lekce. Jejím parametrem je řetězec představující název nové hodiny. Na začátku probíhá kontrola, zda zadaný název splňuje všechny potřebné podmínky, jako je nepřítomnost prázdného textu nebo překročení maximální délky. Následně se vytvoří kopie uživatelských dat, která je připravena pro další úpravy.

Funkce dále získá aktivní indexy, tedy index aktuálně vybrané třídy, předmětu a výukového celku. Na základě těchto indexů se vytvoří reference na příslušné pole výukových lekcí, kde bude nová výuková lekce přidána. Před přidáním se zkонтroluje, zda již v datech neexistuje výuková lekce se stejným názvem, aby nedošlo k duplikaci.

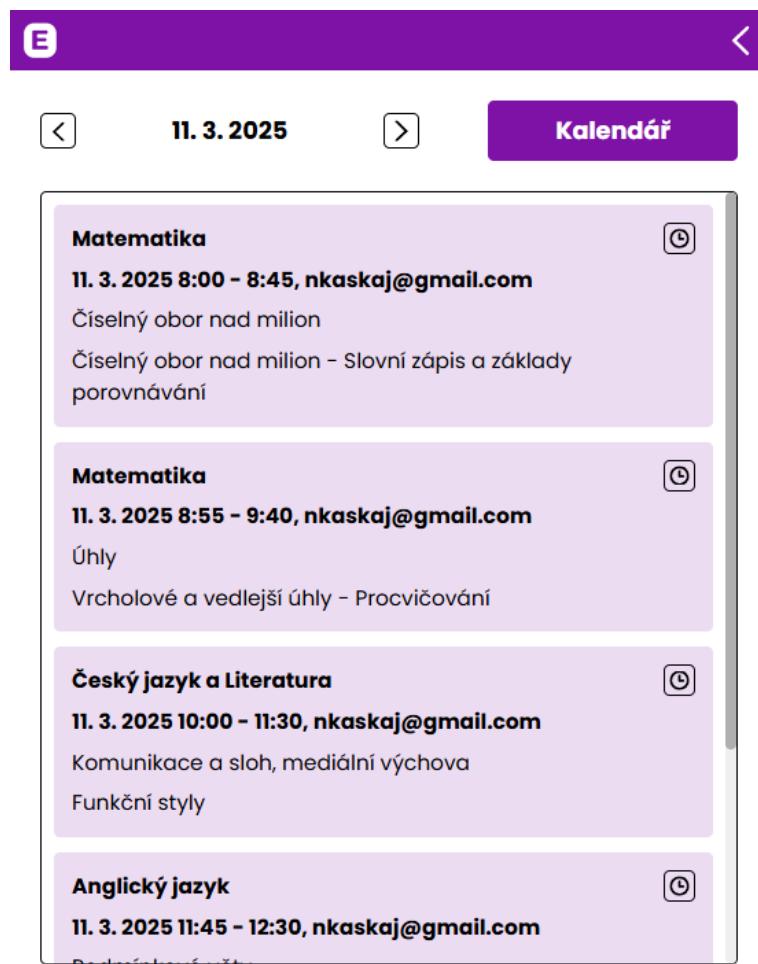
Pokud je kontrola úspěšná, vytvoří se objekt nové výukové lekce s unikátním identifikátorem,

názvem a výchozími hodnotami pro další vlastnosti. Tento objekt je následně přidán do pole výukových lekcí.

Na závěr funkce vytvoří **Promise** (JavaScriptový příslib), který zajistí paralelní odeslání dvou požadavků na backend. První požadavek aktualizuje uživatelská data v databázi, zatímco druhý vytvoří novou instanci hodiny v databázi. Tento postup zajišťuje synchronizaci mezi lokálními daty a databází a umožňuje správnou funkčnost aplikace.

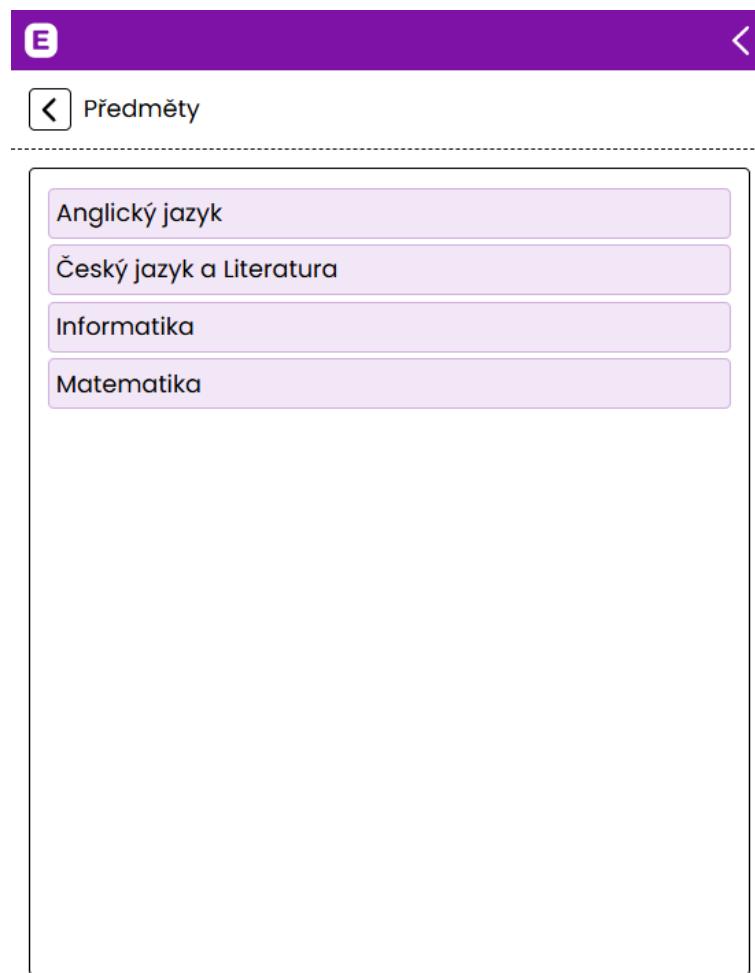
### Uživatelské rozhraní studenta

Uživatelské rozhraní studenta je navrženo tak, aby umožňovalo snadný přístup k výukovým plánům a organizaci výukových hodin. Studenti mají k dispozici dvě hlavní zobrazení – **Kalendář** a **Předměty**.



Obrázek 4.19.: Mobilní zobrazení aplikace *EDUBO* s náhledem na kalendář studenta.

V zobrazení **Kalendář** se zobrazují všechny naplánované výukové hodiny pro vybraný den. Každá hodina obsahuje informace o předmětu, čase, vyučujícím a tematickém zaměření výukové hodiny. Studenti si mohou přepínat mezi jednotlivými dny pomocí navigačních tlačítek nebo otevřít kalendář a zvolit libovolné datum, pro které chtějí zobrazit výuku.



Obrázek 4.20.: Mobilní zobrazení aplikace *EDUBO* s náhledem na předměty studenta.

Zobrazení **Předměty** poskytuje přehled všech předmětů, ke kterým má student přístup na základě publikovaných výukových hodin. Po kliknutí na konkrétní předmět se zobrazí výukové celky daného předmětu, které student může dále rozkliknout pro zobrazení jednotlivých výukových hodin.

**Vrcholové a vedlejší úhly - Zavedení**

**cíl:**

**Výstupy:**

**Před hodinou:**   Software, aplikace, PC

5 min	Aktivování vstupních znalostí
	<b>Bádání</b>
	<b>Délka:</b> 10 minut
	<b>Popis:</b> Každý žák (nebo dvojice) dostane jednu skládačku (samozřejmě rozloženou). Jejich úkolem je seskládat jeden obrázek tak, aby využili všechny díly a žádné dva se vzájemně nepřekrývali.
	<b>Pomůcky:</b> Vystřízené (a nejlépe zalaminované) skládačky na úhly (viz soubor/odkaz)
	<b>Soubory:</b> <a href="#">Vrcholove_vedlejsi_uhly_slozene.jpg</a> Fotografie skládačky
	<b>Odkazy:</b> <a href="https://drive.google.com/file/d/1KuohwVcEhLbWmTvnVoNT0Db_M48u7n0W/view?usp=share_link">https://drive.google.com/file/d/1KuohwVcEhLbWmTvnVoNT0Db_M48u7n0W/view? usp=share_link</a>

**Pomůcky**    **Uložené materiály**    **Poznámky**

Obrázek 4.21.: Mobilní zobrazení aplikace EDUBO s náhledem na výukový plán.

Zobrazení výukové lekce poskytuje studentům přehledný a interaktivní způsob, jak se seznámit s výukovým plánem dané výukové lekce. V horní části okna se nachází název výukové lekce, její cíle a očekávané výstupy. Tyto informace umožňují studentům pochopit hlavní zaměření výukové lekce a to, co se očekává, že se během ní naučí.

Pod tímto úvodním blokem se nachází samotný časový plán výukové lekce. Struktura je rozdělena do tří částí: **Před hodinou**, kde se nacházejí aktivity určené k přípravě před výukou, **Samotná hodina**, obsahující všechny naplánované činnosti, a **Po hodině**, která zahrnuje následné aktivity k upevnění učiva. Aktivity jsou vizuálně rozděleny podle délky trvání a po kliknutí na konkrétní aktivitu se rozbalí její detailní popis, včetně pomůcek, přiložených souborů a odkazů na online zdroje.

Ve spodní části obrazovky se nachází tři hlavní tlačítka. **Pomůcky** zobrazí seznam všech potřebných pomůcek pro danou výukovou lekci. **Uložené materiály** umožní přístup ke všem souborům a odkazům, které jsou součástí výuky. **Poznámky** poskytují doplňující informace k výukovému plánu, které mohou učitelé přidat pro lepší porozumění a přípravu na výukovou lekci.

## 4.4. Implementace webového agregátoru výukových plánů

Během vývoje webového agregátoru výukových plánů bylo nutné navrhnout a implementovat platformu, která umožní učitelům efektivně sdílet své výukové plány vytvořené v aplikaci *EDUBO*. Hlavním cílem bylo vytvořit intuitivní webovou aplikaci, která nabídne pokročilé vyhledávání a filtrování plánů podle různých kritérií, možnost jejich stažení a hodnocení. Důležitou součástí systému bylo také umožnit učitelům přidávat alternativní či upravené verze výukových plánů a tím podpořit spolupráci v pedagogické komunitě. Vývoj probíhal ve spolupráci se společností Mr. Cloud s.r.o., která poskytla odborné konzultace a podporu při návrhu architektury aplikace.

### Vývojový proces a týmová spolupráce

Třetí iterace projektu byla zaměřena na vývoj webového agregátoru výukových plánů, který umožňuje učitelům sdílet a spravovat výukové materiály. Tato část vývoje probíhala pod vedením projektového manažera Lukáše Polidara, který měl na starosti organizaci práce, komunikaci se stakeholders a koordinaci vývojového týmu. Roli Scrum mastera jsem v rámci týmu zastával já. Byl jsem zodpovědný za efektivní fungování týmu podle agilní metodiky Scrum, vedení sprintů a dohled nad dodržováním vývojového postupu.

Hlavní frontendový vývoj vedl Lukáš Priban, který koordinoval práci na uživatelském rozhraní aplikace. Spolu s ním na frontendu pracovali Andrej Dunda, Vlasta Michalcová a Tomáš Svoboda, který měl zároveň na starosti testování aplikace. Backendovou část vývoje měl na starosti hlavní backend programátor Jakub Kopecký, který zodpovídal za návrh a implementaci aplikační logiky a komunikaci s databází. Na rozdíl od předchozích iterací nebyl UI/UX návrh (návrh uživatelského rozhraní a uživatelského zážitku) zpracován interně, ale byl vytvořen UI/UX designérem společnosti Mr. Cloud s.r.o., který připravil návrh uživatelského rozhraní ve Figma včetně funkčního prototypu.

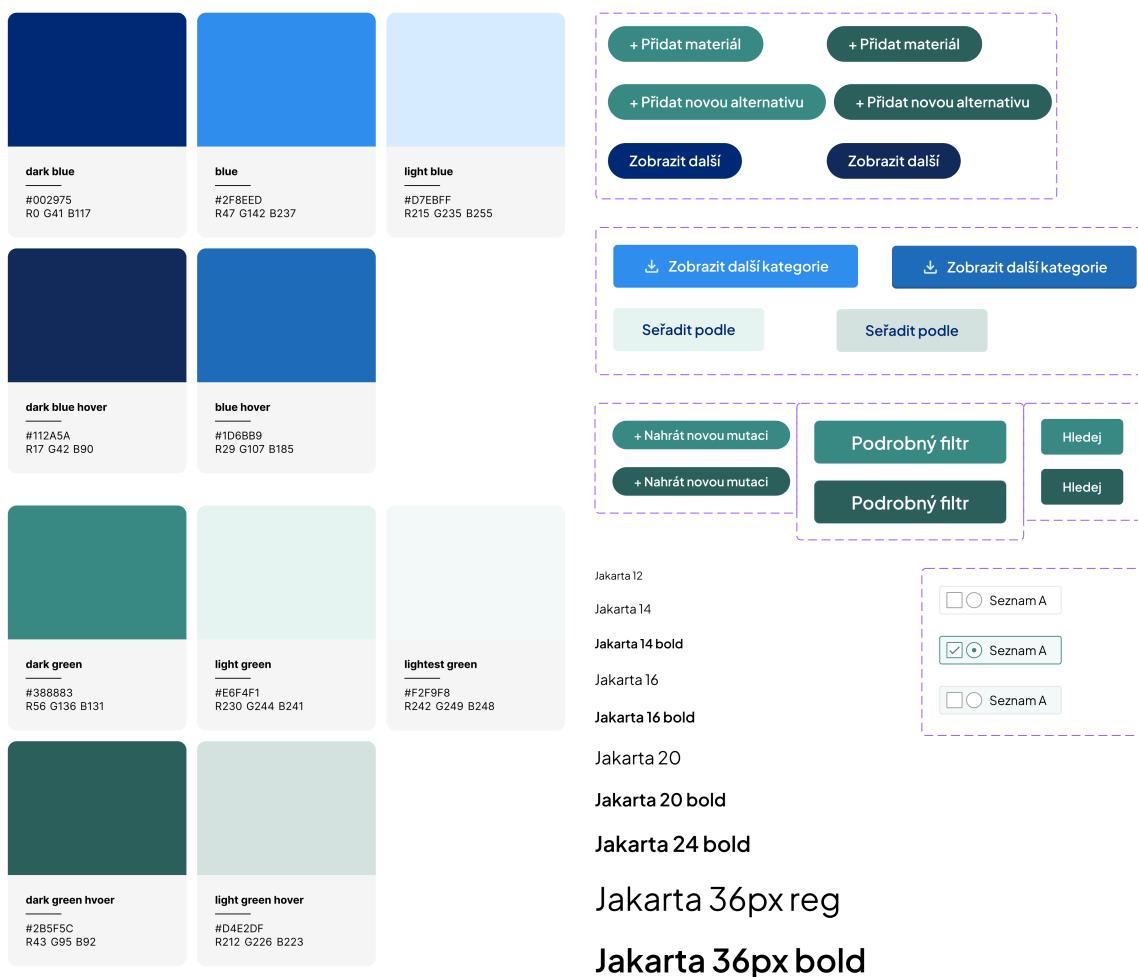
Metodika vývoje, komunikace a správa verzí zůstaly stejné jako v předchozích iteracích. Hlavním nástrojem pro komunikaci byl Discord, kde probíhaly pravidelné stand-up meetingy, hodnotící pokrok a plánující další kroky. Pro řízení úkolů a sledování postupu práce byla využívána Kanban tabule v ClickUpu. Verzování kódu probíhalo pomocí softwaru Git na platformě GitHub, kde bylo dodržováno standardizované pojmenovávání větví, pull requesty a code review.

### Návrh uživatelského rozhraní ve Figma

Návrh uživatelského rozhraní ve Figma byl klíčovým podkladem pro vývoj webového agregátoru výukových plánů. Byl nám předán na začátku třetí iterace a obsahoval funkční prototyp, což umožnilo interaktivní náhled na plánovanou podobu aplikace. Díky této funkcionalitě si vývojáři mohli rozhraní předem proklikat a otestovat, což pomohlo s lepším pochopením způsobu, jakým uživatelé interagují s aplikací, a usnadnilo implementaci.

Kromě samotného prototypu obsahoval návrh kompletní dokumentaci designu, tedy definované barvy, fonty, tlačítka a všechny potřebné UI komponenty. Tato strukturovaná dokumentace výrazně urychlila vývoj, protože umožnila nejdříve vytvořit základní vizuální komponenty a styly, které bylo následně možné znovu využít v celém projektu.

Přestože návrh ve Figma tvořil pevný základ, výsledná aplikace není identická s původním designem. Během vývoje se totiž objevily nové požadavky a funkcionality, které bylo nutné přidat nebo upravit. Některé části návrhu bylo proto nutné rozšířit nebo modifikovat, aby odpovídaly reálným potřebám uživatelů.



Obrázek 4.22.: Dokumentace návrhu uživatelského rozhraní v aplikaci Figma pro *Eduklub – plány výuky*.

The screenshot shows the 'Plány výuky' section of the Eduklub website. At the top, there's a search bar with a magnifying glass icon and the word 'Hledej' (Search). To the right are icons for user profile, heart, and shopping cart. Below the search bar, there are navigation links: 'Plány výuky', 'eduWiki', 'Odborné články', and a green button '+ Přidat materiál'. The main content area is titled 'Matematika první stupeň'. On the left, a sidebar contains filters for 'Typy' (checkboxes for Celky, Bloky, Hodiny), 'Hodnocení' (checkboxes for Fantasticé: 5 hvězd, Výborné: více než 4 hvězd, Dobré: více než 3 hvězd, Uspokojivé: více než 2 hvězd), 'Počet recenzí' (a slider from 0 to 100), 'Určeno pro' (checkboxes for První stupeň, Druhý stupeň), 'Inkluzivní výuka' (checkboxes for Ano, Částečně, Ne), 'Certifikace' (checkboxes for Jen certifikované, Bez certifikátu), and 'Obsahuje přílohy' (checkboxes for Kompletní výukové plány, Částečné výukové plány, Bez výukových plánů).

**Content Cards:**

- Jak si hrát s čísla** by Jiří Klobzuba (set 6 hodin, 6 mutací).  
Description: Understanding color theory: the color wheel and finding complementary colors.  
Details: Čtyřleté Gymnázium, 3. ročník, Matematika, Analytická geometrie.  
Rating: 24 hodnocení, 18 stažení, ★ 4,3/5, Certifikováno.
- Jak si hrát s čísla** by Jiří Klobzuba (set 6 hodin, 6 mutací).  
Description: Understanding color theory: the color wheel and finding complementary colors.  
Details: Čtyřleté Gymnázium, 3. ročník, Matematika, Analytická geometrie.  
Rating: 24 hodnocení, 18 stažení, ★ 4,3/5, Certifikováno.
- Jak si hrát s čísla** by Jiří Klobzuba (set 6 hodin, 6 mutací).  
Description: Understanding color theory: the color wheel and finding complementary colors.  
Details: Čtyřleté Gymnázium, 3. ročník, Matematika, Analytická geometrie.  
Rating: 24 hodnocení, 18 stažení, ★ 4,3/5, Certifikováno.

Obrázek 4.23.: Návrh uživatelského rozhraní vyhledávací stránky aplikace *Eduklub – plány výuky*.

## Domovská stránka webového agregátoru výukových plánů

Domovská stránka webového agregátoru výukových plánů slouží jako hlavní rozcestník pro vyhledávání a objevování výukových celků. Součástí stránky je vyhledávací pole, které umožňuje hledání podle klíčových slov a usnadňuje orientaci mezi výukovými celky.

Pro zpřesnění hledání je k dispozici několik filtrů, mezi které patří předmět, stupeň vzdělávání (1. stupeň, 2. stupeň, nižší gymnázium, vyšší gymnázium a střední škola), ročník, jazyk, hodnocení (podle počtu hvězdiček 0–5) a certifikace, která označuje výukové celky schválené pedagogy z Univerzity Karlovy. Tyto filtry umožňují snadnější orientaci v obsahu a pomáhají uživatelům rychleji najít relevantní materiály.

Pod vyhledávací sekcí se nachází oblast s doporučenými výukovými celky. Administrátor má možnost ručně označit konkrétní výukový celek jako doporučený, čímž se zvyšuje jeho šance

na zobrazení na domovské stránce. Tento systém umožňuje zdůraznit kvalitní materiály, které by mohly být pro uživatele nejvíce přínosné.

The screenshot shows the main interface of the Eduklub - plány výuky application. At the top, there is a navigation bar with links for PLÁNY VÝUKY, ČLÁNKY, VIDEA, and FÓRUM. On the right side of the bar are icons for user profile, heart, and shopping cart, along with a button labeled '+ Nahrát materiál'. Below the navigation bar, the title 'Najděte inspiraci na každou vyučovací hodinu' is displayed, followed by a subtitle 'Využijte náš vyhledávač a pomocí filtrů vstupte do světa objevování vzdělávacích materiálů pro učitele'. A search bar with placeholder text '+ Přidejte více...' and a 'Hledej' button is present. Below the search bar are several filters: Předmět (dropdown), Stupeň (dropdown), Ročník (dropdown), Certifikováno (checkbox), Jazyk (dropdown), and Hodnocení (dropdown). A section titled 'Doporučujeme' contains four course cards:

- Komunikace a sloh** (set 4 hodin) by Antonín Jančářík. It includes tags for Český jazyk a Literatura, Střední škola, 2. ročník, Vyprávění, Publicistický styl, and Prostředkovací styl. Below the tags is a brief description: 'Komunikace a sloh Tento vzdělávací modul poskytuje studentům druhého ročníku střední školy ucelený přehled o vyprávění, publicistickém a prostředkovacím stylu. Kurz začíná...'. Below the description are icons for heart, hodnocení, stažení, and settings.
- Množiny** (set 6 hodin) by Antonín Jančářík. It includes tags for Matematika, Střední škola, 1. ročník, CLIL, and Množiny. Below the tags is a brief description: 'Množiny Kurz "Množiny" určený pro studenty prvního ročníku střední školy se zaměřuje na základní principy a aplikace množin v matematice. Zabývá se množinami bodů daných vlastností,...'. Below the description are icons for heart, hodnocení, stažení, and settings.
- Goniometrie a trigonometrie** (set 5 hodin) by Antonín Jančářík.
- Časy, Hodnocení a Ekologie** (set 3 hodin) by Antonín Jančářík.

Obrázek 4.24.: Okno aplikace *Eduklub – plány výuky* s náhledem na domovskou stránku.

## Nahrání nového výukového plánu

V navigační liště se nachází tlačítko „Nahrát materiál“, které po kliknutí otevře okno pro nahrání exportovaného výukového celku z *EDUBO*.

Po výběru souboru je nutné zadat název a popis, který lze upravovat pomocí WYSIWYG editoru. Předmět se automaticky nastaví podle názvu předmětu z *EDUBO*, ale lze jej upravit. Dále je třeba vybrat ročník, třídu, kategorii a klíčová slova, která pomohou s filtrováním obsahu. Uživatel má také možnost změnit jazyk výukového celku nebo přidat výstupy z RVP, čímž zajistí přesnější zařazení materiálu do vzdělávacího systému.

### Přidat materiál

Název  
Rozšiřování slovní zásoby, Čtení, Poslech: Online

Počet hodin  
4

Popis výukového celku

**B** *I* U ~~S~~ 16 ▾

Předmět Ročník Třída  
Anglický jazyk Ročník Třída

Kategorie  
Kategorie

Klíčová slova: Pro přidání klíčového slova stiskněte Enter...

Výukové hodiny  
AJ\_05 Procvičovací online hodina s ... AJ\_06 Procvičovací hodina - online AJ\_04 Opakovací hodina AJ\_07 Čtení a poslech s porozuměním

Další možnosti ▾

**Nahrát soubor**

Obrázek 4.25.: Modální okno pro přidání nového výukového celku v aplikaci *Eduklub – plány výuky*.

### Vyhledávací stránka agregátoru výukových plánů

Vyhledávací stránka poskytuje rozšířené možnosti filtrování a umožňuje uživatelům najít výukové celky podle specifických kritérií. Kromě základních filtrů, které jsou dostupné i na domovské stránce, zde lze filtrovat výukové celky podle toho, zda se jedná o mutace – tedy upravené verze existujících plánů.

Další možností je filtrování podle kategorií, které jsou přiřazeny ke každému předmětu, což

umožnuje přesnější vyhledávání. Výsledky lze také seřadit podle počtu stažení, hodnocení a počtu hodin, což usnadňuje nalezení nejpopulárnějších nebo nejlépe hodnocených výukových plánů.

The screenshot shows the homepage of the **eduklub/plány výuky** website. At the top, there are navigation links for **PLÁNY VÝUKY**, **ČLÁNKY**, **VIDEA**, and **FÓRUM**. On the right, there are icons for user profile, heart, and shopping cart, along with a button to **+ Nahrát materiál**.

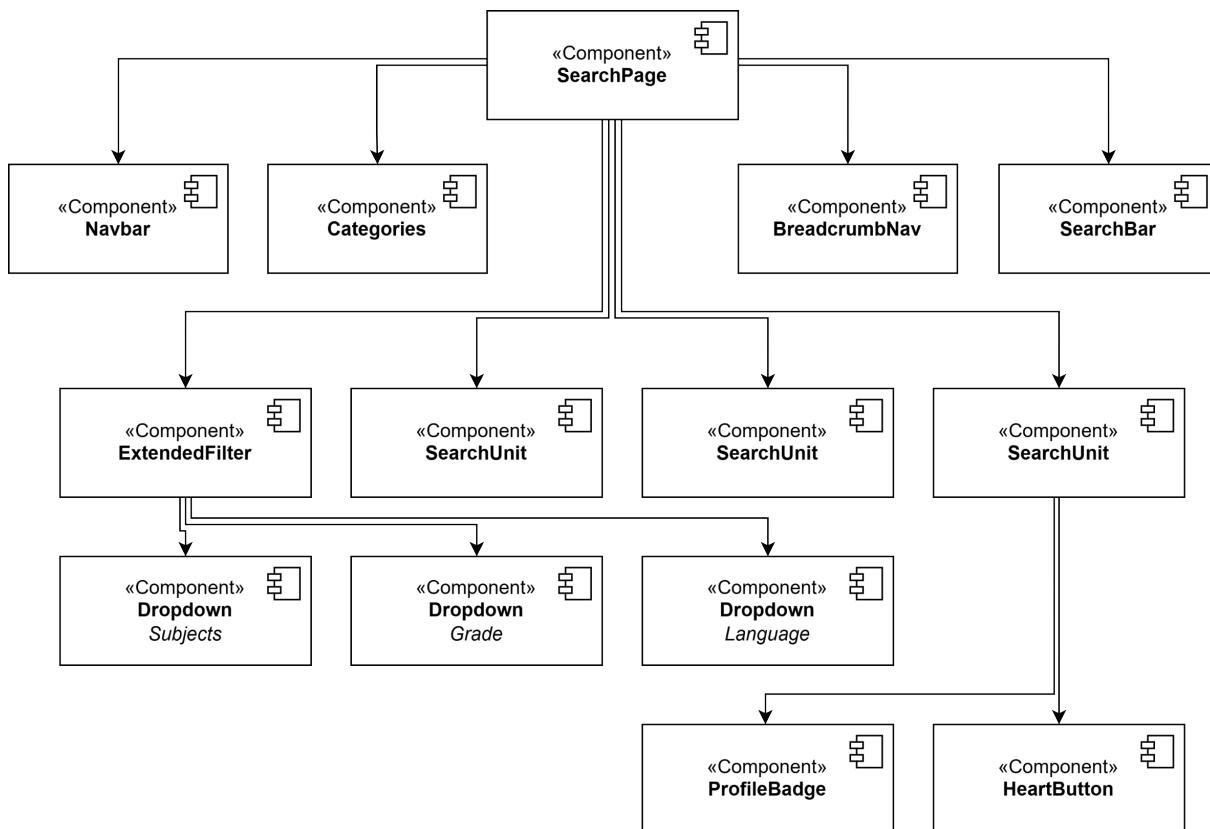
The main content area shows a breadcrumb trail: Home > Plány výuky > Matematika. Below this, a search bar displays the keywords: **Klíčová slova: Matematika X Druhý stupeň X 6. třída X**. A search input field contains the placeholder **+ Přidejte více...**.

On the left, there is a **Podrobný filtr** sidebar with dropdown menus for **Předmět** (selected: Vyberte předmět), **Stupeň** (selected: Druhý stupeň), **Ročník** (selected: Vyberte ročník), **Jazyk** (selected: Vyberte jazyk), **Hodnocení** (selected: 4-3 hvězdy), and **Certifikace** (selected: Certifikované).

The main content area displays two course cards:

- Převody jednotek** by Antonín Jančářík (set 4 hodin, 0 mutací). It includes a preview snippet: "Převody Jednotek V tomto kurzu pro šestou třídu základní školy se studenti zaměří na převody jednotek obsahu . Mod...". It shows 0 hodnocení, 4 stažení, a 0/5 rating. A **Zobrazit** button is present.
- Vrcholové, souhlasné a střídavé úhly** by Antonín Jančářík (set 4 hodin, 0 mutací). It includes a preview snippet: "Vrcholové, souhlasné a střídavé úhly". It shows 0 hodnocení, 0 stažení, and a 0/5 rating.

Obrázek 4.26.: Okno aplikace *Eduklub – plány výuky* s náhledem na vyhledávací stránku.



Obrázek 4.27.: Komponentní diagram znázorňující strukturu React komponent na vyhledávací stránce.

Tento komponentní diagram znázorňuje architekturu React komponent používaných na stránce pro vyhledávání. Hlavní komponentou je `SearchPage`, která řídí zobrazování výsledků a poskytuje přístup k filtrům a vyhledávání. Mezi klíčové komponenty patří `Navbar`, který slouží jako navigační lišta, `BreadcrumbNav`, jenž zobrazuje cestu mezi kategoriemi, a `SearchBar`, který umožnuje hledání podle klíčových slov. Komponenta `Categories` umožňuje rychlý výběr z předem definovaných kategorií.

Důležitou částí je komponenta `ExtendedFilter`, která nabízí pokročilé filtrování výukových plánů. Umožňuje uživatelům vybírat předměty, ročníky a jazyky prostřednictvím rozbalovacích menu, reprezentovaných komponentami `Dropdown`. Samotné výsledky vyhledávání jsou reprezentovány komponentou `SearchUnit`, která zobrazuje jednotlivé výukové celky. Každá karta výukového plánu obsahuje komponentu `ProfileBadge`, jež zobrazuje informace o autorovi a `HeartButton`, který umožňuje přidat plán do oblíbených.

```

1 export const getCachedCategories = async () => {
2   const cachedObject = sessionStorage.getItem("categories");
3   if (cachedObject) {
4     const objects: TSubjectWithCategories[] = JSON.parse(cachedObject);
5     if (objects) {
6       return objects;
7     }
8   }
  
```

```

9
10 const response = await axios.get("/api/categories/subjects");
11 if (response.status === 404) {
12   return [blankSubjectWithCategories];
13 }
14 const loadedObject: TSubjectWithCategories[] =
15   response.data.sort((a:Category, b:Category) => a.id - b.id);
16
17 sessionStorage.setItem("categories", JSON.stringify(loadedObject));
18 if (loadedObject) {
19   return loadedObject;
20 }
21 return [blankSubjectWithCategories];
22 };

```

Tato funkce `getCachedCategories` slouží k efektivnímu načítání seznamu kategorií s využitím mezipaměti (`sessionStorage`). Nejprve se pokusí načíst uložená data ze `sessionStorage` pomocí klíče „categories“. Pokud jsou kategorie v mezipaměti, převedou se zpět na objekt a ihned se vrátí, čímž se šetří volání na server.

Pokud kategorie v mezipaměti nejsou, provede se asynchronní požadavek na backend. V případě chyby 404 se vrátí výchozí prázdný objekt `blankSubjectWithCategories`. Jinak se získaná data seřadí podle `id` a uloží do `sessionStorage`, aby se při dalším volání funkce nemusel znova kontaktovat backend. Poté se vrátí načtený a uložený seznam kategorií.

## Stránka náhledu výukového celku

Stránka náhledu výukového celku poskytuje podrobný přehled o konkrétním výukovém celku a jeho obsahu. Uživatel zde vidí základní informace, jako je název plánu, autor, klíčová slova a další metadata. Struktura stránky je rozdělena do několika záložek, které umožňují přístup k různým detailům výukového celku.

Záložka **Popis** obsahuje textový přehled výukového plánu. Náhled plánu umožňuje prohlédnout PDF exporty všech hodin, které výukový celek obsahuje. Sekce **Hodnocení** umožňuje uživatelům hodnotit výukový celek pomocí hvězdičkového systému (0–5 hvězdiček) a přidávat plusy a minusy. **Výstupy RVP** zobrazují všechny kurikulární výstupy rámcového vzdělávacího programu, které jsou v celku zahrnuty. Záložka **Poradna a návody** obsahuje odkazy na články v Eduklubu, které mohou být užitečné pro danou výuku. V sekci **Alternativy** může autor přidat výukové plány, které jsou tematicky podobné. V sekci **Mutace** se zobrazují upravené varianty původního výukového plánu.

## 4. Praktická část

The screenshot shows a user interface for managing lesson plans. At the top, there's a navigation bar with links to Home, Plány výuky, Matematika, Algebra, and Lineární algebra. Below that is a title 'Operace s čísly - set 5 hodin' with icons for like, edit, and delete. A sidebar on the left lists various lesson plan items. The main content area displays details for a specific lesson: 'EDUBO' subject, 'Matematika' subject, '7. třída' class, and 'viktoria.tsyba@gmail.com' email. It shows the 'Výukový celek: Operace s čísly' and 'Cíl hodiny: Žák vynásobí dvě celá čísla s libovolnými znaménky.' Below this, there's a section for 'Před výukou' (Pre-class) with a note to 'Zopakujte si předchozí hodinu.' (Review the previous lesson). There are also icons for video and audio, a duration of '2 min', and a 'Popis:' (Description) link. At the bottom, there's a timer icon and the text 'Aktivita na rozhýbání' (Activity for physical activity).

Obrázek 4.28.: Okno aplikace *Eduklub – plány výuky* s náhledem na stránku náhledu výukového celku.

## 4.5. Uživatelský manuál

V této kapitole si představíme dva základní scénáře práce s aplikací *EDUBO* z pohledu učitele a žáka. Nejprve si krok za krokem ukážeme proces registrace učitele, vytvoření třídy, předmětu, výukového celku, výukového plánu a publikace výukové hodiny. Následně se zaměříme na to, jak se do systému přihlašuje žák, jaké prostředí ho po přihlášení čeká a jakým způsobem se dostane ke své výukové hodině. Každý krok je podrobně popsán, aby byl snadno srozumitelný i pro uživatele, kteří s aplikací začínají.

### 1. scénář: Prvotní registrace učitele, vytvoření výukového plánu a vytvoření přístupových klíčů pro žáky

#### Krok 1: Registrace učitele

Aplikaci *EDUBO* můžete otevřít v testovacím prostředí na adrese <https://app.eduklub.cz/>. Na úvodní obrazovce *EDUBO* klikněte na tlačítko „Registrovat se“, které vás přesměruje na registrační formulář. Zde vyplňte svůj e-mail, heslo a registrační klíč. Tento klíč určuje vaše školní

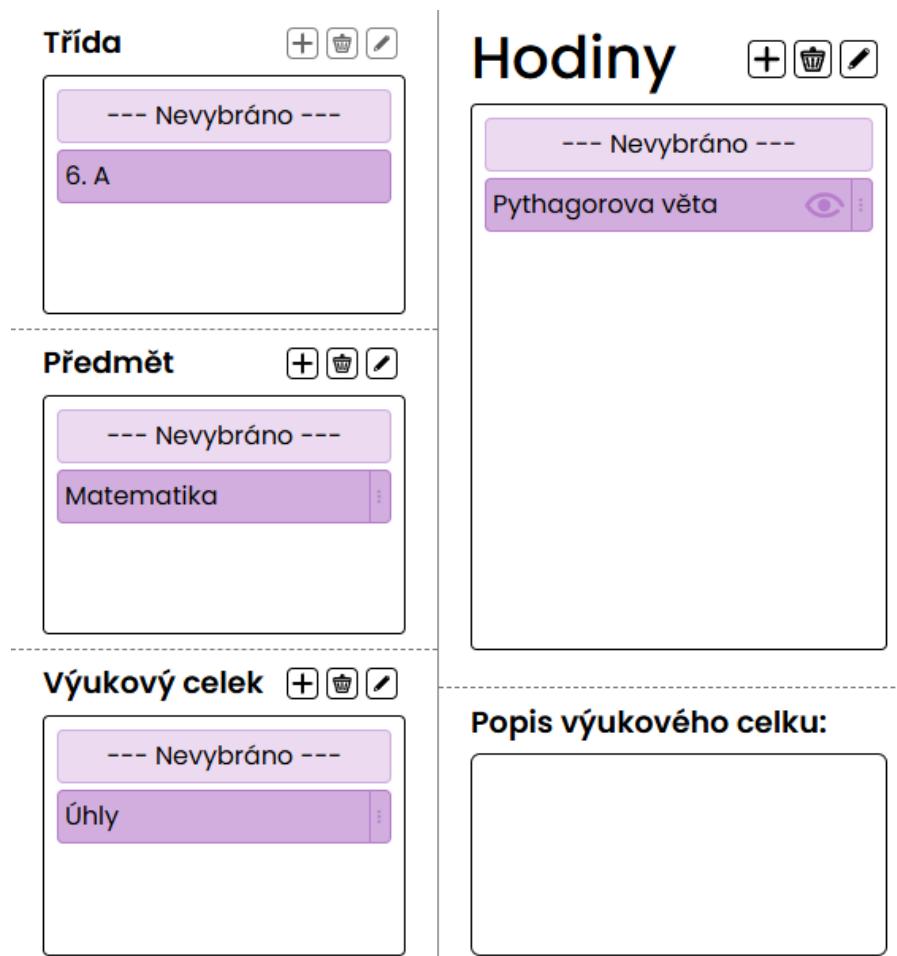
zařízení a je poskytován školou. Na testovacím nasazení je klíč „teacher“. Zaškrtněte souhlas se zpracováním osobních údajů. Po registraci se přihlaste do *EDUBO* se svými novými přihlašovacími údaji.

### Krok 2: Vytvoření třídy

Po přihlášení je potřeba vytvořit třídu, do které budete přidávat žáky. Klikněte na ikonu „plus“ v sekci Třídy. Zde máte možnost buď přidat se do již existující třídy nebo vytvořit třídu novou. Zadejte název třídy a klikněte na tlačítko pro vytvoření.

### Krok 3: Vytvoření předmětu

Po vytvoření třídy přejděte na sekci „Předměty“, kde klikněte na tlačítko „plus“. Zadejte název předmětu, který bude v rámci třídy vyučován. Můžete také přiřadit kurikulární výstupy kliknutím na tlačítko „Přiřadit výstupy k předmětu“, vyberte požadované kurikulární výstupy a přiřaďte je k předmětu.



Obrázek 4.29.: Část uživatelského menu aplikace *EDUBO*.

#### **Krok 4: Vytvoření výukového celku**

Následně vytvořte výukový celek kliknutím na tlačítko „plus“ v sekci „Výukové celky“. Zadejte název výukového celku a potvrďte jeho vytvoření.

#### **Krok 5: Vytvoření výukového plánu**

Nyní přejděte do sekce „Hodiny“ a klikněte na tlačítko „plus“ pro vytvoření výukového plánu. Zadejte název plánu a klikněte na „Vytvořit“. V pravém dolním rohu menu klikněte na tlačítko „Editovat“, které vás přesměruje do editoru výukového plánu.

#### **Krok 6: Editace výukového plánu**

V editoru výukového plánu můžete v horní části změnit název, přidat cíl hodiny a přiřadit kurikulární výstupy. V pravém dolním rohu se nachází textové pole pro poznámky. V levém sloupci můžete vybírat mezi různými typy aktivit a přetáhnout je na plátno pomocí drag and drop interakce. Pokud nevíte, jak využít konkrétní aktivitu, klikněte na informační tlačítko u dané aktivity, které otevře podrobné metodické materiály s návody na jejich efektivní využití ve výuce.

Každou aktivitu můžete upravit kliknutím na tlačítko tužky u konkrétní aktivity. Zde můžete měnit název, popis, délku, pomůcky, soubory, odkazy, organizační formu a formu činnosti.

**Název:**

Aktivování vstupních znalostí

**Popis:**

**B** **I** **U** **S** 16 ▾

Co už o tématu víme? (z minulého učiva tohoto nebo jiného předmětu, ze zkušenosti žáků nebo z kulturního kontextu?)

**Délka aktivity:**

5

**Pomůcky:****Soubory:**

Sem klikněte nebo přetáhněte soubor...



Přidat komentář ▾

**Odkazy:**

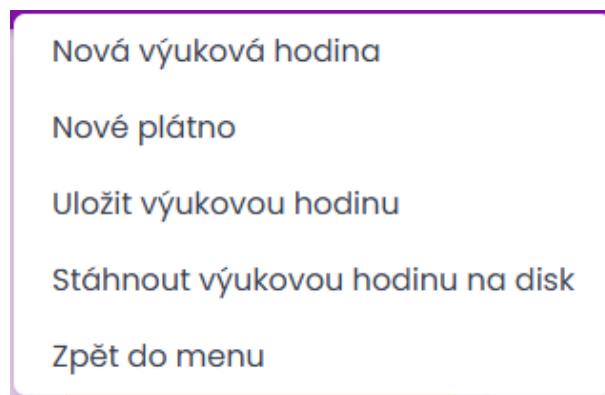
Přidat komentář ▾

**Organizační forma****Forma činnosti****Uložit**

Obrázek 4.30.: Modální okno pro úpravu aktivity v aplikaci *EDUBO*.

**Krok 7: Uložení výukového plánu**

Po vytvoření výukového plánu a přidání všech aktivit, uložte svůj výukový plán pomocí navigační lišty: *Soubor* → *Uložit výukovou hodinu*. Poté se vraťte zpět do menu: *Soubor* → *Zpět do menu*.



Obrázek 4.31.: Rozbalovací nabídka tlačítka „Soubor“ v navigační liště v aplikaci *EDUBO*.

### Krok 8: Publikace výukového plánu

Vyberte právě vytvořený výukový plán, nastavte datum a čas konání hodiny v pravém dolním rohu a klikněte na tlačítko „Publikovat“, aby byla hodina zveřejněna pro žáky.

#### Publikovat hodinu:

duben 2025						
po	út	st	čt	pá	so	ne
31.	1.	2.	3.	4.	5.	6.
7.	8.	9.	10.	11.	12.	13.
14.	15.	16.	17.	18.	19.	20.
21.	22.	23.	24.	25.	26.	27.
28.	29.	30.	1.	2.	3.	4.
5.	6.	7.	8.	9.	10.	11.

**Čas (od)**      **Čas (do)**

08:00      08:45

**Aktuální status:** Publikováno

**Publikovat**

Obrázek 4.32.: Část uživatelského menu určeného pro publikování výukového plánu v aplikaci *EDUBO*.

### Krok 9: Generování registračních klíčů pro žáky

Pro vytvoření registračních klíčů pro žáky vyberte svou třídu a v navigační liště klikněte na tlačítko „Generovat klíče“. Zadejte požadovaný počet klíčů, které chcete vygenerovat. Po zadání počtu se

vygenerují klíče, které si můžete vytisknout a předat žákům.

## Přístupové klíče studentů pro třídu 6. A

---

c945c11c

---

80b5704e

---

e7211989

---

b1d7f07b

---

Obrázek 4.33.: Ukázka vygenerovaných přístupových klíčů studentů pro třídu 6. A připravených k tisku.

## 2. scénář: Prvotní registrace žáka, zobrazení pohledu a zobrazení výukové hodiny

### Krok 1: Registrace žáka

Aplikaci *EDUBO* můžete otevřít v testovacím prostředí na adrese <https://app.eduklub.cz/>. Na úvodní obrazovce klikněte na tlačítko „Registrovat se“, které vás přesměruje na registrační formulář. Zadejte svůj e-mail, zvolte heslo a vložte registrační klíč, který vám předal učitel. Na testovacím nasazení si můžete klíč vygenerovat sami (viz 1. scénář, krok 9). Nezapomeňte zaškrtnout souhlas se zpracováním osobních údajů. Po dokončení registrace se přihlaste do *EDUBO* pomocí zadaných údajů.

## Krok 2: Výběr pohledu

Po přihlášení si žák volí mezi dvěma pohledy – **Předměty** a **Kalendář**.

**Vyberte pohled:**



Obrázek 4.34.: Rozhraní pro výběr pohledu ve studentském menu.

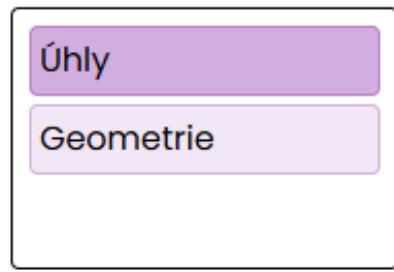
### Pohled Předměty

Pohled Předměty nabízí přehled všech předmětů vyučovaných ve třídě. Po kliknutí na konkrétní předmět se zobrazí výukové celky, a následně i publikované hodiny. V dolní části obrazovky žák najde materiály k hodině – přílohy, odkazy, pomůcky a možnost hodnocení pomocí smajlíků. Vpravo jsou zobrazeny informace o hodině.

### Předměty



### Výukové celky



Obrázek 4.35.: Navigace předmětů a výukových celků v pohledu předměty.

**Přehled materiálů k hodině:**

images.png

[https://cs.wikipedia.org/wiki/Pythagorova\\_v%C4%9Bta](https://cs.wikipedia.org/wiki/Pythagorova_v%C4%9Bta)

Obrázek 4.36.: Rozhraní přehledu materiálů ve vybraném výukovém plánu.

**Pohled Kalendář**

Pohled Kalendář zobrazuje v levé části měsíční kalendář. Po výběru konkrétního dne se ve střední části zobrazí zkrácený přehled hodin publikovaných v daný den. Ten připomíná PDF export hodiny. I zde je ve spodní části seznam všech materiálů a v pravé části podrobný popis hodiny.

**Kalendář**

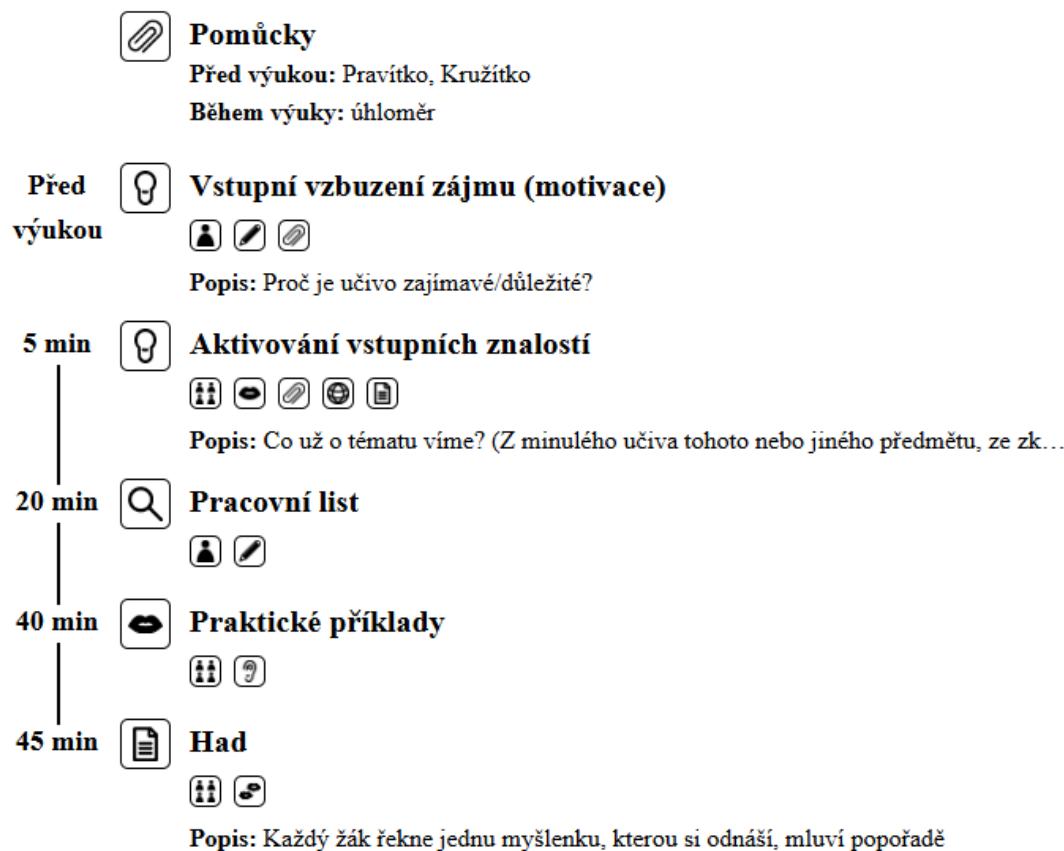
< duben 2025 >

po	út	st	čt	pá	so	ne
31.	1.	2.	3.	4.	5.	6.
7.	8.	9.	10.	11.	12.	13.
14.	15.	16.	17.	18.	19.	20.
21.	22.	23.	24.	25.	26.	27.
28.	29.	30.	1.	2.	3.	4.
5.	6.	7.	8.	9.	10.	11.

**Hodiny**

**Pythagorova věta**

Obrázek 4.37.: Navigace hodin a kalendář v pohledu kalendář.



Obrázek 4.38.: Zkrácený přehled výukového plánu v pohledu kalendář.

### Krok 3: Zobrazení výukového plánu

Pomocí tlačítka „Otevřít“ vpravo dole se žák dostane k podrobné struktuře výukového plánu. Zde si může zobrazit samotné plátno s aktivitami – každá aktivita se dá rozkliknout a zobrazí se její popis, délka, potřebné pomůcky, odkazy či přiložené soubory.

## Pythagorova věta

Před hodinou:



Vstupní vzbuzení zájmu (motivace)



Obrázek 4.39.: Zobrazení výukového plánu ve studentském rozhraní.



## 5. Testování softwaru uživateli

Aplikace *EDUBO* byla testována průběžně během celého vývoje. V počáteční fázi jsme testovali aplikaci interně. Zaměřovali jsme se na ověření základní funkčnosti jako registrace, přihlašování, vytváření výukových plánů a přetahování aktivit v plátně. První iterace sloužila především k technickému ověření základních scénářů a stabilitě editoru.

Od druhé iterace jsme začali zapojovat skutečné uživatele – pedagogy z Pedagogické fakulty Univerzity Karlovy, kteří začali v *EDUBO* vytvářet své výukové plány, které se později publikovali na webu *Eduklub – plány výuky*. Tím se ukázala silná stránka nástroje jako podpory pro reálnou pedagogickou činnost, ale současně vyvstala i řada nedostatků. Mezi nejvýznamnější patřilo špatné vykreslování názvu výukového plánu – při delším názvu nebyl text zcela viditelný, dále padání aplikace při vkládání rovnic z Microsoft Word nebo nemožnost posouvat modální okno při úpravě aktivity na touchpadech gestem. Všechny tyto chyby byly postupně identifikovány, zaznamenány a odstraněny.

V roce 2024 se *EDUBO* stalo nástrojem i pro výuku studentů Pedagogické fakulty Univerzity J. E. Purkyně. Přibližně 250 studentů mělo za úkol vytvořit vlastní výukový plán. Jejich zpětná vazba byla velmi cenná – většina z nich hodnotila aplikaci pozitivně, chválili intuitivní ovládání, přehlednost a celkový přínos při strukturování hodiny. Objevila se však také závažná technická chyba: pokud uživatel přetáhl aktivitu do plátna a v rychlém sledu chytil jinou, aplikace spadla. Jednalo se o problém v knihovně *react-beautiful-dnd*, která zajišťuje přetahování prvků a od roku 2022 nebyla aktualizována. Vzhledem k tomu bylo nutné implementovat vlastní řešení – uživatel může nyní pracovat vždy pouze s jednou aktivitou najednou, čímž se problém eliminoval. Potěšující bylo, že i po skončení kurzu někteří studenti pokračovali v používání *EDUBO* a kontaktovali mě s dotazy ohledně dalšího vývoje aplikace, což potvrzuje její využitelnost i v praxi.

V roce 2025 proběhla rozsáhlá technická revize kódu. Došlo k aktualizaci používaných balíčků, refactoringu struktury aplikace, opravě mnoha drobných stylových chyb a přidání nových animací. Zároveň došlo k významnému zrychlení načítání aplikace – velikost přenášených dat se snížila z původních 8 MB na 0,8 MB, což se pozitivně projevilo na výkonu i použitelnosti aplikace při slabším připojení.

Na rozdíl od *EDUBO* nebyla aplikace *Eduklub – plány výuky* zatím reálně otestována cílovými uživateli. Ačkoli je její technická příprava dokončena, čeká se na rozhodnutí o dalším směřování projektu. Věříme však, že i tato platforma má potenciál stát se užitečným nástrojem pro sdílení inspirativních výukových plánů mezi pedagogy.

Celkově lze říci, že *EDUBO* se díky intenzivní zpětné vazbě a opakovanému testování v reálném provozu stalo stabilním a prakticky využívaným nástrojem. Zkušenosti učitelů i studentů se výrazně promítly do vývoje a pomohly aplikaci přiblížit jejím uživatelům.

## 6. Diskuse a výsledky

Vývoj softwaru *EDUBO* byl technicky i organizačně náročným, ale zároveň velmi přínosným procesem. V rámci hlavního vývoje bylo do repozitáře *EDUBO* odevzdáno přes 1500 commitů (příspěvků kódu), přičemž hlavní vývoj trval 11 měsíců. Následná fáze údržby pokračuje dodnes, kdy jsou prováděny menší opravy, aktualizace a úpravy dle zpětné vazby uživatelů. V případě doplňkové aplikace *Eduklub – plány výuky* šlo o více než 600 commitů a vývoj trval přibližně čtyři měsíce, přičemž také tato část je i nadále udržována.

Technologicky šlo o náročný projekt, jehož největší výzvou byl bezesporu vlastní *drag and drop* editor výukových plánů. Právě jeho vývoj byl velmi náročný zejména kvůli původnímu rozhodnutí použít knihovnu, která se již dlouho neudržovala a obsahovala množství chyb. To vedlo ke komplikacím, které se musely řešit implementací oprav, čímž se zvýšila udržitelnost a stabilita této klíčové části systému.

Aplikace *EDUBO* je v současnosti nasazena na serverech společnosti Mironet.cz a.s., kde funguje stabilně a je dostupná pro každého, kdo si ji chce vyzkoušet. Ve fázi testování byla aplikace využívána pedagogy Pedagogické fakulty Univerzity Karlovy, kteří v ní vytvořili přes 200 výukových plánů a studenty Pedagogické fakulty Univerzity Jana Evangelisty Purkyně, kteří v ní vytvořili přes 200 výukových plánů. Přestože v tuto chvíli není aplikace využívána plošně, základ je připraven a existuje potenciál pro širší nasazení.

V rámci vývoje se ukázalo několik oblastí, které by bylo vhodné dále rozpracovat. Hlavní prioritou je přepsání backendu *EDUBO* z Flasku na Django a přechod z databáze MongoDB na PostgreSQL, což by do budoucna výrazně zjednodušilo údržbu i rozšiřování systému. Mezi další rozvojové oblasti patří například vytvoření správy tříd a jejich organizace, vytvoření jednoduššího instalačního skriptu pro nasazení vlastních verzí *EDUBO* a vylepšení UI/UX studentského pohledu, aby byl pro žáky přehlednější a atraktivnější.



## 7. Závěr

Vytvoření softwaru *EDUBO* pro mě představovalo zásadní milník – nejen po technické, ale i osobní stránce. Šlo o první velký projekt, který jsem vedl jako leader týmu a který měl zároveň dopad na vzdělávání. Ačkoliv jsem původně do projektu vstupoval jako programátor, postupně jsem převzal i zodpovědnost za organizaci celého vývoje, komunikaci s týmem a pedagogickými pracovníky a koordinaci jednotlivých etap práce.

V této roli jsem získal zkušenosti, které bych jinde získával mnohem déle. Musel jsem vést pravidelné stand-upy, řešit krizové situace, ale zároveň i pomáhat méně zkušeným kolegům a přispívat k učení v týmu. Například na konci první iterace, kdy se spojoval frontend s backendem a došlo ke kolapsu funkcionalit, jsem trávil celé dny opravami a uvědomil si, že bez lepsí metodiky a kontroly nelze takto velký projekt efektivně řídit. I proto jsem se ujal vedení vývoje a nastavoval standardy, které do té doby chyběly.

Nejvíce mě naplňovala kombinace samotného programování a podpory ostatních. Měl jsem možnost se učit nové technologie jako React a Django od kolegů a zároveň předávat své zkušenosti dál. Díky tomu se rozvinulo nejen moje technické porozumění, ale i schopnost předávat informace jiným.

Projekt *EDUBO* získal mnohem pozitivnější odezvu, než jsem očekával. Studenti, kteří s ním pracovali, ho označili za užitečný nástroj a využili ho k tvorbě desítek výukových plánů. Někteří dokonce projevili zájem pokračovat v jeho používání a zajímal se o další vývoj. Přestože zatím čekáme, zda se projekt posune směrem k širšímu nasazení do škol, věřím, že má skutečný potenciál a stojí za to na něm dále pracovat.

Tato závěrečná práce tak není jen popisem vývoje softwaru, ale především dokumentem o růstu – jak profesním, tak osobním. Ať už se budoucnost *EDUBO* vyvine jakkoliv, pro mě byl tento projekt zásadní kapitolou mého studia i kariéry.



## 8. Externí přílohy

Externí přílohy této bakalářské práce jsou umístěny na adrese:

[https://github.com/matej-kaska/thesis\\_ki\\_ujep](https://github.com/matej-kaska/thesis_ki_ujep).

V GitHub repozitáři se nachází tyto externí přílohy:

---

ki-thesis.pdf	text práce v PDF
ki-thesis.tex	zdrojový kód práce v $\text{\LaTeX}$ u
kitheses.cls	definice třídy dokumentů (rozšířená třída <code>scrbook</code> )
images	složka se všemi použitými obrázky
codes	zdrojové kódy použité v této práci
diagrams_svg	diagramy ve formátu SVG

---

Všechny tyto soubory jsou potřeba pro překlad dokumentu.



# **9. Základní členění závěrečné práce**

## **9.1. Kapitoly**

Závěrečná práce by měla být členěna na tři až šest kapitol nejvyšší úrovně (nepočítaje úvod a závěr). Ty by měly odpovídat osnově práce, ale nikoliv doslovně. Jiné mohou být názvy a dokonce i počet kapitol. To jest, některé body osnovy lze rozpracovat ve více kapitolách a naopak některé body lze združit do jediné kapitoly.

Důležité je, aby bylo členění přibližně rovnoměrné. Problematické jsou především kapitoly malého rozsahu (1-3 stránky) respektive naopak kapitola zahrnující podstatnou část textu.

Kapitoly nejvyšší úrovně vždy začínají na nové stránce, která musí být lichá tj. na pravé straně. Pokud předchozí kapitola končí na liché straně, vkládá se tzv. vakát – zcela prázdná strana.

## **9.2. Sekce resp. podkapitoly**

**L****A****T****E**X podporuje čtyři úrovně vnoření kapitol (kromě kapitol i sekce, podsekce, a podpodsekce), což by mělo být více než dostatečné. Druhou úrovní jsou sekce, jejichž rozsah by měl být alespoň jednu stranu, mohou však být i výrazně delší. Zobrazují se v obsahu a jsou číslované a lze je tudíž snadno odkazovat. Měli by být použity b každé kapitole kromě úvodu a závěru.

### **Podsekce**

Podsekce se už používají méně často a běžně se vyskytují i dále nečleněné sekce. Nejsou číslovány a nejsou zobrazeny v obsahu. Rozsah by měl být alespoň dva odstavce.

### **Podpodsekce**

Podpodsekce se používají jen velmi zřídka, typicky jen u kapitol či sekcí popisující systémy s výraznou hierarchií. Tvořeny mohou být i jediným odstavcem (nikoliv však ale jedinou řádku, zde zvažte spíš použití definičního výčtu).

### 9.3. Rozsah závěrečné práce

Optimální rozsah bakalářské práce není možné stanovit, neboť závisí na jejím charakteru. Práce jejíž součástí je vytvoření aplikace mohou být menšího rozsahu než práce čistě popisné. I v tomto případě by však měla mít alespoň třicet stran (nepočítaje úvodní stránky až po obsah a stránky od seznamu použitých zdrojů dále a prázdné stránky tzv. vakáty).

Maximální rozsah práce je ještě obtížnější stanovit. Obecně však doporučuji v případě, že pokud rozsah (opět bez povinných částí) přesáhne osmdesát stran je vhodné uvažovat o redukci (ne všechno, co jste napsali musí být nakonec uvedeno ve finální verzi) respektive přesunu do externích příloh umístěných na Githubu (viz příloha A na straně 131). Týká se to především různých schémat, obrazového materiálu apod.

# 10. Typografie

Při tvorbě bakalářské práce byste měli dodržovat základní zásady typografie. Bakalářská práce by měla být přehledná a dobře čitelná. A možná i krásná (uvědomuji si, že krása je subjektivní).

## 10.1. Písmo

Na naší katedře není předepsáno, jaké písmo by mělo být použito při sazbě práce. Měli byste však dodržovat několik zásad.

- pro sazbu většiny textu by mělo být použito tzv. knižní písmo tj. písmo určené pro sazbu soudobých knih (především odborných). Doporučit lze antikvová písma a resp. písma lineární (bezpatková).
- pro výpisy kódu používejte neproporcionalní písmo (pro ostatní text naopak proporcionalní). Pro základní přehled volně šířitelných písmen se můžete podívat do dokumentu Monotypář.
- pokud používáte více druhů písma, pak by měla být vizuálně kompatibilní (nejlepší je použít písma, která jako kompatibilní vytvořili či alespoň doporučili typografové)

V šabloně je využita rodina písma *Libertinus* (odvozená z písma *Linux Libertine*) s výjimkou neproporcionalního písma, pro něž bylo zvoleno písmo *Source Code Pro*, které podporuje výrazně větší počet řezů (kurziva tučné).

Standardní velikost písma kvalifikační práce je 12 bodů (přesněji řečeno tzv. pica bodů tj. 12/72 palce). Tato velikost však ve většině případů neodpovídá žádnému viditelnému rozdílu písma (v klasické typografii je to výšky kuželky, na níž je umístěn vystouplý reliéf písma). Z tohoto důvodu se skutečná výška a především šířka písmen může u jednotlivých písmen viditelně lišit. To je problém, pokud se znaky různých písem vyskytují vedle sebe. Často se výrazněji liší například výška antikvy a neproporcionalního písma. V tomto případě je často nutné velikost neproporcionalních písem mírně zmenšit (tato šablona tak v případě využití XeTeXu nebo LuaTeXu činí automaticky).

Optimální řádkování pro bakalářské práce je 1½ (použité i v této šabloně).

## 10.2. Základní typografická pravidla

### 10.3. Znaky, které nenajdete na klávesnici

I v běžném textu se bohužel vyskytují znaky, které nenajdete na klávesnici. Jedná se především o uvozovky a pomlčky (znak – je spojovník nikoliv pomlčka). Ostatní znaky se v práci vyskytnou jen výjimečně.

V L<sup>A</sup>T<sub>E</sub>Xu lze tyto znaky vkládat pomocí tří základních mechanismů:

1. příkazem
2. pomocí sekvence speciálních znaků
3. přímo jako Unicode znaky (v editoru)

Následující tabulka ukazuje nejčastěji používané znaky, jež nenajdete na klávesnici, ale jejich použití je při české sazbě povinné.

jméno znaku	L <sup>A</sup> T <sub>E</sub> X makro	Unicode	popis
„ spodní dvojité 9-uvozovky	\quotedblbase	U+201E	užívány jako úvodní uvozovky
“ horní dvojité 6-uvozovky	‘	U+201C	užívány jako koncové uvozovky
» pravé francouzské uvozovky	>>	U+00BB	alternativní úvodní uvozovky
« levé francouzské uvozovky	<<	U+00AB	alternativní koncové uvozovky
, spodní jednoduché 9-uvozovky	\quotesinglbase	U+201A	vnořené úvodní uvozovky
‘ horní jednoduché 6-uvozovky	‘ ‘	U+2018	vnořené koncové uvozovky
... výpustka	\ldots	U+2026	vypuštěný text
- pomlčka	--		od–do, oddělovač vět

Znak tzv. obousměrných uvozovek " , jež vznikly v době psacích strojů a přežily na počítačových klávesnicích lze využívat jen na při zápisu zdrojáků programovacích jazyků. Podobně jen ve zdrojácích lze využít trojtečku (= tři tečky). Rozdíl oproti výpustku je malý ale viditelný: výpustek... versus trojtečka...

### Jednopísmenné předložky a spojky na konci řádku

Dle českých typografických pravidel je chybou použití jednopísmenných předložek a spojek na samém konci řádků. Určitou výjimkou jsou spojky „a“ a „i“, které jsou tolerovatelné (i když nikoliv

v případě, že jsou psány velkým písmenem například na začátku věty resp. pokud následují hned za otvírací závorkou).

Jediným řešením je vložení nezalomitelných mezer mezi předložku resp. spojku a následující slovo. To se může dít automaticky (v  $\text{\LaTeX}$ u se používá externí program `vlna`), ale lze to dělat i ručně (nejlépe jen na místech, kde bylo toto pravidlo porušeno).

Pozor: Toto pravidlo se týká i případů, kdy je bezprostředně před předložkou či spojkou i nějaký nepísmenný znak např. otvírací závorka či uvozovka).

Striktní typografická pravidla si vynucují použití nezalomitelné mezery i na dalších místech. Pro kvalifikační práce je důležité jeho využití mezi číslem a kvantifikovanou veličinou nejčastěji fyzikální jednotkou tj. např. 42 kg nebo 256 bytů). V tomto případě nezbývá nic jiného než ruční vkládání.

Nezalomitelná mezera se v  $\text{\TeX}$ u zapisuje znakem vlnka `\sim` (kolem znaku nesmí být žádné mezery).

## Parchanti

Jako *parchant* se označují případy, kdy stránka začíná posledním krátkým rádkem odstavce (tzv. sirotek), případně končí-li stránka samostatným nadpisem nebo prvním rádkem odstavce (tzv. vdova). Pozor na případy, kdy stránka začíná či končí plovoucím obrázkem či tabulkou (i za nimi či před nimi mohou být parchanti).

Typografické systémy se snaží parchantům zabránit, ne vždy se jim to ale s úspěchem podaří. Pokud se to napodaří je nutné manuální řešení. Nejjednodušším bývá zkrácení či prodloužení předchozího textu. Není-li to možné, lze výjimečně vložit na vhodné místo ruční zalomení stránky (v  $\text{\TeX}$ u doporučuji příkaz `clearpage`).

## Vertikální a horizontální zarovnávání

Běžný text bakalářské práce by měl být zarovnán do bloku tj. jak zleva tak zprava (automatickou výjimkou jsou tzv. východové rádky odstavců, které se zarovnávají jen zleva).

V  $\text{\TeX}$  je relativně časté přetečení rádků, tj. situace, kdy systém nenajde vhodné místo pro zalomení a tak rádek pokračuje i do okraje. Pokud už tato situace nastane, pak existují dvě řešení, buď změňte text na rádku (zkrátíte ho či prodloužíte) nebo pomůžete algoritmu pro dělení slov, tím že explicitně vyznačíte místa, kde může dojít k dělení pomocí příkazu `\-`. Toto řešení je možné využít i v případě, že se nějaké slovo rozdělí viditelně chybně (slovo se nebude dělit v jiných než vyznačených místech).

Dělení slov je jen pomocný mechanismus (nikoliv pravopisný) a tak pro něj v češtině neexistují závazná pravidla pouze úzus a doporučení. Při dělení se primárně vychází z výslovnosti nikoliv ze zápisu slov (to je klíčové především cizích slov, která se čtou podle jiných pravidel než v češtině). Pro základní orientaci stačí znát tato pravidla (aplikují se postupně):

- nedělí se jednoslabičná slova
- neoddělují se jednoznakové předpony pokud je jasné složení slov z morfémů (předpona, kmen, přípona) pak se dělí podle morfémů (výjimkou jsou přípony začínající samohláskou). Slovo může mít i více předpon a přípon: např. ne-pře-mosti-tel-ný.
- v ostatních případech se dělí podle slabik (nikoliv však mezi samohláskami!) např. fi-lo-zo-fie
- v případě styku dvou a více souhlásek se dělí (kdekoliv) mezi nimi s výjimkou případů, kdy toto seskupení může stát na začátku slov (např. „st“, „př“, „str“), kdy je vhodnější dělit před skupinou hlásek např. pe-strý (i když i dělení pes-trý a pest-rý jsou možná).

Detailnější pravidla najeznete například v jazykové poradně Ústavu pro jazyk český [X], doporučená dělení pro konkrétní slova můžete nalézt například ve Wikislovníku.

Pokud k vertikálnímu přetečení dochází často, je možné lokálně či globálně (tj. v tzv preambuli) dát sázecímu algoritmu více volnosti. Nejvhodnější je použití balíku *microtype*, který zapíná další možnosti po vyrovnání sazby např. drobné změny v šířce písmen. Navíc se tím aktivuje tzv. optické zarovnávání (vpravo), kdy je jemně porušeno přesné zarovnání za účelem vyrovnání optického klamu (např. řádka zakončená tečkou se při přesném zarovnání jeví jako kratší). Méně vhodné, je použitá příkaz `\sloppy`, které zvyšuje toleranci algoritmu. Výsledkem tak může být odstranění některých přetečení, za cenu příliš velkých mezer mezi slovy (efekt je podobný MS Wordu).

Z tohoto důvodů je v šabloně bakalářské práce vložen balík *microtype*, ale příkaz `\sloppy` je využit pouze v rámci seznamu literatury, kde je správné zarovnávání komplikované (především při použití dlouhých URL).

*LATEX* se snaží i o vertikální zarovnání tj. udržování stejného horního a dolního okraje. Zatímco v případě horního okraje je zarovnání zcela přirozené, může zarovnání dole vést k problémům. Aby ho bylo dosaženo musí totiž *LATEX* roztahovat tzv. pružné vertikální mezery např. mezery před a za nadpisy sekcí, před plovoucími objekty, apod.

Takových míst je ale často na stránkách málo a tak se to nemusí vůbec podařit resp. se použije neakceptovatelná vertikální výplň.

Nejjednodušší řešením je příkaz `\raggedbottom`, jenž zarovnání dole vypíná. Nevznikají tak enormně velké horizontální mezery, chybějící zarovnání je však často rušivé hlavně u vedlejších stran u oboustranného tisku. V praxi se však ukazuje, že u závěrečných prací je závažnějším nedostatkem nadměrné roztahování a proto je `\raggedbottom` v této šabloně použit (na konci preambule), lze jej však přirozeně smazat či zakomentovat.

# **11. Grafika**



## **12. Sazba ukázek kódu**

Sazba ukázek kódu je klíčová pro bakalářské práce věnované implementaci nějaké aplikace či knihovny.

Základní zásady pro sazbu ukázek kódu:

1.



# 13. Citace

Citace tvoří jeden ze základních pilířů závěrečné práce. Platí zde základní pravidlo: pokud použijete jakoukoliv zdroj informací, pak je nutné tento zdroj citovat, tj. uvést příslušný zdroj.

Zdrojem je ve většině případů text, ale může to být i obrázek, audiovizuální materiál či ve speciálních případech i ústní sdělení. V případě informatických prací je častým zdrojem u zdrojový kód.

Informaci ze zdroje můžete použít dvěma různými způsoby:

- přímo převzít (u textů je to známé Ctrl-C, Ctrl-V)
- použít jako základ vlastního intelektuálního výtvoru (textu, grafiky, programu, apod.), tj. použijete jen informaci, ale její formu změníte.

Prví druh tzv. přímé citace by měli mít v informatických závěrečných prací jen velmi omezený rozsah (méně než stránka), neboť jejich přínos pro hodnocení práce je diskutabilní. Přesto jsou však případy, kdy jsou vhodné:

1. matematické definice a tvrzení (věty, axiomy)
2. definice termínů z neinformatických oborů (např. společenských věd)
3. citace norem resp. standardů

Citace mají tři základní cíle:

1. určují, co je váš vlastní intelektuální přínos a co jste pouze převzali
2. pomáhají určovat primárního autora (resp. autory)
3. definují kontext vaší práce resp. mohou usnadňovat nalezení dalších souvisejících informací

Jakákoliv vědecká práce nevzniká na zelené louce a tak jsou citace její nezbytnou součástí. V rámci práce běžně navazujeme na existující výzkumy, projekty, technologie apod. Stejně tak se můžeme odkazovat na autority či s nimi polemizovat.

První dva cíle také úzce souvisejí s plagiátorstvím. Pokud v práci použijete myšlenku či údaj bez citování je vaše práce plagiátem. I když se to v obecném mínění vztahuje jen na přímé kopírování, není tomu tak. Přímé kopírování se jen snadněji vyhledává a prokazuje. Je také obtížnější jej kvantifikovat.

V případě přímého kopírování, jež není označeno jako přímá citace, postačuje i relativně malý rozsah (například věta, jeden obrázek, jedna procedura), aby byla práce označena jako plagiát.

Plagiát není možno obhájit a v případě většího rozsahu hrozí i vyloučení ze studia. Za přímé kopírování se považují i případy, kde je změna jen formální (změna slovosledu, náhrada synonym, zkrácení, vložení textové výplně, změna barvy či afinní transformace obrázku).

V případě převzetí myšlenky jde o zjevné plagiování, pokud je tato myšlenka důležitou částí práce (podílí se na splnění cílů).

To, že není plagiátorství odhaleno před obhajobou práce, není důkazem, že se nejedná o plagiát. Pokud je plagiátorství zjištěno později, může vám být odebrán titul i zpětně (a jak jste si jistě všimli, plagiátorství je běžně využíváno v politickém boji).

V každém případě si uvědomte, že plagiátorství je druh krádeže a že ani vy nechcete, aby někdo vaše myšlenky nebo dokonce váš text vydával za vlastní.

## 13.1. Označování citací

Označování citace má dvě části. Za prvé je nutno označit, jaká část práce je citací (rozsah) a jaký je původní zdroj.

Zdroj je vždy určen odkazem na bibliografický záznam, které jsou v případě bakalářské práce uvedeny v kapitole *Použité zdroje* na konci práce. Odkaz může mít různý tvar, ale preferovaný styl je uvedení čísla záznamu v hranatých závorkách. V případě použití našeho latexovského stylu stačí použít příkaz `\cite{id-zaznamu}`. V případě potřeby lze zdroj zpřesnit uvedením např. stránky či kapitoly, jež se uvádí za číslem záznamu (po čárce a ještě před uzavírající hranatou závorkou). V  $\text{\LaTeX}$ u lze využít nepovinný parametr příkazu `cite`.

Označení rozsahu se poněkud liší u přímých a nepřímých citací.

U přímých citací je označení rozsahu kritické. V případě citací, které jsou kratší než odstavec je nutné text vyznačit kurzívou a zahrnout do uvozovek. Odkaz musí následovat hned za označným textem.

U citací v rozsahu odstavce či více odstavců se využívá zvětšení okrajů na levé i pravé straně odstavců (viditelné na první pohled). V  $\text{\LaTeX}$ u lze použít prostředí `quote` nebo `quotation`. Text by měl být navíc v uvozovkách. Kurzíva je možná, ale u rozsáhlejších citací není příliš vhodná. Odkaz se umisťuje na konec posledního převzatého odstavce.

Speciální případ je možný v případě matematických definic a vět. Pokud jsou v rámci kapitoly převzaty jen z jediného zdroje, lze na počátku kapitoly uvést hromadný odkaz například v podobě věty: Všechny definice a věty uvedené v této kapitole jsou převzaty z [X].

V případě převzatých obrázků se odkaz umisťuje na konec popisku. Aby však bylo zřejmé, že se jedná o přímé převzetí (kurzívu ani uvozovky nelze použít) je nutné explicitně vyjádřit, že obrázek byl převzat ze zdroje bez podstatných změn například: (převzato z [X]) nebo (překresleno z [X]).

U nepřímých citací je vyznačování rozsahu volnější. Nejjednodušší je uvádění holé citace na konci vět (před tečkou) nebo konci odstavce (za poslední tečkou). V mnoha případech je ale možné citace uvádět explicitněji a stylistiky je provázen s okolním textem.

příklady:

- zajímavá alternativa je popsána v [x]
- údaj je je převzat z [x]
- použití návrhového vzoru poprvé popsal N.N v [x]
- volně přeloženo z [x]
- řešení bylo navrženo uživatelem N v [x] (vhodné např. pro stackoverflow a podobné zdroje)

Explicitnější vyjádření je nutno použít i v případě, že rozsah citace přesahuje odstavec.

- následující příklad je převzat z [x]
- výčet vychází z [x] je však doplněn o ...

Teoreticky lze podobné řešení využít i u celých sekcí či kapitol (kapitola je zpracována na základě [x]). V tomto případě je však nutné předpokládat, že v dané kapitole není žádná autorská myšlenka, a že autor se snažil najít alternativní pohledy či zdroje (a hodnotit tak, lze pouze autorovu schopnost výběru informací či stylistiky).

Výjimečně lze uvádět i několik citací se shodným či překrývajícím se rozsahem např. *následující specifikace je převzata z [x] a [y]*. To je však tolerovatelné jen v případě, v kdy by oddělení oddělení zdrojů bylo obtížné nebo nepřehledné a spojení nepřináší problémy s intelektuálním vlastnictvím (mají stejného autora či copyright). Zcela nepoužitelné jsou v případě většího rozsahu citace (např. na úrovni sekcí či kapitol)!

V případě obrázků je vhodné uvést explicitnější specifikaci, jak byl originální obrázek pozměněn resp. rozšířen.

Příklad:

- (převzato z [x] a doplněno)
- (převzato z [x], přeloženo)
- (upraveno z [x] pro novou verzi technologie ...)
- (inspirováno diagramem [x])
- (viz také [x] pro data X)

## 13.2. Bibliografický záznam

Bibliografický záznam je datová struktura, jenž má dvě základní funkce:

1. jednoznačné identifikování zdroje
2. určení primární odpovědnosti (typicky je to autor resp. autoři, u webových zdrojů to však často bývá korporace).

Pro každý typ zdrojového dokumentu (zdroje) existuje množina klíčových atributů, které by měly být specifikovány (ne zcela vhodně označované jako povinné) a další, které hrají jen pomocnou roli.

V praxi však může nastat situace, kdy není zřejmé, jaký typ dokumentu pro daný zdroj zvolit resp. nelze zjistit hodnoty klíčových atributů. V tomto případě je nutné improvizovat a snažit se, aby záznam plnil v maximální míře obě funkce.

Struktura bibliografického záznamu je v zásadě dána těmito dimenzemi:

**médium** – základní dělení je na tištěné dokumenty a online dokumenty (dokumenty na elektro-nických nosičích tvoří jakási přechod mezi oběma typy dokumentů)

**samostatnost** – zdroj může být samostatný nebo součást rozsáhlejšího zdroje

**periodičnost** – periodický dokument vychází po jednotlivých částech, přičemž počet částí není předem znám (např. časopis).

## **Tištěné samostatné dokumenty neperiodické**

Typickým příkladem samostatného tištěného dokumentu je kniha či monografie.

Základním zdrojem informací pro bibliografický záznam u knih je tzv. tiráž, tj. soupis vydavatelských údajů uvedený na konci knihy či na stránce za titulem. Využít lze i další zdroje (např. katalogy knihoven či knižní e-shopy, bibliografické záznamy v jiných dokumentech), ale v tomto případě je nutné provádět kontrolu, neboť tyto sekundární zdroje často obsahují chyby.

### **klíčové atributy:**

**ISBN** : ISBN je celosvětový jedinečný identifikátor neperiodických tištěných dokumentů. Pokud ho kniha má, pak je dokument jednoznačně identifikován (a další identifikace už hraje jen sekundární roli). Pomlčky v ISBN nejsou součástí identifikátoru a lze je vynechávat (i když občas se jedno ISBN přiděluje více svazkům). Navíc existují ve dvou podobách ISBN-10 s deseti číslicemi a ISBN-13 s třinácti. Pokud jsou k dispozici oba je vhodnější uvítat ISBN-13 (i když ISBN-10 lze snadno mapovat na ISBN-13).

**název** : název knihy je povinný údaj a měl by být vždy vyplněn. Použit by měl být vždy originální název bez úprav. Jedinou přípustnou úpravou je změna velkých písmen (verzálek) na malá, které by mělo odpovídat pravidlům příslušného jazyka.

**podnázev** : některé knihy mají i podnázev. Někdy je těžké rozeznat, co je název a podnázev. Zde platí pravidlo, že název by neměl obsahovat dvojtečku, tečku, středník apod. Od podnázvu je potřeba odlišit název edice. Podnázev je nepovinný (doporučuji uvádět pokud obsahuje klíčové informace).

**autoři** : v bibliografickém záznamu by měli být uvedeni všichni primární autoři (tj. není potřeba uvádět překladatele, ilustrátory, apod.)

**vydání** : označení konkrétního vydání. Je důležité především tehdy, když není známo ISBN a existuje více odlišných vydání (s různým obsahem)

**nakladatel** : uvádí se jméno nakladatelství, a to především z důvodů odpovědnosti

**místo vydání** : uvádí se jméno města, popřípadě stát, především tehdy pokud není jednoznačné (např. Cambridge) a to ve stručné podobě (např. stačí *United Kingdom*). Podobně stručný by měl být název nakladatelství (tj. bez označení typy společnosti, apod., rodičovské společnosti, apod.) V dnešní době globalizace je tento údaj v mnoha případech nevýznamný (tj. ho lze vynechat, především tehdy pokud je nakladatelství neznámé).

**rok vydání** : rok vydání přesněji identifikuje dokument. Pokud ho nelze zjistit, lze jej nahradit rokem copyrightu (v tomto případě je uvozen znakem c např. c2022)

**edice** : kniha může být vydána v rámci edice. Edici doporučuji neuvádět, výjimkou jsou edice, které jsou všeobecně známé.

**URL** : uvádí se pouze v případě, že je kniha dostupná onlině a to oficiálně a bez poplatků. Uvedení URL v tomto případě usnadňuje její získání (v tomto případě je ale často lepší citovat ji jako elektronickou knihu).

### příklad:

Následující biblografický záznam byl získán z katalogu systému knihovny UJEP (volba Citace Pro v dolní části výpis záznamu).

RASCHKA, Sebastian a Vahid MIRJALILI. *Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow*. Second edition. Birmingham: Packt, 2017. Expert insight. ISBN 978-1-78712-593-3.

Tento záznam splňuje základní požadavky, neboť obsahuje údaje týkající se odpovědnosti i jednoznačnou identifikaci dokumentu (a to jak ISBN tak přesným určením vydání). Zahrnutí podnázvu je vhodné, neboť obsahuje dodatečné informace (jména frameworků). Nakladatelství je uvedeno ve stručné podobě (tj. *Packt*) je uvedeno ve stručné podobě. Nadbytečné je jen uvedení edice (*Expert insight*).

## Online samostatné dokumenty neperiodické

Typickým příkladem je online PDF dokument (včetně elektronické knihy). Dalším příkladem je webové sídlo (*web site*) tj. typicky hierarchický systém více stránek (nikoliv tedy jedna konkrétní web stran).

**medium** : u online zdrojů se jako médium uvádí slovo **online**.

**URL** : klíčový údaj pro online zdroje. Některé systémy (např. Wikipedia) poskytují tj. fixní URL, které odkazují na konkrétní verzi dokumentu, resp. stránek. I když jsou tato URL obecně delší, je nutné jim dát přednost, neboť zaručují jedinečnost.

**název** : název nelze vynechat i když ne vždy je jasné, co je hlavním názvem. V tomto případě je možné využít obsahu elementu title v hlavičce HTML (pokud je zdroj v HTML) nebo jiná metadata (například jak je zdroj pojmenován v odkazu).

**autoři** : autor nebývá u mnoha online dokumentů dohledatelný (a v tomto případě je nutné ho vynechat). Rozhodně však věnujte čas zjištění autorství (může být uvedeno i mimo dokument).

**odpovědná korporace** : typicky je to držitel intelektuálních práv (copyrightu). Důležitý je především v případě, že není znám autor, ale uvádějte ho ve všech případech, kdy je dohledatelný. Většina bibliografických stylů tento atribut nepodporuje resp. ho běžně nezobrazuje. Proto je vhodné pro tento účel využívat atribut *nakladatel* (i když to není totéž).

**verze/čas poslední aktualizace** : nahrazuje rok vydání. V případě, že není použit fixní odkaz, je klíčovým zdrojem informací, jaká z verzí dokumentu byla použita jako zdroj. Online dokumenty se mění často, a tak je vhodné uvádět, co nejpřesnější specifikaci (číslo verze, čas poslední aktualizace). Jen v případě, že dokument není verzován a nelze zjistit přesnější čas poslední modifikace, lze využít vročení (stejně jako u knih může být odhadnuto z copyrightu).

**datum použití** : je to povinný údaj i když důležitý je jen v případě, kdy nelze určit přesnější verzi. Měl by být v ISO formátu tj. ve tvaru RRRR-MM-DD. Toto datum běžně generuje editor bibliografických citací podle data vytvoření záznamu. V každém případě by mělo ležet v časovém intervalu od poslední modifikace zdroje (je-li uvedeno) do data odevzdání závěrečné práce.

#### **příklad:**

### **Dílčí tištěné dokumenty**

U dílčích tištěných dokumentů je typické, že kromě identifikace dílčí části obsahují i identifikaci dokumentu jako celku.

Klasickým příkladem jsou články ve sborníku nebo vědeckém časopise. Kapitoly v knize (monografii) se citují, jen případě, že každou z nich vytvořil jiný autor (či kolektiv autorů)

V zásadě platí tato pravidla:

- uvádí se jen autoři dílčí časti, nikoliv například editoři sborníku nebo časopisu
- uvádí se pozice části v celém dokumentu nejlépe pomocí rozsahu stránek
- pokud má dílčí část vlastní jednoznačný identifikátor (například DOI), není potřeba uvádět identifikátor knihy nebo periodika.

## Dílčí online dokumenty

Tento typ citací se používá pro webové stránky, jež jsou součástí webového sídla například pro konkrétní stránky s dokumentací nebo dokumenty uložené na GitHubu. Pro jiné elektronické dokumenty, pokud nejsou výslovně součástí webového sídla (např. elektronického sborníku) je vhodnější použít záznam samostatného dokumentu (viz výše).

Název stránky je doplněn jménem webového sídla (to je typicky uvedeno v záhlaví každé stránky resp. na hlavní stránce webového sídla). Autoři se vztahují ke stránce zatímco korporátní odpovědnost je typicky vztažena k celému sídlu (pokud jsou známy autoři i korporátní odpovědnost je vhodné uvést oba údaje, vždy však musí být uveden alespoň jeden z těchto údajů). Všechny ostatní atributy se vztahují

### příklady:

What's New In Python 3.9: Summary – Release highlights. *Python 3.9.0 documentation [online]*. Python Software Foundation, October 14, 2020 [cit. 2020-10-15]. Dostupné z: <https://docs.python.org/3/whatsnew/>

Záznam obsahuje název dílčí části a také název celého webového sídla (v kurzívě). Odpovědná organizace je uvedena na místě nakladatele (autor není uveden a tak je tato informace klíčová). Verze je určena datem poslední modifikace (je uvedeno přímo ve tvaru použitém na stránce). Datum citování je povinné, ale v tomto případě nenesе žádnou přidanou informaci (jen to, že citace byla vytvořena jen den po poslední modifikaci. Poslední součástí je URL.

Python nonlocal statement. *Stack Overflow [online]*. Stack Exchange, 2022-03 [cit. 2022-07-27]. Dostupné z: <https://stackoverflow.com/questions/1261875/python-nonlocal-statement>

Struktura záznamu je stejná. Čas poslední modifikace byl určen z informace, že poslední modifikace proběhla před čtyřmi měsíci (je uvedena v ISO formátu, ale odpovídající by byl i údaj například ve tvaru *březen 2022* nebo *March 2022*).

## 13.3. Často kladené otázky

### Co není potřeba citovat?

Obecně platí, že citovat není potřeba znalosti, které jste získali v průběhu studia a to jak při výuce tak i z učebních materiálů (opor, skript). Citovat není potřeba ani zdroj formálních údajů (např. významu zkratek), pokud je lze snadno získat (například na Wikipedii).

To jest není nutné uvádět citaci při uvedení zkratek HTTP (zkratka je všeobecně známá a běžně využívána v mnoha kurzech). Podobně není nutné odkazovat pojmy jako Internet, počítačová síť, programovací jazyk, procesor, apod.

Běžně se také necitují (původní) myšlenky vedoucího práce, pokud si vedoucí práce nevyžádá jinak. Pokud vám zprostředkuje nepůvodní myšlenku, měl by vám pomoci najít originální zdroj (který uvedete v citaci).

Citování není možné v případě, kdy není znám původní zdroj, resp. je v podobě, kterou není možné citovat (lidová řčení, apod.) Pravděpodobnost výskytu takových textů v informatické bakalářské práci je však velmi nízká.

## Jak citovat informace z (podnikových) školení

Pokud se jedná o evidentní výtvar školitele, můžete odkazovat příslušný výukový materiál (i když je neveřejný). Pokud je informace nepůvodní, pak je vhodné citovat primární resp. alespoň dostatečně autoritativní zdroj.

## Jak citovat ústní sdělení?

Ústní sdělení je potřeba citovat jen tehdy, když je od autoritativní osoby v oblasti její odbornosti. Pokud například píšete práci o nasazení databáze, pak je autoritativní osobou například správce databázového systému (který vám sdělí například zkušenosti s nasazením).

Jak je uvedeno výše, ve většině případů se necitují ústní sdělení učitelů, školitelů, vedoucího práce a dalších sekundárních zdrojů.

Pokud citujete ústní sdělení je vhodné s tím danou osobu seznámit či získat alespoň neformální souhlas, neboť sdělené informace nemusí být veřejné.

Navzdory důležitosti ústních sdělení v některých typech prakticky zaměřených prací, není citace ústních sdělení standardizována. Jednoduchý návod nabízí například blog na [citace.com](#) [XXX].

Ústí sdělení je však neověřitelné a nelze ho jednoznačně identifikovat. Proto je lepší pokud se sdělení děje například e-mailem. Citace e-mailové komunikace i dalších netradičních zdrojů shrnuje dokument [XXX].

## Je možno citovat Wikipedii?

Citování Wikipedie se obecně nedoporučuje, neboť se jedná o terciární zdroj (encyklopédia vytvořená na základě druhotných informací) a její kvalita je značně kolísavá.

Na druhou stranu Wikipedia (především v anglické verzi) často obsahuje i hodnotný a jinak jen obtížně dostupný materiál, a tak nelze citování z Wikipedie striktně zakázat.

Základní doporučení pro citování z Wikipedie:

- citujte jen tehdy, pokud nemáte k dispozici primární zdroje (ty jsou často odkazovány přímo z Wikipedie)
- citujte jen kvalitní články (které nejsou označeny jako problematické), které se v oblasti informatiky a matematiky objevují spíše na anglické Wikipedii
- citace z Wikipedie by měli tvořit jen malou část zdrojů (typicky méně než 10 %)

Z Wikipedie rozhodně necitujte články věnované běžně známým technologiím a poznatkům, které jsou běžnou součástí kurzů.



## **14. Zhodnocení**



## **15. Závěr**

Závěr je klíčovou kapitolou, která může nejvíce ovlivnit vaši obhajobu. Základní částí závěru je přehledné shrnutí výstupů práce tj. co jste udělali pro dosažení cílů práce. Je nutné se vyhnout hodnocení, zda tím byli splněny cíle práce, či nikoliv (to je úkol posudků a především komise).



# A. Externí přílohy

Externí přílohy této bakalářské práce jsou umístěny na adrese:

[https://github.com/Jiri-Fiser/thesis\\_ki\\_ujep](https://github.com/Jiri-Fiser/thesis_ki_ujep).

Na úložišti GitHub mohou být uloženy tyto externí přílohy:

- **zdrojové kódy**
- **doplňkové texty** (například jak instalovat aplikaci, manuály aplikace)
- **schémata** (především, pokud se nevejdou na stranu A4 a jejich vytisknutí je tak problematické)
- **screenshoty** (v textu práce lze použít jen omezený počet snímků obrazovky, které navíc nemusí být při černobílém tisku příliš přehledné)
- **videa** (například ovládání aplikace)

V každém případě by to však měli být pouze materiály, které jste vytvořili sami. Materiály jiných autorů uvádějte v seznamu použité literatury (včetně případných odkazů na jejich originální umístění).

V této kapitole stačí uvést pouze základní strukturu úložiště (co se kde nalézá a jakou má funkci) například v podobě tabulky.

---

ki-thesis.pdf	text práce v PDF
ki-thesis.tex	zdrojový kód práce v L <sup>A</sup> T <sub>E</sub> Xu
kitheses.cls	definice třídy dokumentů (rozšířená třída scrbook)
thesis.bib	bibliografická databáze (exportována z citace.com)
LOGO_PRF_CZ_RGB_standard.jpg	logo fakulty s českým textem
LOGO_PRF_EM_RGB_standard.jpg	logo fakulty s anglickým textem

---

Všechny tyto soubory jsou potřeba pro překlad dokumentu (logo stačí jedno v příslušné jazykové verzi).



## B. Další přílohy

Výjimečně může práce obsahovat i další tištěné přílohy. Obecně však dávejte přednost elektronickým přílohám umístěným na GitHubu (tato kapitola tak bude úplně chybět).