



**Vyšší odborná škola a Střední průmyslová  
škola elektrotechnická Olomouc,  
Božetěchova 3**

## **DLOUHODOBÁ MATURITNÍ PRÁCE**

Programování stránek pro Worldee.com

Autor práce: **Jan Dlabaja**

Obor: **Technické lyceum**

Vedoucí práce: **Bc. Martin Meravý**

Školní rok: **2024/2025**

# ZADÁNÍ PRAKTICKÉ ZKOUŠKY Z ODBORNÝCH PŘEDMĚTŮ

**Jméno žáka:** Jan Dlabaja

**Studijní obor:** 78-42-M/01 Technické lyceum s IT specializací

**Třída:** 4L

Ředitelství Vyšší odborné školy a Střední průmyslové školy elektrotechnické Olomouc Vám podle vyhlášky Ministerstva školství, mládeže a tělovýchovy č. 177/2009 Sb., o bližších podmínkách ukončování vzdělávání ve středních školách maturitní zkouškou, ve znění vyhlášky č. 90/2010 Sb., vyhlášky č. 274/2010 Sb., vyhlášky č. 54/2011 Sb. a vyhlášky č. 273/2011 Sb., určuje tuto praktickou zkoušku z odborných předmětů.

**Téma:** Programování stránek pro Worldee.com

**Způsob zpracování a pokyny k obsahu:**

- Popište firmu stojící za portálem Worldee a její působnost v oboru.
- Vysvětlete funkci jednotlivých technologií, které jste při tvorbě stránky použili.
- Popište práci na stránce s uživatelským nastavením.
- Znázorněte architekturu frontendu.
- Dokumentaci zpracujte v editoru LaTeX v připravené šabloně.
- Vytvořte poster (formát PDF, velikost A1, jméno autora, škola, rok) prezentující maturitní práci
- Povinnou součástí práce je registrace a aktivní vystoupení v libovolné soutěži (např. soutěžní přehlídky technických VŠ, perFEKT Merkur, RoboCup apod.). Pokud vhodnou soutěž nenaleznete, bude tento bodem splněn účastí ve školním, případně okresním kole SOČ.
- Všechny body zadání jsou závazné, při jejich neplnění může škola změnit formu praktické zkoušky z dlouhodobé na jednodenní.

**Rozsah:** 25 až 35 stran

**Kritéria hodnocení:** Hodnocení práce probíhá ve třech fázích.

Průběžné hodnocení zohledňuje postupné plnění zadaných úkolů, dodržování termínů, míru samostatnosti žáka. Hodnocení závěrečné posuzuje míru splnění všech požadavků vyplývajících ze zadání práce a funkčnost produktů. Hodnocena je přehlednost, úplnost, srozumitelnost a formální stránka textové části práce. Hodnocení obhajoby práce zahrnuje způsob a srozumitelnost projevu, vzhled prezentace, odpovědi na dotazy.

**Délka obhajoby:** 15 minut

**Počet vyhotovení:** 1x PDF verze (Thesaurus) + 1x poster (Thesaurus)


**Vedoucí práce:** Bc. Martin Meravý

**Oponent práce:** Mgr. Jana Krejčová

**Datum zadání:** 4. října 2024

**Datum odevzdání:** 2. dubna 2025

V Olomouci dne 4. října 2024

  
ředitel školy

VYŠŠÍ ODBORNÁ ŠKOLA 2  
A STŘEDNÍ PRŮMYSLOVÁ ŠKOLA  
ELEKTROTECHNICKÁ  
Božetěchova 755/3, 772 00 Olomouc  
tel. 585 208 121 IČ: 00844012

Zadání převzal dne - 4 - 10 - 2024

  
podpis žáka

Prohlašuji, že jsem seminární práci vypracoval samostatně a všechny prameny jsem uvedl v seznamu použité literatury.

.....

Jan Dlabaja

Prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé práce nebo její části se souhlasem školy.

.....

Jan Dlabaja

Chtěl bych vyslovit poděkování Bc. Martinu Meravému za odborné konzultace a poskytnuté informace. Také bych chtěl poděkovat panu Tomáši Zapletalovi za podklady k první kapitole.

.....

Jan Dlabaja

# Obsah

<b>Úvod</b>	<b>6</b>
<b>1 Firma Worldee</b>	<b>7</b>
1.1 O firmě . . . . .	7
1.2 Historie . . . . .	8
1.3 Produkty . . . . .	10
1.4 Business model a škálovatelnost . . . . .	11
1.5 Konkurence . . . . .	12
<b>2 Použité webové technologie</b>	<b>13</b>
2.1 HTML . . . . .	13
2.2 CSS . . . . .	15
2.3 JavaScript . . . . .	16
2.4 TypeScript . . . . .	18
2.5 Sass, Less, CSS moduly a Tailwind . . . . .	19
2.6 JSX a React . . . . .	20
2.7 MobX a state management . . . . .	22
2.8 API . . . . .	23
2.9 Storybook a mocky . . . . .	24
2.10 Webpack . . . . .	25
2.11 Node.js . . . . .	26
2.12 Next.js . . . . .	27
2.13 Jest . . . . .	30
2.14 I18n a Localazy . . . . .	31
<b>3 Tvorba stránky s nastavením účtu</b>	<b>32</b>

3.1	Vizuální struktura stránky . . . . .	33
3.1.1	Komponenty . . . . .	33
3.1.2	Kompozice . . . . .	36
3.2	Datová struktura stránky . . . . .	45
3.2.1	MobX store a validace dat . . . . .	45
3.2.2	Data, konvertory a API . . . . .	47
3.2.3	Unit testy . . . . .	48
3.2.4	Architektura webu . . . . .	49
3.3	Finální vzhled stránky . . . . .	51
<b>4</b>	<b>Články v časopisu Elektronika</b>	<b>52</b>
	<b>Závěr</b>	<b>53</b>
	<b>Seznam použitých zdrojů</b>	<b>54</b>
	<b>Seznam obrázků</b>	<b>59</b>
	<b>Seznam tabulek</b>	<b>60</b>
	<b>Přílohy</b>	<b>61</b>
1	Plakát . . . . .	61

# Úvod

Asi spousta z vás už někdy vytvořila webovou stránku. Někteří z vás k její tvorbě dokonce využili i jeden z frameworků. Jak hluboko ale tato králičí nora jménem tvorba frontendu vlastně může jít?

Podobnou otázku jsem si pokládal koncem května 2024 po vstupu do startupu Worldee. Z webu jsem uměl jen HTML, CSS a základy JavaScriptu a zhruba tušil, k čemu je Sass. Během léta jsem si nejen tyto znalosti upevnil, ale navíc se naučil i pro mě úplně nové technologie, jako TypeScript, React, MobX, Next.js, nebo práci se Storybookem. Nakonec jsem v těchto technologiích vytvořil stránku s uživatelským nastavením, která bude na webu Worldee ještě několik dalších let. A přesně o tom bude tato práce.

V první kapitole se pokusím objasnit, co je prací a posláním společnosti. Pojem „cestovka 2.0“ nebo „cestovatelská platforma“ je totiž sice docela jednoznačný, ale zdaleka nepokrývá vše, co firma dělá. V další kapitole se pokusím popsat jednotlivé technologie, od těch základních, které většina lidí zná, jako HTML, CSS a JavaScript, přes ty pokročilejší, jako React, TypeScript, nebo Next.js, až po některé nástroje používané na denní bázi, například next-intl a Jest. A nakonec se pokusím tvorbu takové stránky objasnit. S jakými komponentami jsem pracoval, jaké designy jsem skládal a jak vypadá jejich datová struktura. Řada přijde i na zmíněnou ukázkou architektury webu.

Ale nyní – co je to to Worldee, jak vzniklo, a proč má potenciál stát se příštím „jednorožcem“?

# 1 Firma Worldee

## 1.1 O firmě

Co je vlastně to Worldee? Na to jsem si sám dlouho pokládal otázku. Jak jsem již říkal, ve zkratce je to taková cestovka 2.0. Nejlépe to ale asi popíše Tomáš Zapletal, její zakladatel, o němž se v této části práce budu opírat.<sup>[52]</sup>

„Členové týmu Worldee vytváří platformu, která umožňuje lidem jednoduše objevovat svět. Spojením autenticity a jednoduchosti může díky naší platformě kdokoliv objevit téměř jakékoliv místo na zemi a cestovat v tom pravém slova smyslu. Naše poslání je ukázat lidem opravdový svět a pomoci jim odemknout skutečný potenciál jejich osobnosti - skrze zážitky, technologii a dokonalou zákaznickou zkušenost.“

A jak to vlastně funguje? Worldee je takový hybrid mezi sociální sítí a cestovní kanceláří, který se skvěle doplňuje. Itineráře na stránce tvoří sami uživatelé a zkušení cestovatelé, a pokud se vám nějaký líbí, můžete si ho koupit. Worldee vyřídí vše od letenek, ubytování, auta a dalších služeb. Zároveň můžete cestovat s průvodcem nebo na vlastní pěst, s čímž vám pomůže jejich mobilní aplikace.

Cíl je jediný – stejně jako si pod rezervací hotelů představí člověk Booking.com, tak pod itineráři by se vám mělo vybavit právě Worldee.



**Obrázek 1.1:** Logo firmy Worldee.<sup>[41]</sup>

## 1.2 Historie

Vše začalo, když zakladatel firmy Tomáš Zapletal, potřeboval někam nahrát své zážitky z cest.

„Jiné sociální sítě k ukládání cestovatelských zážitků (celých itinerářů) nebyly uzpůsobené. Facebook, Instagram, ani jiná sociální síť. No a vzhledem k tomu, že jsem žádnou takovou, která by mi vyhovovala, nenašel, založil jsem ji sám.“

První prototyp vznikl v roce 2019 a vyvíjela ho externí firma, které Tomáš zaplatil asi dva miliony korun. Po prvním týdnu dostal na stránku přes 1500 uživatelů, které přilákal publikovaný článek na portálu jaknaletenky.cz. K tomu vydal i první video na YouTube kanál Worldee.

Časem bohužel zjistil, že se bez investice Worldee neposune. Společně se spoluzakladatelem Tomášem Nakládalem jezdili po republice a pokoušeli se získat 400 tisíc euro, které nakonec získali od Ondřeje Průši, zakladatele firmy prodávající známé Prusa 3D tiskárny.

První tým jedenácti lidí se v kanceláři poprvé sešel 1. 7. 2020. Vývoj a akvizice nových uživatelů šly pomalu. Největším problémem se ukázala být monetizace.

„Už existovalo mnoho startupů, které měly podobně ušlechtilou myšlenku - pomáhat lidem objevovat svět, pomocí plánovače, nebo cestovatelského deníku. Všechny ale shořely ve chvíli, kdy chtěly nápad monetizovat. Budoucnost tedy byla jasná - nejtěžším úkolem pro nás bude postavit kolem tohoto nápadu udržitelný business.“

Nápad udělat z Worldee sociální síť tak částečně zkrachoval. Dobře nedopadl ani pokus o monetizaci mobilní aplikace. Jako nejlepší řešení se nakonec ukázal prodej cest – pokud se vám nějaký itinerář líbí, můžete si ho koupit jako balíček a bez starostí se na tu samou cestu vydat také. Byl to úspěch a první zájezd se vyprodal do 48 hodin.

A tím se pomalu dostávám do současnosti, kdy jsem nastoupil i já. S Worldee byl ve Skotsku už i YouTuber Kovy se svým přítelem Mírou<sup>[26]</sup> a během toho přišla další, tentokrát stomilionová investice.<sup>[31]</sup> Přispěl do ní zakladatel portálu Bazoš.cz Radim Smička a již



podruhé společnost Kaiperi Venture Capital. Pomalu se rozjíždí expanze do zahraničí a tým se rozrostl na více než padesát lidí. A jak říká název jedné z Tomášem zaregistrovaných firem – „The sky is no longer the limit (s. r. o.)“.<sup>[1]</sup>

## 1.3 Produkty

Pokud se s Worldee rozhodnete vydat na cestu, můžete si vybrat ze tří variant.<sup>[41]</sup>

### Cesta s průvodcem

- cesta s Čechem, který je se zákazníkem celou dobu od odletu po přílet
- v malé skupince (8 - 12 osob)
- v konkrétní termín
- musí se naplnit minimální kapacita cesty

### Cesta za expatem

- cesta za Čechem, který se z ČR přestěhoval do zahraničí, kde si postavil/koupil malý rodinný hotel
- individuálně - je možné jet i v jedné osobě
- libovolný termín
- expat se o zákazníka celou dobu stará - vyzvedne ho na letišti, vaří mu, seznamuje ho s místními, radí mu, jaká místa navštívit. Stará se o to, aby destinaci opravdu autenticky poznal

### Self-guided itinerář

- jakýkoliv itinerář lze zakoupit také jako self-guided itinerář a na cestu se vydat na vlastní pěst
- zákazníka neprovádí fyzická osoba, ale zkušený cestovatel zprostředkovaně přes aplikaci, do které už předem vše napsal
- na itinerář je možné jet kdykoliv, odkudkoliv na světě

## 1.4 Business model a škálovatelnost

Ještě pár let zpátky většinu modelu tvořilo premium, kterým si uživatel mohl vylepšit účet a získat několik výhod včetně většího místa na disku pro své cesty. To se ale ukázalo jako nedostatečné, a tak je dnes příjem tvořen prodejem itinerářů, za které si Worldee bere 25% marži.

Aby mohly cesty s průvodcem vůbec fungovat, jsou všichni průvodci placení (např. 2000 Kč na den) a vedení jako Travel Buddy. Těmi se může stát kdokoliv, kdo vyplní přihlašovací formulář a projde náborovým procesem. Podobné je to s expaty, jejichž odměna je již přímo zahrnuta v ceně služeb. Protože má Worldee licenci a pojištění proti úpadku, mohou portál průvodci použít pro legalizaci svého podnikání a zároveň si stále budovat svoji značku. Ačkoliv bude jejich škálovatelnost pro expanzi do zahraničí náročná, protože budou potřeba stovky lidí různých národností, žádná platforma, která by něco podobného nabízela, na globální úrovni zatím neexistuje. A přitom mají tito průvodci vysokou přidanou hodnotu, protože pomáhají zákazníkům překonat jazykovou bariéru a tím otevírají Worldee novým cílovým skupinám.

Daleko lépe škálovatelnější jsou cesty na vlastní pěst. Sám Tomáš o nich dokonce mluví jako o „svatém grálu“. Vše by mělo do budoucna fungovat automaticky a data o hotelech, letenkách a autech brát od třetích stran. Všechny itineráře už Worldee nyní umí automaticky překládat do všech světových jazyků. Stačí, aby si zákazník vybral termín, a pak se nechal provádět mobilní aplikací v kapse.

## 1.5 Konkurence

Ačkoliv je Worldee v Česku poměrně unikát, v zahraničí už existuje několik firem, které jsou s podobným projektem částečně úspěšné.

### **TripLegend**

Jedná se o lokální německou cestovní kancelář založenou před čtyřmi lety. Nabízí cesty s průvodcem a spolupracují s cestovními agenturami, nedávno se začali pokoušet i o self-guided cesty. Zatím fungují v režimu break-even s jednotkami cest měsíčně, navíc bez jakéhokoliv vývojového týmu. Nicméně jsou důkazem, že podobný produkt může levně fungovat i na vysoce konkurenčním trhu.

### **WeRoad**

Tato cestovka má už mnohem větší dosah než TripLegend. Ačkoliv dělají jen cesty s průvodcem doma a na některých západních trzích, a v balíčku ani nenabízí letenky, mají miliardový obrat a dokázali, že produkt může uspět i na chudším trhu, jako je Itálie.

### **FlashPack**

Tento britský projekt, který se po pandemii dostal díky investorovi z insolvence, funguje hlavně na anglicky mluvících trzích. Staví si na prémiových službách a cenách, ale je možné, že se části z nich po saturaci trhu vzdají a díky penězům z marží expandují i na levnější trhy. Nicméně nemají ani vývojový tým, ani self-guided itineráře.

### **PolarSteps**

Jako poslední z firem tu uvedu projekt z Nizozemska. Podobně jako Worldee nabízí cestovatelský deník, ze kterého si ale navíc můžete nechat udělat fotoknihu. Zatím to pro Worldee může znamenat jen inspiraci pro další monetizaci, ale pokud své itineráře začnou prodávat, mohla by to být vzhledem k jejich velikosti potencionální hrozba.

## 2 Použité webové technologie

Než se dostanu k tvorbě samotné stránky, je potřeba si rozebrat technologie, které jsem k její tvorbě potřeboval a které mi značně ulehčily život. Půjdu postupně od těch nejdůležitějších až k těm méně známým.

### 2.1 HTML

Na webu neexistuje důležitější technologie, než právě HTML. Bez tohoto značkovacího jazyku by na našem webu chyběl jakýkoliv obsah, slouží totiž k definování struktury stránky. Jeho syntax se skládá z párových (`<span>potomek</span>`) a nepárových (`<img/>`) značek, které se liší podle toho, zda v sobě obsahují potomky.

Zde je seznam několika základních elementů:

Element	Popis
span	Jednoduchý text
div	Prázdný element
p	Odstavec
button	Tlačítko
img	Obrázek
br	Nový řádek
h1, h2, h3, h4	Nadpisy
ol, ul	Číslovaný/nečíslovaný seznam
a	Odkaz
table	Tabulka

**Tabulka 2.1:** Seznam základních elementů v HTML.

Každý validní HTML dokument by měl mít dvě části. První je hlavička (head), ve které jsou definované informace, které na stránce nejsou vidět, jako její název, lokace skriptů

a stylů, jazyk, font, nebo metadata. Druhá část je tělo (body). Tady už můžete používat všechny dříve zmíněné elementy a zobrazovat obsah samotné stránky.

Každý element má několik atributů, které můžete nastavit. *Id* slouží jako unikátní identifikátor a *class* zase reprezentuje skupinu prvků. Pomocí *href* můžete prvku nastavit url adresu, a eventy jako *onclick* nebo *onmouseover* zase po splnění podmínky (kliknutí na element, přejetí myší) spustí definovaný skript.

V raných verzích se HTML dalo použít nejen k definování struktury stránky, ale i k jejímu stylování a pozicování jednotlivých elementů. Postupem času se ale toto paradigma stalo zbytečně komplexním, a tak se struktura a prezentace oddělily a vzniklo CSS.

## 2.2 CSS

Styly vám na stránce umožní upravit téměř vše – měnit barvy, odsazení, pozice, velikost, .... Dají se psát buď do samostatných souborů s příponou `.css`, do `<style></style>` elementu, nebo do `style` atributu. Pro výběr konkrétních elementů se používají tzv. selektory. Zde uvedu příklad na tlačítku.

Selektor	Popis
<code>button {}</code>	Změní všechny <code>&lt;button/&gt;</code> elementy na stránce
<code>.blue-button {}</code>	Změní všechny elementy se třídou ( <i>class</i> ) <code>save-button</code>
<code>#top-button {}</code>	Změní jeden konkrétní element s identifikátorem ( <i>id</i> ) <code>top-button</code>

**Tabulka 2.2:** Příklady CSS selektorů a jejich funkcí.

Do složených závorek následně můžete psát konkrétní styly. Například `background: red;` nastaví pozadí prvku na červenou, `margin-right: 10px;` ho zase odsadí zprava o 10 pixelů, apod. Pokud chcete na element odkázat nepřímě, můžete za selektorem použít symboly `>` (zaměřuje potomka) nebo `<` (zaměřuje rodiče). Můžete také přidávat pseudo třídy jako `:hover` (změní element, pokud na něj najede kurzor myši), `:first-child` (aplikuje se na prvního potomka rodiče) a mnohé další. Pro potřeby pokročilých stylů se staly rozšířenými i pseudo elementy (`::before`, `::first-line`, ...). Jejich prací je vybrat konkrétní část elementu, například jeho potomka nebo první řádek.

Důležitou znalostí je také chápat box model a jeho jednotlivé části, které obklopují každý element. Jsou tři – vnitřní okraj (`padding`) určuje místo mezi obsahem a okrajem elementu (`border`), a vnější okraj (`margin`) zase oblast mezi elementem a ostatními prvky. U všech se dá nastavit velikost shora, zdola, zprava a zleva.

Pro responzivní design stránek, které se přizpůsobí jakémukoliv obrazovce, vznikl počátkem minulého desetiletí flexbox. Ten konečně zjednodušil pozicování položek, které umožnil dávat do řádků nebo sloupců, zarovnávat je na začátek, střed, nebo konec, nastavovat mezi nimi mezery nebo je roztahovat pro zaplnění kontejneru.<sup>[17]</sup> Touto dobou také začala masová podpora media queries, které umožňují dynamicky měnit styly na základě aktuální velikosti stránky nebo vlastností displeje.

## 2.3 JavaScript

Nástup JavaScriptu umožnil webovým vývojářům udržovat na stránkách aktuální data a přidat interaktivní prvky, zlepšující uživatelský zážitek.

Ačkoliv se to tak nemusí podle názvu zdát, JavaScript má s Javou jen pramálo společného. Vytvořil ho v roce 1995 během deseti dnů zaměstnanec firmy Netscape, tehdy ještě pod jménem Mocha a LiveScript. Java se do jména dostala až kvůli popularitě stejnojmenného jazyka. Standardy pro něj vyvíjí organizace Ecma International, podle níž se značí i jeho verze (ES5, ES6, ...).<sup>[25][51][3]</sup>

Protože se jedná o jediný jazyk, který umí prohlížeče číst, a zároveň je jednoduchý na pochopení i programování, docela rychle se rozšířil. O to více, když v roce 2009 vzniklo Node.js a umožnilo ho používat i mimo webové prostředí. Od té doby v něm lze napsat opravdu téměř vše a i díky velkému množství dostupných knihoven (například v online repozitáři NPM) se těší vysoké popularitě.

JavaScript má dva typy proměnných – `let` (dá se měnit) a `const` (neměnná). Do nich lze přiřazovat různé zabudované typy jako `boolean`, `number` (64bitové desetinné číslo), `bigInt`, `string`, `array`, `date`, objekt a několik dalších. Existuje i tři prázdné typy, jako je `undefined` (bez přiřazené hodnoty), `null` (prázdná hodnota) a `NaN` (Not-a-Number, doslova *není číslo*). Typy se nicméně kontrolují až při běhu, takže se na špatné přiřazení často přichází, až když je příliš pozdě.<sup>[6][7]</sup>

Pokud píšete podmínky, musíte dávat pozor na porovnávání. Dvě rovnítka porovnávají typy, tři zase hodnoty. Záměnou prvního rovnítka za vykřičník vznikne negace. `&&` značí logický operátor AND a `||` zase OR. Některé typy (`undefined`, `null`, `NaN`, nebo prázdné řetězce) se označují jako *falsy* a v podmínkách reprezentují hodnotu `false`. Dále můžete používat ternary operátory (`x ? y : z`, pokud je `x` pravda, vykoná se `y`, jinak `z`) nebo jeho zkrácenou verzi (`x && y`, pokud je `x` *falsy*, vykoná se `y`), která se převážně používá v Reactu pro podmíněné renderování.

Existuje i několik druhů cyklů. V obyčejném `for` cyklu nastavujete počáteční hodnotu, krok a podmínku, přičemž všechny argumenty jsou nepovinné. Nechybí ani podmíněný `while` a pro iteraci elementů v seznamu se nejlépe hodí `forEach()`.



Jazyk umožňuje funkcionální i objektové programování. Třídy stejně jako u ostatních jazyků podporují konstruktory, dědičnost, zapouzdření, nebo interní metody (prefix #). Objekty lze volně kombinovat s funkcemi. Ty mohou požadovat argumenty, které jsou buď kopií původních hodnot (primitivní typy), nebo referencí na původní proměnnou (objekty, pole, funkce). Každá funkce má návratový typ, který je v základu `undefined`.<sup>[5][2]</sup>

## 2.4 TypeScript

TypeScript vznikl jako reakce na některé problémy s vývojem v JavaScriptu, primárně dynamickou kontrolu typů. Používá se pouze během vývoje a při bundlingu se transpiluje do JavaScriptu, aby ho prohlížeče dokázaly číst a spustit.

Již zmiňovanou výhodou a hlavním prodejním bodem je typová podpora. Výhodou dynamicky typovaného jazyka sice může být vyšší rychlost vývoje, ta ale (společně s programem) padá, jakmile se nesprávný typ dostane do produkčního prostředí. Pokud to tak není podle použití zřejmé, musíte typ přímo definovat nebo použít `any`, které bude TypeScript ignorovat. Po spuštění pak staticky analyzuje celý kód a pokud najde chybu, odmítne ho sestavit.

S typy lze dělat mnoho operací. Spousta z nich je postavená na obecných typech, umožňujících bezpečně použít více než jeden typ.<sup>[10]</sup> Například `Omit<Typ, Parametry>` umožňuje z typu odstranit určité parametry a pomocí `Pick` je zase přidat. `Partial<Typ>` zase narozdíl od `Required` všechny parametry udělá nepovinné, `Readonly` je nastaví jen pro čtení a `NonNullable` odstraní všechny hodnoty s `null` nebo `undefined` typem. S typy lze dělat i logické operace, jako průnik nebo sjednocení.

S udržováním všech typů vám mohou pomoci interfaci, ve kterých můžete například definovat parametry pro funkce a metody a následně je (jako šablony) znovupoužít na dalších místech v projektu.

TypeScript přidává i rozšířenou podporu enumů, které umožňuje definovat sadu pojmenovaných konstant. Hodnota každé položky může být `number` nebo `string`.<sup>[4]</sup>

V poslední řadě jazyk přidává nové funkce pro třídy. Nad metodami nyní můžete volat dekorátory modifikující jejich chování, nebo vytvořit nové se stejným jménem ale jinými argumenty a metodu takzvaně „přetížit“. Zatímco dekorátory mají prefix `@`, u interních funkcí nutnost prefixu odpadá a nahrazuje ho klíčové slovo `private`. Pro ještě lepší zážitek z OOP lze používat abstraktní třídy, které nelze instancovat, ale jejichž implementaci mohou využít všichni její potomci.

## 2.5 Sass, Less, CSS moduly a Tailwind

Problémy s vývojem se nevyhnuly ani CSS. O jejich řešení se snaží různé preprocesory, konkrétně Sass<sup>[15]</sup> (a jeho verze s CSS syntaxí SCSS) nebo Less<sup>[13]</sup>. Stejně jako TypeScript se jedná pouze o nástavbu, která se v tomto případě převádí zpět do CSS.

Jeden z problémů, které preprocesory řeší, je znovuvyužitelnost stylů. Nyní se dají organizovat do mixinů a ty následně používat jako jeden styl. Stejně jako funkce umožňují předávat parametry a s těmi dále pracovat. Například zapomocí nově přidaných podmínek a cyklů. Konzistenci webu napomáhají také proměnné, do kterých se nejčastěji ukládají barvy nebo velikosti písma a prvků a které se dají globálně používat v celém projektu.

Ať už se preprocesor rozhodnete používat nebo ne, jistě vám život ulehčí CSS moduly.<sup>[18]</sup> Jak stránka roste, rostou i její styly, a může se jednoduše stát, že nová třída bude mít stejné jméno jako úplně jiná třída na druhé straně projektu. Pokud se to stane, styly se mohou navzájem ovlivňovat a velmi jednoduše vzhled stránky rozbít. Tento nástroj při skládání projektu všem třídám přidá do jména identifikátor, díky čemuž je každé jméno lokální.

Za zmínku ještě stojí Tailwind, framework, který všechny zmíněné technologie dokáže kompletně nahradit. Místo toho, aby vývojář tvořil třídy s vlastními styly, má již Tailwind třídy předdefinované a ty se následně kombinují mezi sebou. Nespornou výhodou je eliminace externích souborů se styly a rychlejší vývoj, výčet nevýhod zahrnuje horší čitelnost a nepřehlednost kódu kvůli dlouhým obsahům tříd. O tom, zda je lepší používat Tailwind nebo CSS se dodnes kvůli jejich rozdílnému pohledu na stylování stránek vedou spory.<sup>[24]</sup>

## 2.6 JSX a React

JSX je syntax, která umožňuje zbavit se statických HTML souborů a začít psát UI kompletně v JavaScriptu, a to se všemi známými tagy, které jste používali doposud.<sup>[42]</sup> Na stránce tak můžete jednoduše používat dynamické hodnoty a ani pro definici chování eventů si už nemusíte zakládat nové skripty.

Místo celých stránek nyní můžete tvořit samostatné komponenty, které jsou izolované a znovupoužitelné. Ty následně můžete spojovat do kompozic reprezentujících části stránek a ty zase do stránek samotných. Celá struktura je tak mnohem modulárnější a chování komponent předvídatelnější.

K JSX je standardně potřeba knihovna React, která všechny elementy na stránce vykresluje. Každý z nich má svůj stav, a stejně jako u funkcí, i elementy by měly být deterministické a pro každou kombinaci vstupů mít maximálně jeden výstup. Stav se dá dále rozdělit na vnitřní a vnější podle jeho přístupnosti. Pokud se kterýkoliv z nich změní, React na to zareaguje a element překreslí, tentokrát s novými daty.

Protože se s každým dalším překreslením znovu spustí celá funkce, která element definuje, nelze vnitřní stavy uchovávat v mutabilních ani globálních proměnných.<sup>[38]</sup> Místo toho React používá hooky, speciální funkce, které se dají během renderu volat. Pro uchování stavu mezi rendery se používá hook `useState()`. Při změně jeho obsahu se zavolá překreslení, ale hodnota se přenesou. Pokud nic překreslovat nechcete, můžete hodnotu uložit do `useRef()`<sup>[39]</sup>.

Jak ale React vlastně funguje?<sup>[21]</sup> Pokaždé, když se změní stav, vytvoří virtuální kopii DOMu a v něm provede změny.<sup>[48]</sup> Následně ho porovná s původním virtuálním DOMem, zjistí, co se změnilo, a následně se změny pokusí v jednom balíku aplikovat na reálný DOM. Aby při porovnávání ušetřil výkon, kromě změněného elementu překreslí i všechny jeho potomky. Nové elementy tvoří dynamicky pomocí JavaScriptu, proto vidíte jen prázdnou stránku, pokud ho zakázete.

Každý z elementů má určitou životnost. Ta začíná po připojení k DOMu (prvním vykreslení). Následně se kvůli změně stavů několikrát překreslí, než nadejde jeho čas a je ze stránky odpojen. Na to vše se dá reagovat `useEffect()` hookem.

Několik hooků má na starost zase optimalizaci výkonu. Pro zabránění zbytečnému přepočítávání hodnot se používá hook `useMemo()`, který vrací její naposled vypočítanou verzi. Pro funkce se zase používá `useCallback()`.

## 2.7 MobX a state management

Jednou z nevýhod Reactu je správa stavů. Pokud jich máte v komponentě několik, špatně se s nimi pracuje a data se nedají testovat skrz unit testy. Zároveň jediný způsob jak je sdílet, je dědit je z nadřazené komponenty.

Existují desítky knihoven<sup>[27]</sup>, které vám umožňují efektivněji spravovat stav. Redux (první a nejpoužívanější), Hookstate (minimalistický, několik druhů stavů), Recoil (od Facebooku), nebo právě MobX, který jsem při tvorbě stránky použil. Ačkoliv se funkce každého z nich jmenují úplně jinak, jsou si dost podobné a mají jediný úkol. Uchovat stav mimo React.

MobX to dělá takto. Vytvoříte si třídu a do ní vložíte všechny stavy, které ve vaší komponentě potřebujete – aktuální text, barvu, obrázek, atd. A ke všem přidáte `@observable` dekorátor, který funguje podobně jako `useState()` hook. Pro aktivaci uvnitř Reactu ještě musíte všechny kompozice a komponenty, které MobX využívají, obalit `observer()` funkcí.

To ale není vše, co umí. Pokud potřebujete změnit několik observable proměnných najednou a předejít zbytečným překreslením, můžete jejich aktualizaci obalit do metody `runInAction()` a naznačit MobXu, aby vše vykonal najednou. MobX disponuje i funkcí podobnou již zmiňovanému hooku `useMemo()` - dekorátor `@computed`<sup>[8]</sup>, který se přepočítá pouze tehdy, pokud se některá ze závislostí (použitých observable proměnných) změní.

## 2.8 API

API (Application Programming Interface) je způsob, kterým mezi sebou mohou komunikovat dva rozdílné systémy. Vlastně jde o soubor pravidel, které si oba definují a které následně používají pro dorozumívání.<sup>[50]</sup>

Jedna forma API se používá i na webu. Stojí na protokolu HTTP a jeho metodách GET (získání dat), POST (odesílání dat), nebo PUT (ukládání souborů na server). Můžete v nich používat libovolný formát, nejčastější je HTML, JSON a XML.

Komunikace mezi klientem a serverem vypadá většinou takto - klient odešle GET požadavek na server a ten mu vrátí HTML stránku nebo jiná data. Pokud chce klient něco uložit, odešle data metodou POST nebo PUT, server s pomocí jiné formy API uloží data do databáze a klientovi vrátí odpověď obsahující úspěšnost požadavku.

## 2.9 Storybook a mocky

Doted' jste mohli své komponenty testovat a vidět jen na finální stránce. To je ale poněkud nepraktické, protože musíte čekat na kompilaci celého projektu, používáte reálná data a komponenty nejsou izolované. Pro to se používá knihovna Storybook. V ní si můžete ve webovém rozhraní všechny komponenty prohlédnout a jednoduše i bez znalosti kódu měnit jejich stav. Tím usnadníte práci i designerům, produktovým manažerům a testerům.

Jediné, co potřebujete, je (kromě instalace) vytvořit pro každou komponentu soubor s příponou `*.stories.*`. V něm definujete stavy komponenty, které půjdou měnit, a ke kterým Storybook automaticky přiřadí odpovídající vizuální prvek (například řetězce můžete psát do textového pole, seznamy vybírat z dropdownu a rozsah vybírat posuvníkem) a následně mu řeknete, kterou komponentu má zobrazit.

Pro využití Storybooku naplno je ještě třeba přerušit komunikaci se serverem. Každý projekt na to jde po svém. Pokud například tlačítko volá API skrze `onClick()` metodu, pak ji stačí jednoduše zpřístupnit zvenku a nechat ji prázdnou. Ne vždy je to ale takto jednoduché a kód komponent by se kvůli Storybooku nikdy neměl modifikovat. Lepší způsob je tedy vytvořit mocky<sup>[9]</sup>. Stačí vytvořit úplně stejnou třídu, jako ta, kterou chcete „mocknout“, jen s falešnými metodami, které místo volání na server vrací falešná data. Ní pak nahradíte reálnou implementaci třídy. Kromě Storybooku můžete mocky používat i v unit testech, ke kterým se ještě dostanu.



## 2.10 Webpack

Máte hotové nějaké komponenty, dosadili jste si je do stránek a chcete vše publikovat. Protože ale JavaScript není kompilovaný jazyk, každý uživatel teď dostane de facto celý váš zdrojový kód. Kromě toho, že si ho kdokoli může přechíst, trvá odesílání kódu kvůli jeho velikosti déle a některé novější funkce JavaScriptu nemusí daný prohlížeč znát. Z tohoto důvodu se používají bundlery jako je Webpack<sup>[16][20]</sup>, Turbopack nebo Vite, a jejich loadery.

Na začátku bundlingu si Webpack najde vstupní soubor (většinou index.js) a rekurzivně projde všechny jeho importy<sup>[11][46]</sup>, čímž si poskládá vlastní graf závislostí<sup>[12]</sup>. Následně aktivuje loadery. Ty mu řeknou, jak pracovat s jednotlivými soubory. Mezi nejpopulárnější patří ts-loader (TypeScript) a Babel (JavaScript). Oba kód převedou na starší verzi ECMAScriptu, díky čemuž je následně spustitelný na většině prohlížečů bez ztráty funkčnosti. Existují ale i loadery pro CSS, Sass, obrázky, videa, a vlastně jakýkoliv soubor, na který si vzpomenete. I dříve zmiňované CSS moduly jsou vlastně jen jedním z mnoha Webpack loaderů.

Následně jsou ze souborů odstraněny nepotřebné znaky (mezery, nové řádky, komentáře), nepoužité prvky, a celý se minifikuje. Výsledkem je jeden nebo více balíčků, které se klientovi dle potřeby při průchodu webem spouští.

Webpack má bohatý ekosystém nejen loaderů, ale i pluginů. Dají se instalovat skrze konfigurační soubor webpack.config.js, který se exportuje jako objekt a umožňuje definovat i různá nastavení. Jedním z nejpopulárnějších pluginů je Bundle Analyzer, který vám v uživatelsky přívětivém rozhraní ukáže, z čeho je bundle poskládaný a co na něm zabírá nejvíce místa. Pro statickou analýzu souborů dále existují pluginy s podporou pro ESLint nebo TypeScript.

## 2.11 Node.js

Aby byl JavaScript v prohlížeči bezpečný, musí mít nějaká omezení. Například nemít možnost otevírat a číst vaše soubory v počítači, nebo se dostat mimo stránku do ostatních záložek a aplikací. Tím se jazyk stává omezeným pouze na tvorbu frontendu. To se ale změnilo s příchodem Node.js.<sup>[43]</sup>

Node.js je open-source projekt, který má v sobě zabudovaný ten samý engine (V8), co prohlížeče jako Chrome, Edge, nebo Brave používají pro spouštění JavaScriptu. Kromě toho mu ale umožňuje manipulovat se systémem, takže v něm můžete bez problémů napsat celý webový server nebo cokoliv jiného, co lze dělat i s jinými vysokoúrovňovými jazyky.<sup>[14]</sup>

Ačkoliv zde neexistují objekty pro manipulaci se stránkou nebo oknem (např. document nebo window), má několik vlastních modulů. Http pro tvorbu serveru, fs pro práci se soubory, nebo os s informacemi o systému. Díky svojí architektuře má i vlastní správu událostí. Node.js je zároveň důvod, proč vznikla databáze a následná správa knihoven NPM (Node Package Manager).

## 2.12 Next.js

At' už je to směrování, optimalizace obrázků, nebo SEO, o většinu těchto věcí se musíte postarat sami. A často to není nic jednoduchého. Proto existuje framework Next.js<sup>[32]</sup>, nástavba Node.js s podporou Reactu, který dokáže celý web pozvednout s minimem znalostí na úplně novou úroveň. Popsat ho celý by bylo na další dlouhodobou práci, proto tady zmíním alespoň základní výhody. Zároveň doporučuji projít si tuto příručku (<https://nextjs.org/learn>)<sup>[33]</sup>, která je všechny vysvětluje do hloubky a ještě vás naučí pár věcí navíc.

### Routing

Funkci, kterou využijete na každé stránce, je routing a tvorba cest. Už nemusíte endpointy udržovat v samostatných souborech – teď se budou tvořit podle adresářové struktury, neboli tak, jak je umístíte do složky jménem */app*. Takže třeba */app/home/explore/page.tsx* (každá stránka musí mít tento soubor, něco jako *index.html*) bude dostupná na url adrese *www.example.com/home/explore*.

Celý framework je napsaný tak, aby bylo SPA na prvním místě. SPA znamená Single Page Application a stojí na tom, že místo, aby se po změně odkazu načetla znovu celá stránka, stačí jen stáhnout a změnit její část – uživatel tak nic nepozná a web se chová jako aplikace.

Ne všechny části jsou ale pokaždé potřeba znovu stahovat a načítat. Některé, například hlavička, budou po celém webu na stejném místě. Stejně jako má Next.js *page.tsx* soubory, tak má i *layout.tsx*, do kterého tyto neměnné komponenty a kompozice můžete zahrnout a ušetřit tak síťový výkon.

Pokud potřebujete jednoduše vytvořit API endpoint, můžete využít soubor *route.tsx*. Pro chyby je *error.tsx*, pro neplatné cesty *not-found.tsx*, a dokonce existuje i *loading.tsx*, který slouží jako placeholder a uživatel ho uvidí, pokud stránka musí čekat na nějaká data.

Pokud je část cesty dynamická, dá se umístit do složky, jejíž název je ohraničený hranatými závorkami. To se hodí například u uživatelských profilů. Jednoduché závorky zase slouží k logické organizaci, která se neprojeví v URL.

## Serverové a klientské komponenty

Pokud tvoříte komponentu, máte dvě možnosti, kde ji vykreslit<sup>[35]</sup> – na serveru<sup>[36]</sup> nebo na klientovi<sup>[34]</sup>. Obojí má své pro a proti. Na serveru můžete používat asynchronní operace a volat tak na API přímo v komponentách, nicméně nelze pracovat se stavem a hooky jako `useState()` a `useEffect()`.

Klientské komponenty se chovají stejně jako v Reactu, jen s tím rozdílem, že se jejich prvotní stav předkreslí na serveru a následně jsou odeslány do prohlížeče, kde se jich React ujme a pomocí tzv. hydratace jim vrátí interaktivitu. Díky tomu na stránku nemusíte čekat a i s vypnutým JavaScriptem vidíte alespoň její statický obsah.<sup>[40]</sup>

## Optimalizace fontů a obrázků

Pokud používáte na své stránce jakýkoliv font, který není v základu (například od Googlu), ukáže se stránka na klientovi nejdříve se základním fontem, a až poté, co se z dané adresy stáhne, se aplikuje ten správný. To může způsobit posun elementů, protože každý font má trochu jinou velikost a vzhled. A Google, ta samá firma, od které font nejspíš máte, vám pak může snížit celkové SEO hodnocení a zobrazovat vaši stránku méně lidem. Next.js ale fonty stáhne již při bundlingu a pošle vám je spolu s celou stránkou, takže je prohlížeč použije ihned.

Stejně jako fonty musíte mít optimalizované i obrázky. Na ty má Next.js vlastní `<Image/>` komponentu. Ta zabraňuje posunu elementů při načítání stránky, posílá velikosti obrázků úměrné k velikosti použitého displeje, šetří výkon tím, že obrázky načte až poté, co jsou vidět na displeji, a snaží se je všechny odesílat v moderních formátech, jako WebP nebo AVIF.

## Streaming

Next.js podporuje i streaming, který znáte ze všech dnešních sociálních sítí. Jde o postupné posílání obsahu stránky, například při jejím posunutí. K aplikování na stránky stačí obalit danou komponentu pomocí `<Suspense/>` komponenty. Na podobném principu funguje i již zmiňovaný *loading.tsx*, který `<Suspense/>` aplikuje na celý soubor. K této komponentě se váže i tzv. Partial loading, který kombinuje serverové a klientské renderování

a časem má potenciál stát se standardním způsobem pro tvorbu stránek v tomto frameworku.

## Hooky

Existuje několik hooků. Pro získání aktuální cesty se používá `usePathname()`. To můžete použít například u zvýraznění aktuálního odkazu v menu. `UseSearchParams()` zase podle klíče v URL najde jeho hodnotu (třeba `id`) a `useRouter()` umožní přistupovat k routeru a měnit URL nebo stránku znovu načíst. Kromě hooků máte přístup i k hlavičkám požadavky a cookies.

## Metadata

Metadata jsou informace, které se obvykle vkládají do hlavičky a SEO roboti díky nim dokážou pochopit obsah stránky. Zatímco ve staré verzi frameworku jste mohli metadata vkládat přímo do `<Head/>` komponenty, nyní jdou jednoduše exportovat jako objekt skrze vestavěnou funkci.

## 2.13 Jest

Když už je stránka hotová, musíte zaručit, že její aktuální a budoucí funkce budou fungovat, a že pokud najdete a opravíte bug, neobjeví se tam za pár měsíců znovu. Je několik možností, jak stránku a komponenty testovat<sup>[49]</sup>. Vizuální, které pořizují fotky komponent a porovnávají jejich vzhled, end-to-end, které přímo interagují s prohlížečem, a nebo nějaká forma unit testů, které zajišťují, že i když se může někdy rozpadnout vzhled, vždy budou alespoň sedět data. A právě ty má Jest na starost.

V každém souboru označeném *\*.spec\** můžete pomocí funkcí `describe()` a `it()` vytvářet unit testy. Do metody `expect()` následně pošlete věc, kterou chcete porovnat, a do metod `toBe()` (porovnávání hodnot proměnných) nebo `toEqual()` (porovnávání objektů) zase hodnotu, které by se to reálně mělo rovnat. A pokud se to rovnat nebude, test spadne, a vám o tom dá ihned vědět.

Zároveň můžete pracovat i s funkcemi<sup>[44]</sup>. Bud' se dají vytvářet mocky, a nimi nahrazovat funkce nebo celé moduly, nebo, pokud vám záleží na jejich implementaci, můžete použít `spyOn()`. Poté se můžete podívat, kolikrát a s jakými parametry se funkce zavolala.

A Jest umí ještě jednu zajímavou věc. Ukáže vám, jaké máte pokrytí projektu testy<sup>[23]</sup>. A ačkoliv se asi nikdy nedostanete ke sto procentům, čím více testů, tím méně budete řešit v budoucnosti chyb.

## 2.14 I18n a Localazy

I18n je zkratka slova Internationalization (mezi i a n je 18 písmen)<sup>[45]</sup>. A přesně to budete potřebovat, pokud plánujete projekt posunout do zahraničí a přidat více než jeden jazyk. I na tohle existují frameworky. A ten nejpobulárnější se jmenuje i18next<sup>[47]</sup>.

Lokalizace se do něj zapisují jako objekt, ve kterém má každý z jazyků svůj klíč. Klíč má v něm i každé slovo a jeho překlad reprezentuje hodnotu. Jeden klíč zároveň může mít pod sebou celou skupinu překladů, díky čemuž je můžete logicky řadit. Pokud je pak chcete použít v projektu, stačí nastavit aktuální jazyk a zavolat zabudovanou funkci `t()`, která přijímá jako parametr právě klíč. Pokud leží v nějaké podskupině, oddělují se jednotlivé skupiny tečkami.

I18next toho ale umí mnohem více, než jen hledat v překladovém slovníku<sup>[30]</sup>. Dvojitými složenými závorkami můžete do překladů přidávat proměnné a ty za chodu měnit. Framework umí pracovat i s počty a vracet překlady podle toho, zda je vstup jeden, dva, pět, nebo žádný. Problém mu nedělá ani formátování čísel.

Nevýhodou I18next je, že funguje pouze na klientovi, a není tudíž příliš kompatibilní s Next.js. Pro lepší funkčnost jsme začali používat `next-intl`, který se inicializuje na serveru a jehož překlady fungují všude. Výhodou je i velmi podobná syntax, což nám celkovou migraci ulehčilo.

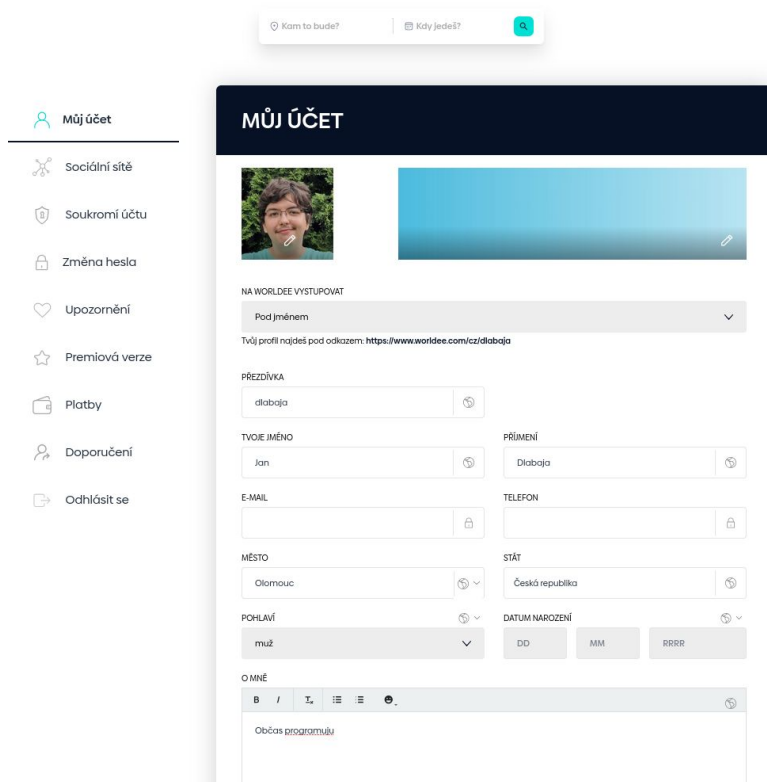
Veškeré překlady se ale stále musí napsat někde do projektu. A zatímco programátoři nejsou moc dobří v překládání, většina překladatelů zase asi nebude schopná si naklonovat repozitář, upravovat JSON (nebo i jiný formát) a pak vše nahrát zpět na server. Proto existují různá řešení na webu, která jsou grafická a vnitřních implementací překladatele odstiňují. A jedno z nich, které ve Worldee používáme, je Localazy.

Tento, jak jsem později zjistil, český produkt, nám umožňuje vytvořit projekt, do něj nahrát soubory pro přeložení a přidat překladatele. Ti mají práci ulehčenou jak grafickým rozhraním, tak i nástroji, jako například zabudovanými překladači, podporou plurálů ve všech světových jazycích, slovníky, nebo řešení duplicit<sup>[29]</sup>. Zároveň se dá integrovat nejen se všemi známými frontendovými frameworky, ale i s Figma, Unity enginem, GitHubem, a dokonce Excelem<sup>[28]</sup>.

### 3 Tvorba stránky s nastavením účtu

Na tomto projektu jsem začal pracovat asi dva týdny po příchodu do firmy. Uměl jsem jen HTML, CSS, základy JavaScriptu a díky dvou týdnům převádění Less stylů na Sass jsem měl určité povědomí o struktuře repozitáře. Stále jsem ale nerozuměl Reactu, Next.js, Storybooku, ani ostatním technologiím, o kterých jsem v předchozí kapitole psal. Proto bylo rozhodnuto, že bude nejlepší se tyto technologie naučit. A volba padla právě na stránku s nastavením účtem.

Bylo to ideální. Šlo o poslední stránku (kromě administrace, která je ale samostatný projekt sama o sobě) napsanou v českém Nette frameworku. Nette renderuje stránky na serveru, což snižuje jejich interaktivitu, nepodporuje SPA a celkově si moc s React ekosystémem nerozumí<sup>[22]</sup>. Navíc se s přechodem na Next.js a Sass styly stránky částečně rozbily, a než ji stále kontrolovat a opravovat, bylo lepší zbavit se technického dluhu a konečně ji přepsat.



**Obrázek 3.1:** Staré nastavení účtu napsané v Nette. Zdroj: vlastní práce.



## 3.1 Vizuální struktura stránky

Celou stránku jsem rozdělil na levé menu a pravý kontejner, do kterého se bude vkládat obsah stránky.

### 3.1.1 Komponenty

Aby se dal projekt co nejlépe škálovat, shlukuje se kód do samostatných komponent. Takto se dá využívat na několika místech v projektu.

#### Div

Aby byly komponenty konzistentní, neměly by se nikdy stylovat zvenku. Můžete ale stylovat elementy, které je obalují. Díky obalení do `<div/>` tak najednou můžete komponenty posouvat, přidávat přes ně vrstvy, a různě s nimi manipulovat.

#### FlexRow a FlexColumn

Tyto dvě jednoduché komponenty jsem v rozvržení používal skoro všude. Mají na sobě flex box a dá se jim nastavovat řádkování, zarovnání obsahu, a kombinace sloupců a řádků umožňuje vytvořit jakkoliv komplexní design.

#### Text

Jedna z našich nejjednodušších komponent je obyčejný text. Dá se na něm měnit velikost, barva, tučnost, podtržení, a několik dalších vlastností. Pro nadpisy používáme Heading, který automaticky velikost textu přizpůsobuje rozlišení obrazovky.

#### Ikona

Na stránkách používáme dva typy ikon – Fluent ikony jsou externí (viz <https://fluenticons.co>) a obsahují knihovnu několika tisíc piktogramů, které se dají použít téměř pro všechny účely. Ostatní ikony a obrázky, které knihovna nemá, si pak tvoříme sami, ve formátu SVG nebo PNG.

## **Tlačítko**

Tlačítek můžete na webu najít osm druhů, každé z nich používá trochu jinou barevnou paletu (zelené, černé, transparentní, ...). Kromě textu podporují i ikony (zleva nebo zprava), velikost, dají se vypnout, a dokonce umí zobrazovat načítací animaci, kterou jsem použil pro odesílání dat formuláře.

Kromě toho máme vytvořené checkboxy nebo toggly, které na stránce s nastavením naleznete také.

## **Link**

Do normálního odkazu musíte psát celou adresu. Do našeho se vkládá objekt `ActionData`, který ji reprezentuje a jeho předdefinované hodnoty zabraňují duplicitním a chybným url. Umí otevírat i soubory nebo stránky v novém panelu a je plně kompatibilní s Next.js - po najetí na něj pošle požadavek serveru na vykreslení další stránky, takže uživatel po kliknutí nemusí na nic čekat.

## **Text box**

Základ této komponenty tvoří obyčejný `<input/>` element. Je ale obohacený o placeholder, který se přesune nahoru, pokud má Text box nějaký obsah. Pokud je jeho obsah nesprávný, je možnost mu nastavit parametr `errorMessage` a tím ho celý nechat zčervenat, včetně malého textu s vysvětlením chyby pod boxem. Také se mu dá měnit typ (například na heslo), zda se do něj dá psát, nebo jeho velikost.

## **Combo box**

Tato komponenta je užitečná pro výběr z předdefinovaných hodnot. Její základ tvoří knihovna `react-select`. Dá se v něm vyhledávat, má podporu i pro mobilní telefony a v projektu ho často používáme například pro výběr zemí, měn, nebo telefonních předvoleb.

## **Text editor**

Někdy potřebujete text s formátováním. A rozdíl mezi normálním `<input/>` elementem a naší Text editor komponentou je srovnatelný s rozdílem mezi poznámkovým blokem

a Wordem. A tím myslím i v komplexnosti, podobný editor jsem si zkoušel napsat sám a není to nic jednoduchého. Samozřejmě je podpora kurzívy, podtržení, i tučného písma, tvorba seznamů, hypertextových odkazů, i vestavěná klávesnice pro emotikony.

## **Toast**

Toast je obdélníková komponenta, která se (většinou dole) zobrazí, aby uživatele informovala o vykonané akci. Například, zda se uložily změny, nebo naopak, že server neodpovídá.

## **Swipeable drawer**

Díky této komponentě od mého oblíbeného Material UI můžeme poskytovat hezké rozhraní i pro mobil. Ačkoliv umí jen „vysunout“ obsah přes část stránky, používáme ji pro mobilní Combo boxy i Menu v nastavení a administraci. Aktivace probíhá kliknutím na tlačítko a zpět se zasune dotykem kamkoliv mimo sebe.

## **Banner with editor**

První ze zmíněných komponent, kterou jsem částečně vytvořil sám. Původně šlo o banner na profilu, ale protože ho nikdo neplánoval znovu použít, veškerá logika pro úpravu fotky byla mimo něj. Vše se mi povedlo sjednotit a nyní se do něj jednoduše vkládají parametry jako fotka, akce při změně nebo smazání fotky, zda má tlačítko pro úpravy a také její výška, která se na obou stránkách liší.

## **User icon with editor**

Ikona s uživatelskou fotkou měla ten samý problém co banner. Veškerá logika pro úpravu fotky byla součástí velké kompozice profilu. Pokud jste majitel profilu (nebo jen nastavíte parametr canEdit na true), můžete si jednoduše změnit vaši profilovou fotku.

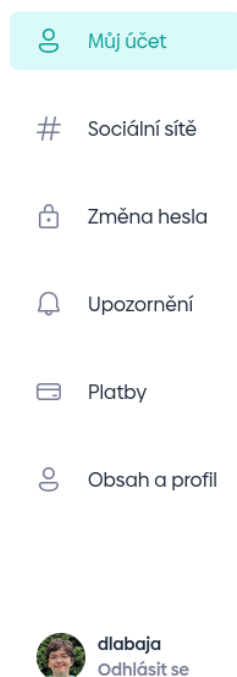
### 3.1.2 Kompozice

#### Menu

Inspirace pro menu vycházela z kompozice, která se už používala na stránce s Travel Buddy administrací. Protože ale nebyla uzpůsobená na to, aby se používala jinde, bylo na mě ji to „naučit“.

Mé nové menu bylo složené ze dvou částí. První, horní, byla samotná navigace. Ta se od té původní zase tolik nelišila, jen jsem podle designu trochu upravil barvy. Jinak se jednalo o několik malých tlačítek, do kterých šel vložit text, ikona, odkaz, akce po kliknutí, a zda je aktivní. Vše ovládala rodičovská komponenta AppMenu (pro desktop) nebo její potomek MenuRight (pro mobil). Protože tlačítek bylo několik pod sebou, ke každému jsem ještě přidal klíč, abych Reactu ulehčil jejich vykreslení. Druhá komponenta byl obrázek uživatele s jeho jménem a textem pro odhlášení. Přes ni byl daný odkaz, po jehož kliknutí byl uživatel odhlášen a vrácen na domovskou stránku.

Celé menu je poměrně modulární. V závislosti na rozlišení je buď viditelné vždy (desktop), nebo po nastavení parametru *open* (mobil). Všechny odkazy se do něj předávají formou odkazu, pro mobil lze nastavit i horní nadpis.



Obrázek 3.2: Menu. Zdroj: vlastní práce.

## **Account Container**

Protože má každá ze stránek stejné menu a horní část, vytvořil jsem pro ně společný kontejner. Ten z aktuální cesty zjišťuje jméno stránky a v mobilní verzi se mu objevuje tlačítko, kterým se dá otevírat postranní menu. Stránky, které vám nyní představím, se do něj vkládají jako potomci.

### **Podstránka Můj účet**

Stránka s účtem se dá označit jako srdce celého nastavení. Na první pohled upoutá pozornost uživatele banner s jeho profilovým pozadím, o kterém již byla řeč. Jediný rozdíl oproti tomu na profilu je jeho výška, která je o 100 pixelů kratší. Hned pod ním se dá nastavit profilový obrázek.

Kromě toho jde na této stránce nastavit jméno, příjmení, přezdívku, pod čím chcete vystupovat, email, telefon, a sekci O mně, ve které se uživatel může představit a napsat něco o sobě.


Ačkoliv komponenta s názvem města Olomouc vypadá jen jako další Text box, není tomu tak. Je totiž připojená na službu Google Mapy a při psaní začne uživateli nabízet názvy měst. Box vlastně ani neobsahuje text, ale objekt, jehož součástí jsou i GPS souřadnice daného místa.

Posledním boxem je již zmiňovaný combo box se všemi státy světa.

Na konci celého nastavení leží tzv. Bookmark komponenta. Když ji rozbalíte, dostanete možnost odstranit váš účet z Worldee.

Po kliknutí na tlačítko se otevře modál – takové malé plovoucí okno. Tento konkrétní lze zavřít jen kliknutím na horní křížek nebo jedním ze dvou spodních tlačítek. Uvnitř jsou checkboxy, ve kterých uživatel vybere, proč chce Worldee opustit. Také může dolů do Text boxu napsat konkrétní důvod. Dole pak už jen potvrdí své heslo (pokud ho má) a kliknutím na tlačítko „Opustit Worldee“ bude odhlášen a přesměrován na hlavní stránku.

## Můj účet



Jméno  
Jan

Příjmení  
Dlabaja

Přezdívka  
dlabaja

Na Wordee vystupovat  
Pod jménem










Tvůj profil najdeš pod odkazem: <https://www.wordee.com/dlabaja>

E-mail

Telefon


Na tvém veřejném profilu se to neobjeví

Na tvém veřejném profilu se to neobjeví

**B** *I* U         

Občas programuju...

Olomouc

 Česká republika

Odstranění účtu

Uložit změny

**Obrázek 3.3:** Podstránka Můj účet. Zdroj: vlastní práce.



## Podstránka Sociální sítě

Na této stránce si uživatel s Worldee může propojit svůj Facebook nebo Instagram účet, který se mu pak objeví na profilu. Každá sociální síť se skládá z názvu, levé části a tlačítka. Pokud jste si ji zatím nepřipojili, bude levá část pouze Text box a pravá tlačítko „Propojit účet“. Pro propojení stačí do kolonky napsat odkaz na profil. Pokud je url validní, Text box se změní na komponentu složenou z ikony a dvou textů oznamujících, že je účet propojen, a z původně zeleného tlačítka se stane tmavě šedé s názvem „Zrušit propojení“.

### Sociální sítě

#### Facebook

✓ Propojeno

Odkaz se zobrazí veřejně na tvém profilu

Zrušit propojení

#### Instagram

Odkaz na tvůj účet

Propojit účet

**Obrázek 3.5:** Podstránka Sociální sítě. Zdroj: vlastní práce.



## Podstránka Změna hesla

Zatím jedna z těch jednodušších stránek – pouhá tři pole, jejichž typ je nastaven jako heslo, a text. Pokud jste přihlášení přes službu třetí strany jako třeba Google, první pole se vám neukáže – ještě u nás totiž žádné heslo nemáte. Objeví se, až si nastavíte nové.

### Změna hesla

Původní heslo

Nové heslo

Potvrzení hesla

Minimum je 8 znaků

Uložit heslo

**Obrázek 3.6:** Podstránka Změna hesla. Zdroj: vlastní práce.

## Podstránka Upozornění

Tato kompozice se skládá ze dvou částí. Ta levá je velmi podobná té ze Sociálních sítí – jen má jiný obsah. Napravo je ale toggle tlačítko, kterým si uživatel může nastavit, z čeho chce (nebo nechce) dostávat webová upozornění.

### Upozornění

 <b>Sledování</b> Dostávat upozornění pokud tě někdo začne sledovat.	<input checked="" type="checkbox"/>
 <b>To se mi líbí</b> Dostávat upozornění pokud někdo označí tvoji cestu jako „To se mi líbí“.	<input type="checkbox"/>
 <b>To se mi líbí</b> Dostávat upozornění pokud někdo označí tvoji fotku jako „To se mi líbí“.	<input type="checkbox"/>
 <b>Facebook</b> Dostávat upozornění pokud se na Worldee připojí někdo z tvých přátel.	<input checked="" type="checkbox"/>

[Uložit změny](#)

**Obrázek 3.7:** Podstránka Upozornění. Zdroj: vlastní práce.

## Podstránka Obsah a profil

A nyní poslední mnou vytvořená stránka. Je tvořená sekcí Profil a Zobrazení obsahu. V té první si můžete již známými Combo boxy vybrat, zda chcete, aby byl váš profil viditelný pro všechny nebo jen pro vás. Hned vedle zase asijsí zákazníci mohou přepnout střed mapy z Evropy na Asii.

Pod komponentami se nacházejí dva toggly. U horního si nastavíte, zda chcete schvalovat, když vás někdo označí na jakékoliv fotografii, než se toto označení zobrazí na vašem profilu. Druhý toggle už se nachází v sekci Zobrazení obsahu a umožňuje vám nejnovější cesty na Worldee zobrazovat ve feedu.

## Obsah a profil

### Profil

Typ profilu

Soukromý

▼

Varianata světové mapy

Evropa jako střed

▼

Svůj profil uvidíš jen ty.

#### Označení

Chceš schválit cesty, na kterých tě někdo označí, než se zobrazí na tvém profilu?



### Zobrazení obsahu

#### Nejnovější cesty

Chceš zobrazit nejnovější cesty na Worldee ve svém feedu?



Uložit změny

**Obrázek 3.8:** Podstránka Obsah a profil. Zdroj: vlastní práce.

## Podstránka Platby











Poslední stránkou na seznamu je ta s platbami. Jako jedinou jsem ji nemusel psát úplně od základů, nicméně pár úprav bylo pro zakomponování do systému potřeba. Odstranil jsem externí stylování a přesunul ho do nadřazených elementů, načítání dat jsem připojil ke svému systému načítání stránky a proběhlo tak až po prvním renderu komponenty, a zmizela i přehnaně komplexní logika pro zobrazení plateb.

### Platby

Všechny platby

↑↓

Listopad 2024

	SG Auto PRG test 22. 11. 2024	7 210,00 Kč	Podrobt
	Cesta kolem Olomouce 21. 11. 2024	45 860,00 Kč	Podrobt
	SG Auto PRG test 20. 11. 2024	7 200,00 Kč	Podrobt
	SG Auto PRG test 19. 11. 2024	13 310,00 Kč	Podrobt
	Cesta kolem Olomouce 18. 11. 2024	6 842,00 €	Podrobt
	Cesta kolem Olomouce 15. 11. 2024	35 590,00 Kč	Podrobt
	Cesta kolem Olomouce 15. 11. 2024	75 130,00 Kč	Podrobt
	Cesta kolem Olomouce 15. 11. 2024	35 590,00 Kč	Podrobt
	Cesta kolem Olomouce 14. 11. 2024	55 023,00 Kč	Podrobt
	Cesta kolem Olomouce 14. 11. 2024	8 720,00 Kč	Podrobt

1 2 ... 4 >

**Obrázek 3.9:** Podstránka Platby. Zdroj: vlastní práce.

## 3.2 Datová struktura stránky

Kdyby byl celý web tvořen jen vizuálními prvky, moc by toho nedělal. O vizuálním stavu stránky se špatně píše, protože většinou stačí pro dosažení cílového vzhledu jen naskládat několik komponent na sebe. To zajímavé, co sice normální uživatel nevidí, ale bez čeho by stránka nefungovala, je její datová část. Díky ní můžete reprezentovat, jaká mají prvky data, jak jsou propojené, a jak spolu celá architektura funguje.

### 3.2.1 MobX store a validace dat

Jak funguje MobX jsem popisoval ve druhé části této práce. Používáme ho hlavně z toho důvodu, abychom měli data izolovaná od Reactu, ale zároveň je v něm mohli jednoduše používat. Můžeme je tak ukládat do tříd a ty následně bez problémů testovat.

Pro odlišení od komponent má každý takový soubor příponu `*.store.*`. Třída v něm je složená z polí obsahujících všechna data, která se budou na stránce používat, jejich náležitých getterů (a většinou i setterů) a konstruktoru. Aby na změny dat reagoval React, musí být všechna pole označena dekorátorem `@observable`.

Konstruktory by měl správně jen dosadit základní hodnoty, spustit nějaké interní funkce a zavolat `makeObservable()` funkci, čímž aktivuje MobX. Pokud chci získat nějaká externí data, musím to udělat zvlášť. Proto má každá ze stránek svoji veřejně přístupnou metodu `init()`, kterou v Reactu volám po prvním renderu a narozdíl od konstruktoru může být asynchronní. Tím ovšem vzniká další problém – když se stránka načte, chybí jí data. Z toho důvodu jsem si vytvořil vlastní funkci `loadDataAsync()`, kterou všechny stránky používají, spolu s `isInitialized` stavem. Ten se změní, jakmile se všechna data načtou, a díky tomu vím, kdy stránku zobrazit uživateli, a kdy ji nahradit načítací animací.

Každá z mých tříd má dvě speciální pole. `FormChanged` se nastaví na `true` po tom, co změníte jakákoliv data na stránce – třeba vlastní přezdívku. Tím se vám zároveň odemkne tlačítko s odesláním dat, čímž jsem alespoň částečně zabránil tomu, aby náš server zaplňovaly prázdné požadavky. Jakmile data odešlete, `formChanged` se opět nastaví na `false` a tlačítko zamkne.

`FormLoading` se zase nastaví na `true` po tom, co data uložíte. Zamkne všechna pole, aby hodnoty zůstaly během odesílání konzistentní, a zůstane tak, dokud server nepošle odpověď.

Metoda `init()` není jediná, která potřebuje přistupovat k nějaké externí API. Pro odeslání dat na server má každá ze tříd i metodu `save()`, některé z nich pak mají metody například na propojení Facebooku nebo Instagramu, nebo mazání a nahrávání fotky na profilu.

U nastavení účtu a změny hesla jsem se nevyhnul ani validaci dat. Pokaždé, když kliknete na tlačítko pro uložení, spustí se metoda `validate()`. Ta aktualizuje hodnoty jako `nicknameIsValid`, nebo `emailIsValid` podle toho, zda nejsou prázdné nebo v neplatném formátu. Getterem `isValid()` se dá následně zjistit, zda validace celého formuláře prošla.

Pokud bylo vráceno `true`, spustí se část, kterou mají všechny stránky stejnou – funkce `saveDataAsync()`. Té předám `save()` metodu dané stránky, a také již zmíněný `formLoading` a `formChanged`, se kterými podle odpovědi serveru manipuluje. Ať už je odpověď jakákoliv, vždy se dole zobrazí `Toast`, díky němuž uživatel zjistí, zda se operace zdařila, server neodpovídá, nebo se například neshodují hesla.

### 3.2.2 Data, konvertory a API

O úroveň výše jsou již samotné datové struktury. Skládají se z konstruktoru, který dostane hodnoty a přiřadí je do getterů. Pro co nejnižší komplexitu by zde neměl být používán MobX a data by měla být pouze pro čtení.

Zhruba na stejné úrovni jako data jsou i konvertory. Ne vždy jsou totiž data ze serveru v pro nás ideálním formátu. Například čas i enumy jsou vyjádřené jako string, s čímž se ale špatně pracuje a museli bychom je před každým použitím převést. Stejně je to i s různými objekty. Konverze probíhá ještě před tím, než se vše vloží do datové třídy, takže na stránce dostávám takovou strukturu dat, se kterou už můžu jednoduše pracovat. Přenos zpátky probíhá podobným způsobem, jen je třeba data zase převést na serverem podporovaný formát.

Ještě výš se nachází námi zpracované metody pro práci s API. Jsou systematicky rozdělené do několika tříd podle místa použití, abych, pokud potřebuji pracovat s nastavením, nemusel importovat něco, co má navíc funkce pro získání nejnovějších cest, přihlášení uživatele, a vrácení posledních hodnocení na Googlu. Každá metoda přijímá kromě dat také parametr na úpravu požadavku, díky kterému můžu měnit jeho hlavičku, cookie, nebo zda chci, aby při chybě stránka nespadla, a já si tak mohl vyjímku odchytil sám.

Každá z metod nakonec zavolá funkci importovanou z `api.ts` souboru. Tento soubor o délce přes čtyřicet tisíc řádků se automaticky generuje pomocí nástroje Swagger, a skrývá reálnou API implementaci, která je vlastně jen soubor všech možných HTTP požadavků, které fungují jako most mezi frontendem a backendem. Výš už pro mě cesta nevede.

Ve Storybooku je struktura trochu odlišná. API funkce se volají stále stejně, ale při načtení jakékoliv story se jejich původní implementace přepíše na mock. Místo volání na server tak dostanu přímo naši datovou třídu, naplněnou falešnými hodnotami. Abychom alespoň částečně simulovali internetovou odezvu, každá z metod vrací data až po uplynutí doby 300 milisekund.

### 3.2.3 Unit testy

Abychom si byli jistí, že web funguje správně alespoň po datové stránce, máme pomocí Jestu (a dříve Karmy) napsaných několik stovek testů. Všechny jsou ve složce test, která má úplně stejnou strukturu jako složka s komponentami, jen s tím rozdílem, že všechny soubory mají příponu *\*.spec.\**

Každá datová část se testuje trochu jinak. Aktuálně nejvýše v hierarchii jsou testy na datové třídy. Protože jsou neměnné a dají se naplnit jen jednou, je potřeba jen vyzkoušet, zda se data dosadila správně. To lze zjistit pomocí tvorby falešných dat, jejich dosazením a následně porovnáním všech veřejných getterů.

Testům se samozřejmě nevyhnu ani konvertory. Zde konkrétně byl potřeba jen jeden, a to pro první stránku s uživatelským nastavením. Testy jsou z principu podobné těm datovým, opět jen stačí vytvořit falešná data, ty dát do konvertoru, a pak zkontrolovat data nového objektu, který od něj dostaneme.

Poslední a největší skupina testů se zaměřuje na samotné komponenty a jejich MobX store. Zároveň jsou také nejkomplexnější, protože musí pokrýt vše od změny a získávání dat, validaci a komunikaci s API.

Stejně jako u ostatních typů se i tady kontroluje dosazení do konstruktoru a následný stav. Kromě toho se ale proměnné zkouší nastavit již mimo konstruktor a zkoumají, zda správně funguje setter. Na všech stránkách se ve většině setterů nastavoval i `formChanged` parametr, který byl také potřeba do testů zahrnout.

Validace už je docela komplikovaná. Například u přezdívky se jen stačilo ujistit, zda není prázdná. Email už zase musel mít správnou strukturu. A combo box, který určoval, zda budete viditelní pod přezdívkou nebo pod jménem, si musel být jistý, že při výběru „Pod jménem“ byla správně nastavená alespoň jedna část jména a „Pod přezdívkou“ zase požadoval nastavenou přezdívku, která se validovala v předchozím kroku.

Následně se testují všechny metody integrující API volání. Ačkoliv jsou všechny z nich jen mocky a ve skutečnosti na server nevolají, můžu vyzkoušet, zda se pod nimi zabalená API metoda zavolala s těmi parametry, které jsem jejímu rodiči předal.

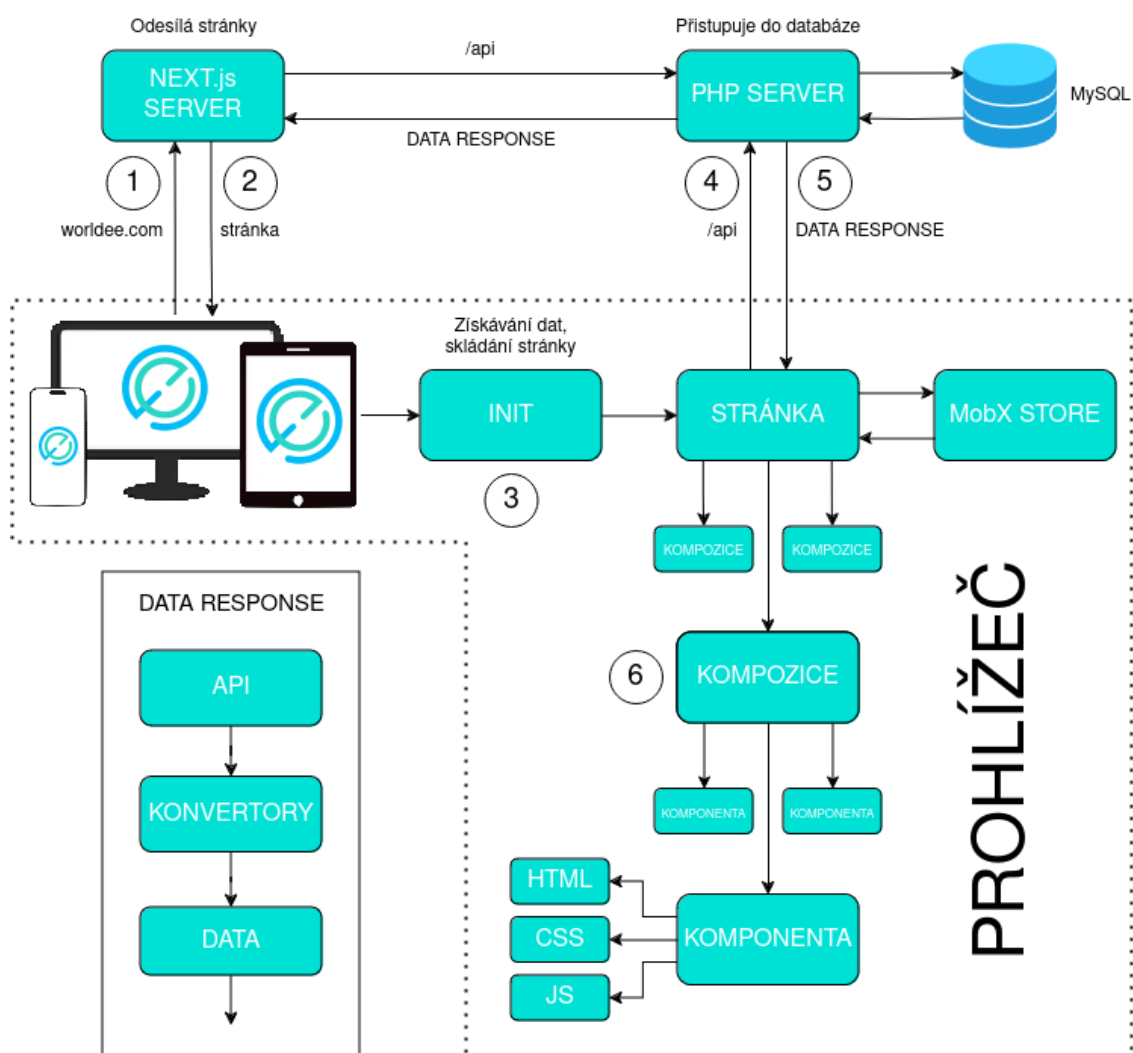


### 3.2.4 Architektura webu

Pro lepší představu o funkci webu vám nyní vysvětlím, co všechno se stane, pokud stránku navštívíte.

Vše začne zadáním adresy worldee.com do adresního řádku (1). Požadavek se odešle na náš Next.js server, kde si ho jako první převezme middleware. Ten zkontroluje a případně upraví cookies a hlavičky požadavku a do url přidá lokalizaci. Potom ho předá routeru, který podle url najde většinou již předkreslenou požadovanou stránku, přeloží ji a posílá zpátky do prohlížeče (2).

Pokud byla stránka předem předkreslená, uživatel už nyní může vidět všechny její prvky, ač jde zatím jen o vizuál postrádající jakoukoliv funkčnost. Pokud ne, stránka je prázdná. Prohlížeč musí spustit React (3), stránku inicializovat a podle odpovědi serveru vytvořit všechny kompozice a komponenty. Po prvním renderu začíná probíhat komunikace s naším PHP serverem (4) a získaná data jsou z JSONu převáděna na použitelný formát (5), kterým jsou plněny datové třídy. Stránka je nyní plně načtená, celý proces trval jednotky sekund.



**Obrázek 3.10:** Diagram architektury webu. Zdroj: vlastní práce.

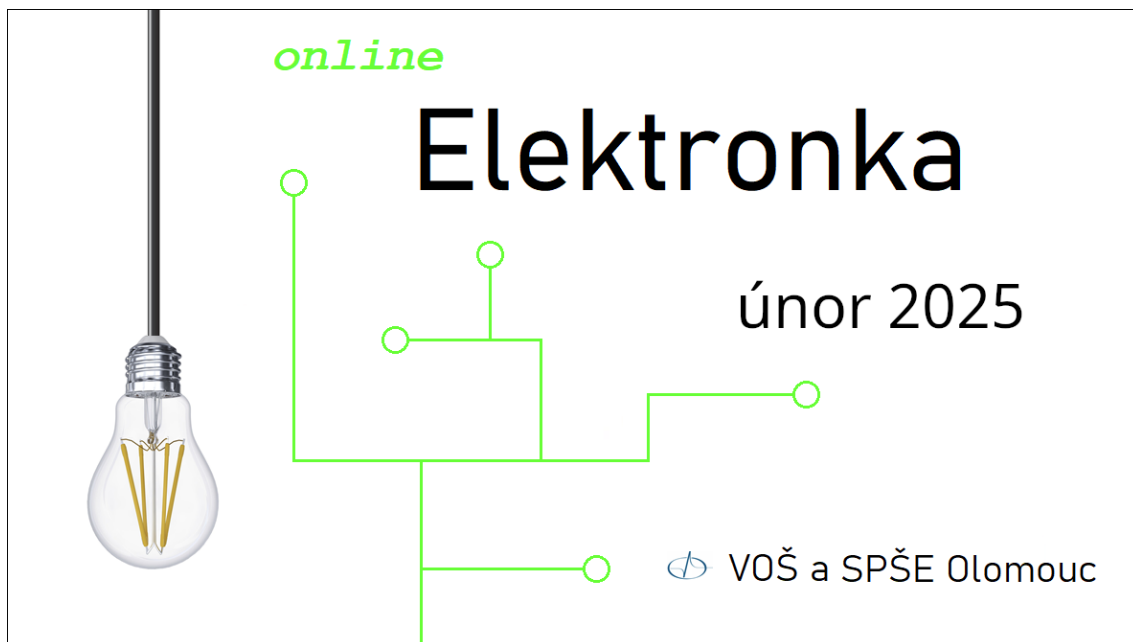
### 3.3 Finální vzhled stránky

The screenshot displays the 'Můj účet' (My Account) page. At the top, there's a navigation bar with links: 'Cesty', 'Průvodce destinacemi', 'O nás', and 'Travel Buddies'. A search bar is also present. The user is logged in as 'diabaja'. The left sidebar contains a menu with 'Můj účet' (highlighted), 'Sociální sítě', 'Změna hesla', 'Upozornění', 'Platby', and 'Obsah a profil'. The main content area is titled 'Můj účet' and features a large blue header image. Below this is a profile picture of a person. The form includes fields for 'Jméno' (First Name: diabaja), 'Příjmení' (Surname), 'Přezdívká' (Nickname: diabaja), 'E-mail' (diabajahonzo@gmail.com), and 'Telefon'. There are also dropdown menus for 'Na Worldde vystupovat' (set to 'Pod jménem') and 'Město' (set to 'Česká republika'). A bio section with a rich text editor is labeled 'O mně'. At the bottom, there's a section for 'Odstranění účtu' and a 'Uložit změny' (Save changes) button.

**Obrázek 3.11:** Vizuální podoba stránky, dostupná po přihlášení na [www.worlddee.com/settings/account](https://www.worlddee.com/settings/account). Zdroj: vlastní práce.

## 4 Články v časopisu Elektronka

Ačkoliv se nejedná o podmínku hodnocení této práce, pro lepší pochopení této problematiky jsem začal psát články pro školní časopis Elektronka<sup>[37]</sup> a obohacoval jimi tak svou rubriku IT doupe<sup>[19]</sup>. První vznikl v září 2024 a mým úkolem je popsat od základů vytvořit web s tím, že každý měsíc do něj implementuju jinou technologii. Od primitivní HTML stránky se tak postupně dostávám k bundlingu, Reactu, Next.js, nebo lokalizacím.



**Obrázek 4.1:** Titulní obrázek časopisu Elektronka (únor 2025).<sup>[37]</sup>

# Závěr

Jak jste asi zjistili, svět webu je komplikovanější, než se může na první (nebo i druhý) pohled zdát. Už dávno mezi sebou neposíláme statické HTML dokumenty a s příchodem stylů, JavaScriptu, JQuery, a prvních frontendových frameworků se toho hodně změnilo. Nové, někdy opravdu revoluční knihovny vycházejí téměř na denní bázi a je stále těžší držet krok s novými technologiemi. Dokonce i popsané technologie jsou jen špička ledovce a je dost možné, že za pár let už některé z nich ani nebudou existovat.

A ačkoliv jsou nové technologie komplikované, díky jejich abstrakci je psaní a hlavně následné škálování webu mnohem jednodušší než dřív. Místo šablon používáme komponenty a kompozice, místo stylů CSS moduly a Tailwind, JavaScript je typově bezpečný, zkompileovaný kód je díky bundlerům malý a efektivní, před chybami nás chrání testy, lokalizaci můžou překladatelé dělat přes webové rozhraní, a spoustu dalších větších a menších vylepšení pro uživatele i vývojáře.

Po dopsání tohoto projektu jsem se účastnil vývoje i jiných částí webu. Opravil a vyčistil jsem Text editor, vytvořil komponentu s telefonním číslem a výběrem předvolby, přidal vyhledávání do mobilního combo boxu, mapuju data ze scraperu na naše API, tvořím a upravuju některé ze stránek a poslední dobou pracuji na serverovém renderování, které bude mít reálný dopad na rychlost webu a udělá web atraktivnější pro vyhledávače, čímž přilákáme více potenciálních zákazníků.

Tvorba webů je něco, v čem jsem se momentálně našel, a v čem bych se do budoucna rád nadále zlepšoval. Na závěr bych chtěl proto poděkovat celému vývojovému týmu, že mi pomáhají na cestě za jeho pochopením.

# Seznam použitých zdrojů

- [1] AliaWeb, spol. s r.o. *We travel the world s.r.o.* 2024. URL: <https://www.podnikani.cz/08351864/we-travel-the-world-sro> (cit. 28.09.2024).
- [2] Avi Aryan. *Introduction to Functional Programming: JavaScript Paradigms.* 2025. URL: <https://www.toptal.com/javascript/functional-programming-javascript> (cit. 22.02.2025).
- [3] aryash. *History of JavaScript.* 2024. URL: <https://www.geeksforgeeks.org/history-of-javascript> (cit. 06.10.2024).
- [4] David Bolton. *What Is an Enum in Programming Languages?* 2024. URL: <https://www.thoughtco.com/what-is-an-enum-958326> (cit. 14.08.2024).
- [5] MDN contributors. *Functions.* 2024. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions> (cit. 07.10.2024).
- [6] MDN contributors. *JavaScript basics.* 2024. URL: [https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/JavaScript\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics) (cit. 06.10.2024).
- [7] MDN contributors. *JavaScript data types and data structures.* 2024. URL: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data\\_structures](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures) (cit. 06.10.2024).
- [8] MobX contributors. *Deriving information with computed.* 2024. URL: <https://mobx.js.org/computed.html> (cit. 18.08.2024).
- [9] Storybook contributors. *Mocking modules.* 2024. URL: <https://storybook.js.org/docs/writing-stories/mocking-data-and-modules/mocking-modules> (cit. 18.08.2024).
- [10] TypeScript contributors. *Generics.* 2024. URL: <https://www.typescriptlang.org/docs/handbook/2/generics.html> (cit. 09.10.2024).
- [11] Webpack Contributors. *Concepts.* 2024. URL: <https://webpack.js.org/concepts/> (cit. 18.08.2024).

- [12] Webpack Contributors. *Dependency Graph*. 2024. URL: <https://webpack.js.org/concepts/dependency-graph> (cit. 18.08.2024).
- [13] Wikipedia contributors. *Less (style sheet language)*. 2024. URL: [https://en.wikipedia.org/wiki/Less\\_\(style\\_sheet\\_language\)](https://en.wikipedia.org/wiki/Less_(style_sheet_language)) (cit. 14.08.2024).
- [14] Wikipedia contributors. *Node.js*. 2024. URL: <https://en.wikipedia.org/wiki/Node.js> (cit. 06.10.2024).
- [15] Wikipedia contributors. *Sass (style sheet language)*. 2024. URL: [https://en.wikipedia.org/wiki/Sass\\_\(style\\_sheet\\_language\)](https://en.wikipedia.org/wiki/Sass_(style_sheet_language)) (cit. 14.08.2024).
- [16] Wikipedia contributors. *Webpack*. 2024. URL: <https://en.wikipedia.org/wiki/Webpack> (cit. 18.08.2024).
- [17] Chris Coyier. *CSS Flexbox Layout Guide*. 2024. URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox> (cit. 14.09.2024).
- [18] css-modules. *CSS Modules*. 2024. URL: <https://github.com/css-modules/css-modules> (cit. 14.08.2024).
- [19] Jan Dlabaja. *IT doupe*. 2025. URL: [https://github.com/dlabaja/IT\\_doupe](https://github.com/dlabaja/IT_doupe) (cit. 26.02.2025).
- [20] Fireship. *Module Bundlers Explained... Webpack, Rollup, Parcel, and Snowpack*. 2020. URL: <https://www.youtube.com/watch?v=5IG4UmULyoA> (cit. 02.11.2024).
- [21] FrontStart. *How React ACTUALLY works (DEEP DIVE 2023)*. 2024. URL: <https://www.youtube.com/watch?v=za2FZ8QCE18> (cit. 02.11.2024).
- [22] David Grudl. *How Do Applications Work?* 2024. URL: <https://doc.nette.org/en/application/how-it-works> (cit. 08.09.2024).
- [23] L. Heider. *Jest coverage: How can I get a total percentage of coverage?* 2020. URL: <https://stackoverflow.com/questions/60967561/jest-coverage-how-can-i-get-a-total-percentage-of-coverage> (cit. 04.09.2024).
- [24] Tailwind Labs Inc. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. 2025. URL: <https://tailwindcss.com/> (cit. 26.02.2025).

- [25] Ilya Kantor. *An Introduction to JavaScript*. 2024. URL: <https://javascript.info/intro> (cit. 11.08.2024).
- [26] Kovy. *Ultimátní roadtrip Skotskem! w/Míra | KOVY*. 2024. URL: <https://www.youtube.com/watch?v=66TMsTKaIbE> (cit. 25.11.2024).
- [27] Alex Kugell. *7 Top React State Management Libraries*. 2022. URL: <https://trio.dev/7-top-react-state-management-libraries> (cit. 18.08.2024).
- [28] Localazy. *Integrate Localazy*. 2024. URL: <https://localazy.com/integrations> (cit. 07.09.2024).
- [29] Localazy. *Localization for teams of any size*. 2024. URL: <https://localazy.com/pricing> (cit. 07.09.2024).
- [30] locize. *i18next Crash Course | the JavaScript i18n framework*. 2022. URL: [https://www.youtube.com/watch?v=SA\\_9i4TtxLQ](https://www.youtube.com/watch?v=SA_9i4TtxLQ) (cit. 04.09.2024).
- [31] Filip Magalhães. *Sto milionů pro Worldee. Český cestovatelský startup nakopne nový investor, je jím Smička z Bazoše*. 2024. URL: <https://www.hrot24.cz/clanek/sto-milionu-pro-worldee-cestovatelsky-startup-nakopne-novy-investor-je-jim-smicka-z-bazose> (cit. 28.09.2024).
- [32] nextjs.org. *Introduction*. 2024. URL: <https://nextjs.org/docs> (cit. 21.08.2024).
- [33] nextjs.org. *Learn Next.js*. 2024. URL: <https://nextjs.org/learn> (cit. 25.08.2024).
- [34] nextjs.org. *Next.js Client Components*. 2024. URL: <https://nextjs.org/docs/app/building-your-application/rendering/client-components> (cit. 01.09.2024).
- [35] nextjs.org. *Next.js Server and client components*. 2024. URL: <https://nextjs.org/learn/react-foundations/server-and-client-components> (cit. 01.09.2024).
- [36] nextjs.org. *Next.js Server Components*. 2024. URL: <https://nextjs.org/docs/app/building-your-application/rendering/server-components> (cit. 01.09.2024).



- [37] Studenti SPŠE Olomouc. *Elektronka Online*. 2025. URL: <https://www.spseol.cz/prace-zaku-a-studentu/casopis-elektronka> (cit. 26.02.2025).
- [38] React team and external contributors. *State: A Component's Memory – React*. 2024. URL: <https://react.dev/learn/state-a-components-memory> (cit. 14.08.2024).
- [39] React team and external contributors. *useRef – React*. 2024. URL: <https://react.dev/reference/react/useRef> (cit. 14.08.2024).
- [40] Rajae Robinson. *Understanding Hydration in Next.js*. 2024. URL: <https://dev.to/rajaerobinson/understanding-hydration-in-nextjs-b5m> (cit. 26.02.2025).
- [41] We travel the world s.r.o. *Vyber si svůj způsob cestování*. 2024. URL: <https://www.worldee.com> (cit. 25.11.2024).
- [42] Sanity. *JSX definition*. 2024. URL: <https://www.sanity.io/glossary/jsx> (cit. 04.10.2024).
- [43] Benjamin Semah. *What Exactly is Node.js?* 2022. URL: <https://www.freecodecamp.org/news/what-is-node-js> (cit. 21.08.2024).
- [44] Pradhumn Sharma. *What is the difference between jest.fn() and jest.spyOn() methods in jest?* 2019. URL: <https://stackoverflow.com/questions/57643808/what-is-the-difference-between-jest-fn-and-jest-spyon-methods-in-jest> (cit. 04.09.2024).
- [45] Robert Sheldon. *What is internationalization?* 2024. URL: <https://www.techtarget.com/whatis/definition/internationalization-I18N> (cit. 25.11.2024).
- [46] Web Tech Talks. *Webpack founder Tobias Koppers demos bundling live by hand*. 2018. URL: <https://www.youtube.com/watch?v=UNMkLHzofQI> (cit. 18.08.2024).
- [47] i18next team. *Introduction*. 2024. URL: <https://www.i18next.com> (cit. 04.09.2024).

- [48] PurelyFunctional tv. *React and the Virtual DOM*. 2015. URL: <https://www.youtube.com/watch?v=BYbgopx44vo> (cit. 02.11.2024).
- [49] web.dev. *Types of automated testing*. 2024. URL: <https://web.dev/learn/testing/get-started/test-types> (cit. 04.09.2024).
- [50] ByteScout Team of Writers. *What is API and why it exists?* 2024. URL: <https://bytescout.com/blog/2019/03/what-is-api-and-why-it-exists.html> (cit. 21.08.2024).
- [51] Sullivan Young. *What really is JavaScript? (Origins and The Basics)*. 2022. URL: <https://medium.com/@sullivanyoung/what-really-is-javascript-origins-and-the-basics-1b4914f355ed> (cit. 28.09.2024).
- [52] Tomáš Zapletal. *Toto je Worldee*. Unpublished. 2024.

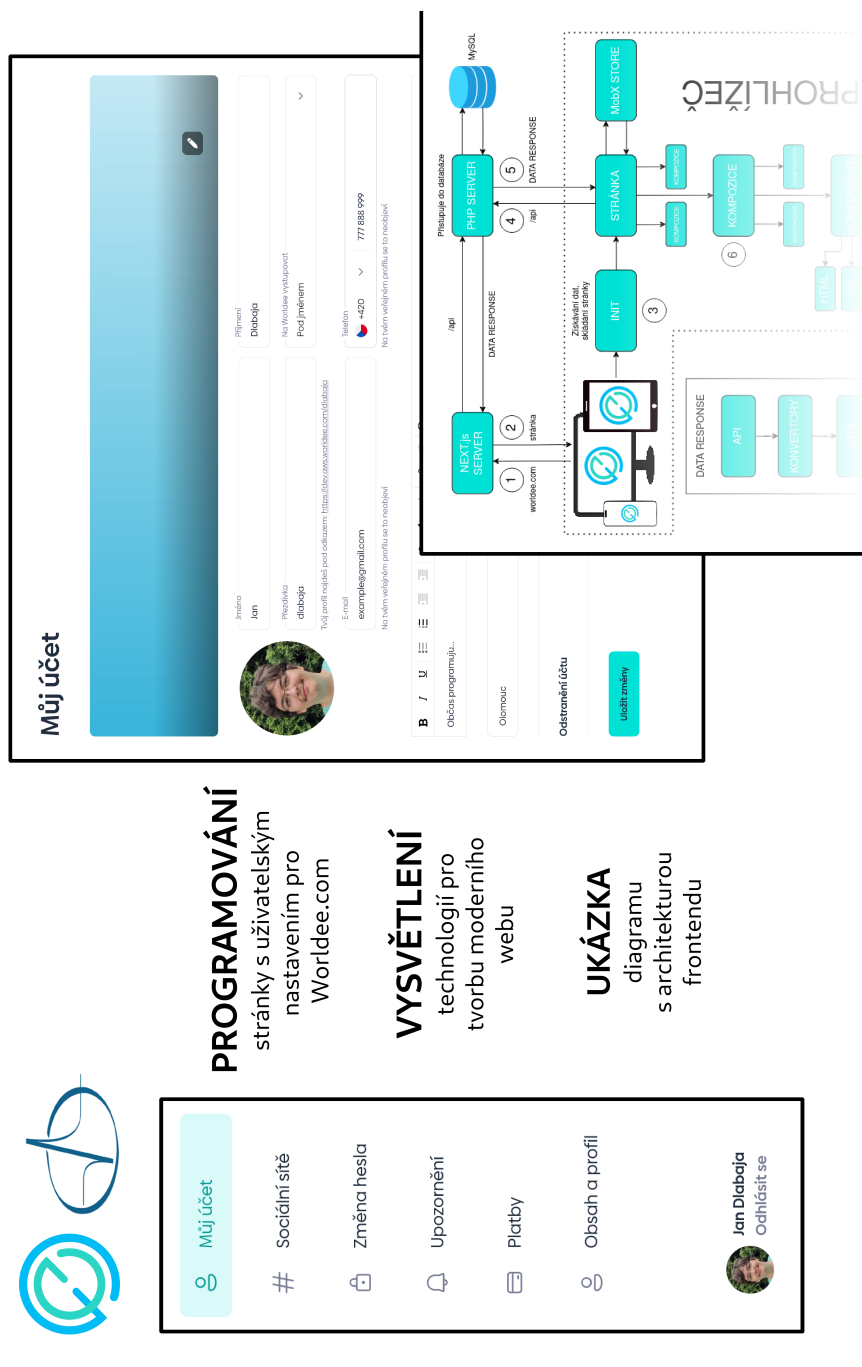
# Seznam obrázků

1.1	Logo firmy Worldee . . . . .	7
3.1	Staré nastavení účtu napsané v Nette. Zdroj: vlastní práce. . . . .	32
3.2	Menu. Zdroj: vlastní práce. . . . .	36
3.3	Podstránka Můj účet. Zdroj: vlastní práce. . . . .	38
3.4	Dialog s odstraněním účtu. Zdroj: vlastní práce. . . . .	39
3.5	Podstránka Sociální sítě. Zdroj: vlastní práce. . . . .	40
3.6	Podstránka Změna hesla. Zdroj: vlastní práce. . . . .	41
3.7	Podstránka Upozornění. Zdroj: vlastní práce. . . . .	42
3.8	Podstránka Obsah a profil. Zdroj: vlastní práce. . . . .	43
3.9	Podstránka Platby. Zdroj: vlastní práce. . . . .	44
3.10	Diagram architektury webu. Zdroj: vlastní práce. . . . .	50
3.11	Vizuální podoba stránky. Zdroj: vlastní práce. . . . .	51
4.1	Titulní obrázek časopisu Elektronka (únor 2025). <sup>[37]</sup> . . . . .	52

## Seznam tabulek

2.1	Seznam základních elementů v HTML. . . . .	13
2.2	Příklady CSS selektorů a jejich funkcí. . . . .	15

# 1 Plakát



**Vypracoval Jan Dlabaja pod vedením Bc. Martina Meravého**  
v rámci své dlouhodobé maturitní práce pro školní rok 2024/2025

