



**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)

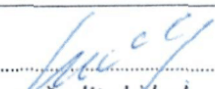
Vedoucí práce: Bc. Martin Meravý

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 PRŮMYSL OVÁ 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 a paní Tereze Vlachové za poskytnutí jazykové korektury.

.....

Jan Dlabaja

Obsah

Úvod	6
1 Firma Worldee	7
1.1 O firmě	7
1.2 Historie	8
1.3 Produkty	10
1.4 Business model a škálovatelnost	11
1.5 Konkurence	12
2 Použité webové technologie	13
2.1 HTML	13
2.2 CSS	15
2.3 JavaScript	16
2.4 TypeScript	18
2.5 Sass, Less a moduly	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
3 Tvorba stránky s nastavením účtu	32

3.1	Vizuální struktura stránky	33
3.1.1	Komponenty	33
3.1.2	Kompozice	37
3.2	Datová struktura stránky	47
3.2.1	MobX store a validace dat	47
3.2.2	Data, konvertory a API	49
3.2.3	Unit testy	50
3.3	Finální struktura stránky	51
	Závěr	53
	Seznam použitých zdrojů	54
	Seznam obrázků	59
	Seznam tabulek	60
	Přílohy	61
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 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 React, Next.js, TypeScript, 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.

Na začátku 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 té nejzákladnější jako HTML a CSS, přes ty pokročilejší jako React, TypeScript, Next.js, až po některé nástroje používané na denní bázi, jedním z nich je třeba právě Storybook nebo 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 frontendu.

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.^[1]

„Členové týmu Worldee vytváří platformu, která umožňuje lidem jednoduše objevovat svět. Spojením autenticity a jednoduchosti může díky naší platformě kdokoli 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^[2]

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 ukázalo 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^[3] a během toho přišla další, tentokrát stomilionová investice.^[4] 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.)“.^[5]

1.3 Produkty

Pokud se s Worldee rozhodnete vydat na cestu, můžete si vybrat ze tří variant.^[2]

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 kdokoli, 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

asi je mrtvý lol 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 nejjednodušších konceptů až k těm náročnějším a méně známým.

2.1 HTML

HTML je značkovací jazyk a slouží pouze k definování struktury stránky. Skládá se z párových (`potomek`) a nepárových (``) značek, které poznáte 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 věci, které na stránce nejdou vidět. Třeba 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 prvek má několik atributů, které můžete nastavit. Id je unikátní a class zase reprezentuje skupiny prvků. Ještě se k nim dostanu. Pomocí href můžete nastavit url adresu (užitečné pro odkazy) a eventy jako onclick zase spustí skript, pokud se naplní nějaká podmínka (v tomto případě kliknutí na tlačítko).

Samo o sobě se dnes HTML defacto nepoužívá. Má pouze základní font, bílé pozadí, a spíše než webovou stránku připomíná Word dokument. Je třeba ho nějak vylepšit. A přesně to je úkol stylů.

2.2 CSS

Styly Vám na stránce umožní upravit skoro všechno – měnit barvy, odsazení, pozice, velikost, Stačí jeden takový soubor připojit do hlavičky stránky. V něm definujete, co konkrétně chcete změnit. Zde uvedu příklad na tlačítku.

Selektor	Popis
<code>button {}</code>	Změní všechny <code><button/></code> elementy na stránce
<code>.save-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 změny. Například `background: red;` nastaví pozadí prvku na červenou, `margin-right: 10px;` ho zase odsadí zprava o 10 pixelů apod. Styly můžete podobně jako u HTML vkládat do sebe – můžete si takto třeba pojistit, aby se změnil jen element s konkrétním rodičem. 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.

Důležité je také chápat rámečky a jak jdou po sobě. Jsou tři – vnitřní okraj (`padding`) určuje místo mezi obsahem a okrajem elementu (`border`), a vnější okraj (`margin`) zase oblast mezi borderem a ostatními prvky. U všech se dá nastavit velikost shora, zdola, zprava a zleva.

Ještě zmíním `display: flex;`. Ten velmi zjednodušuje pozicování položek. Můžete je mít v řádku nebo sloupci, zarovnávat na začátek, střed, nebo konec, nastavovat mezi prvky mezery, a zároveň je i jednoduše roztahovat pomocí `flex-grow: 1;`.^[6]

Ačkoliv už stránka vypadá mnohem lépe, zatím nic nedělá. Tlačítka Vás maximálně přesměrují na jinou stránku a pokud chcete vytvořit něco komplexnějšího než blog, je třeba pochopit ještě třetí důležitou technologii.

2.3 JavaScript

Nyní už jsou Vaše možnosti opravdu neomezené. JavaScript má přístup k HTML ve formě DOMu^{[7][8]}, díky čemuž ho může číst a následně s ním jednoduše manipulovat. Můžete do elementů vkládat aktuální data, filtrovat tabulky, počítat, používat funkce, podmínky, cykly, a vlastně dělat cokoliv, co umí i jakýkoliv jiný programovací jazyk. Tedy pokud Vám nevadí omezení prohlížečem, ale to časem také obejdeme. Díky online repozitářům (například NPM) máte přístup k milionům knihoven, které můžete používat. Stránky jsou najednou živé a působí daleko lépe.

Ačkoliv se to tak nemusí podle názvu zdát, JavaScript má s Javou jen pramálo společného. Vznikl v roce 1995 během deseti dnů, tehdy ještě pod jménem Mocha a LiveScript. Javu má ve jméně víceméně jen proto, že v době jeho vzniku byla populární. Stojí za ním zaměstnanec Netscapu a standardy pro něj vyvíjí organizace Ecma International. Podle ní se značí i jeho verze (ES5, ES6, ...).^{[9][10][11]}

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 osvobodilo ho od sandboxovaného prostředí prohlížeče. Od té doby v něm lze napsat opravdu skoro všechno.

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 pár prázdných typů, jako je `undefined` při prázdné hodnotě, `null` při prázdném objektu a `NaN`, pokud něco počítáte a nevyjde číslo. Typy se nicméně kontrolují až při běhu, takže se na špatné přiřazení často přichází, až když je příliš pozdě.^{[12][13]}

Pokud píšete podmínky, musíte dávat pozor na porovnávání. Dvě rovnítka porovnávají typy, a tři hodnoty. Záměnou prvního rovnítka s vykřičníkem vznikne negace. `&&` znamená AND a `||` zase OR. Některé typy (`undefined`, `null`, `NaN`, prázdné řetězce) se označují jako *false* typy a vždy se v podmínkách přeloží na `false`. Dále můžete používat ternary operátory (`x ? y : z`, pokud je `x` pravda, vykoná se `y`, jinak `z`) nebo `&&` (`x && y`, pokud je `x` prázdný typ, vykoná se `y`), které 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().

Funkce (nebo metoda) je část kódu, který můžete zavolat a případně do něj vložit parametry. Na rozdíl od objektů se nejedná o reference, a tudíž nemůžete zevnitř funkce měnit hodnotu původní proměnné. Pokud funkce nic nevrací, její hodnota je automaticky undefined.

V JavaScriptu je toho samozřejmě mnohem více (třídy, objekty, ...), ale tohle by Vám mělo stačit k pochopení jeho základní struktury a principů. Jak jste si možná všimli, ve věcech jako typování a struktura vytvořená k účelu skriptů se podobá Pythonu, spoustu syntaktických věcí zase jazyk přebírá z Javy.

S vývojem webu se vyvíjely i jeho technologie. Dnes již skoro nikdo nedělá profesionální stránky pouze v HTML, CSS a JavaScriptu. Každá z těchto technologií má dnes svoji alternativu, a to i přesto, že nic jiného, než tyto tři jazyky, prohlížeče spustit neumí.

2.4 TypeScript

Typescript je jazyk, který se transpiluje do JavaScriptu, a díky tomu ho prohlížeče stále umí číst. Má ale několik výhod, které jemu příbuznému chybí.

Jednou z nich je podpora typování proměnných. O tom, jak typy hlídá JavaScript, jsem již mluvil. Ačkoliv se bez typů možná rychleji píše, nikdo nechce, aby se podobná chyba objevila například na produkci. Díky jejich podpoře tak nejen můžete přímo definovat, zda je proměnná číslo, řetězec, nebo objekt, ale hlavně se následná kontrola provádí už při kompilaci. Pokud ta selže, nevytvoří se ani finální stránka. Pro ty, kteří ani přesto nechtějí typy definovat, stále existuje možnost podstoupit veškerá rizika a označit proměnné jako `any`.

Další skvělou funkcí je podpora pro `enums`.^[14] Tato struktura existuje v několika dalších jazycích a můžete definovat řetězcové i číselné. Využít se dají, pokud máte neměnný seznam, ale jeho hodnoty nechcete číst pomocí indexů. Místo `colors[2]` můžete rovnou zavolat `Colors.BLUE` a kromě lepší čitelnosti zamezíte tomu, že barvu někdo v seznamu posune na jiné místo.

Dále můžete tvořit `interface` – rozhraní, které následně můžete použít jako šablonu při plnění objektů. Fungují i jako argumenty pro funkci, což Vám zvláště u těch větších ušetří spoustu místa.

Zpět k typům, je jich více druhů. Například můžete definovat `union` (proměnná může mít více typů) a `intersection` (proměnná musí mít kombinaci typů) typ. Dokonce můžete funkcím nastavit tzv. generický typ a ten pak použít například při převodu hodnoty z jednoho typu na druhý.^[15] A když už mluvím o funkcích, ty můžete nastavit jako privátní (nejdou volat mimo třídu), „přetížit je“ (udělat více funkcí se stejným jménem, kdy každá požaduje jiné argumenty), přidávat na ně dekorátory (modifikuje jejich chování) a následně je importovat nebo exportovat jako moduly.

2.5 Sass, Less a moduly

O CSS už jste četli. A i tato technologie má překvapivě pár nevýhod. Jedna z největších je fakt, že nemůžete znovu použít už napsaný kód. Takže pokud chcete na více místech tučné bílé písmo o velikosti 12 pixelů, musíte ho vždy znovu definovat. A pokud si po nějakém čase uvědomíte, že by písmo mělo být větší, nezbyvá Vám nic jiného, než jeho výšku na všech místech změnit a doufat, že jste na nic nezapomněl.

Tohle už se zřejmě několika lidem stalo, proto v roce 2006 vzniknul Sass^[16] a brzy po něm Less^[17]. Podobně jako TypeScript se jednalo pouze o nadstavbu nad stávající jazyk, který se v tomto případě zkompiloval do CSS. Dokázal ale vyřešit několik do té doby neřešitelných problémů.

Proměnné usnadnily život nejednomu programátorovi. Pokud si designér na poslední chvíli rozmyslel velikost fontu, stačilo ho změnit jen na jednom místě. K tomu se vážou i mixiny – malé části kódu, na které se dá kdekoliv odkázat. Navíc přijímají i parametry, umožňující přizpůsobit chování mixinu zvenku. Kromě toho tyto nadstavby zvládají aplikovat jednotlivé styly v závislosti na podmínkách, umožňují používat cykly (většinou pro aplikování stylů na třídy a identifikátory s podobným názvem), a také dokáží zpracovat vnořování jednotlivých stylů, což dělá soubory menší a kód čitelnější.

Jako poslední zmíním CSS moduly.^[18] 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 Vaší stránky rozbít. Tato jednoduchá knihovna dělá z globálních stylů lokální a při kompilaci ke každé třídě přidá vygenerovaný identifikátor. Protože jsou od teď ale názvy pravidel dynamické, musíte začít psát HTML pomocí JavaScriptu nebo TypeScriptu, který je umí zadat za Vás. Ideálně potřebujete najít způsob, jak HTML udělat více abstraktní a celý proces vykreslování zajišťovat skriptovacím jazykem.

2.6 JSX a React

JSX je syntax, díky které se konečně můžete zbavit statických HTML souborů a psát UI kompletně v JavaScriptu, a to se všemi známými tagy, které jste používali doted'.^[19] 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, znovupoužitelné, a dají se ovládat zvenku. Ty následně můžete spojovat do kompozic reprezentujících části stránek a ty zase do stránek samotných. Celá struktura je tak mnohem modulárnější a chování komponent všude stejné.

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 si toho všimne 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.^[20] 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 přenesení. Pokud nic překreslovat nechcete, můžete hodnotu uložit do `useRef()`^[21].

Jak ale React vlastně funguje?^[22] Pokaždé, když se změní stav, vytvoří virtuální kopii DOMu a v něm provede změny.^[23] 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, při detekci změny považuje za změněný element včetně všech jeho potomků.

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. Podobné chování má i `useCallback()`, ten ale zase ukládá definici funkce.

To by mohlo jako takový rychlý úvod do Reactu stačit. Jak jste asi poznali, úplně mění způsob, jak vnímáme a tvoříme stránky. Teď je před námi ale nový problém. Data jsou smíchaná s UI. Ideální by bylo mít nějakou datovou třídu se všemi stavy, kterou by mohly používat některé větší kompozice. K tomuto účelu se používá některý z manažerů stavů.

2.7 MobX a state management

To, že správa stavů v Reactu není úplně ideální, se ví už dlouho. Už jen to, že musíte data slučovat s vizuální částí, Vám téměř znemožňuje testování dat. A pokud by na jednom stavu záleželo více komponent, bylo by potřeba jej nějakým způsobem sdílet.

Existují desítky knihoven^[24], které Vám umožňují 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. V souboru poblíž kompozice si vytvoříte třídu a do ní vložíte všechny stavy, které ve Vaší kompozici 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`^[25], který se přepočítá pouze tehdy, pokud se některá ze závislostí (použitých observable proměnných) změní.

2.8 API

Frontend je nyní de facto za Vámi. Abyste ale mohli komunikovat se serverem, který se stará o přístup do databáze a posílání stránek uživatelům, potřebujete nějaký univerzální most, který Vás s ním spojí. A právě tomu se říká API^[26].

Používáte ho denně, a ani o tom možná nevíte. Pokud potřebuje stránka dynamická data, musí o to požádat server HTTP požadavkem. Server po jeho vyhodnocení zase použije nějakou jinou API, díky níž se dostane do databáze, a až dostane odpověď, stejným HTTP požadavkem odpoví uživateli. Jeho stránka tak teď může konečně zobrazit data.

Důvodem, proč se většina dat posílá skrze JSON a XML, je jejich jednoduchá extrakce. Například knihovna HTMX ale přijímá data pouze v HTML. To může mít několik nevýhod. Data se z HTML těžko dostávají, a pokud je budete v tomhle formátu ukládat v databázi, pak už je nemůžete použít ani na jiné stránce, ani třeba v mobilní aplikaci. To je velká výhoda právě JSON formátu. Je podobně velký, jako kdybyste posílali normální text, ale zároveň z něj všude dokážete udělat datový objekt a používat jeho obsah.

roze-psat
různ-
druhy
API

2.9 Storybook a mocky

Doteď jste mohli své komponenty testovat a vidět jen na finální stránce. To je ale poněkud nepraktické, protože musíte čekat na kompilaci celého projektu, jste připojení na reálná data a komponenty nejsou izolované. Proto 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 řetězců 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^[27]. 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 kdokoliv může přečíst, trvá odesílání kódu déle a některé novější funkce JavaScriptu nemusí daný prohlížeč znát. Z tohoto důvodu se používají bundlery jako je Webpack^{[28][29]} nebo Vite, a jejich loadery.

Na začátku kompilace si Webpack najde vstupní soubor (většinou index.js) a rekurzivně projde všechny jeho importy^{[30][31]}, čímž si poskládá vlastní graf závislostí^[32]. 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 většinou prohlížečů bez ztráty funkčnosti. Existují ale i moduly na 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.

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. Ačkoliv je to nepochybně správně, tak je tento jazyk automaticky omezený pouze na psaní frontendu. Tedy byl, dokud někdo nevytvořil právě Node.js.^[33]

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.^[34]

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^[35], nástavba Node.js s podporou Reactu, který dokáže celý web posunout 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>)^[36], 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. Některé, třeba 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^[37] – na serveru^[38] nebo na klientovi^[39]. Obojí má své pro a proti. Na serveru můžete přímo volat do databáze, cachovat výsledky a použít je jinde, a posílat uživatelům již vygenerované stránky, takže nemusí čekat, než jim je React poskládá. Na druhou stranu zase nemůžete používat `useState()`, `useEffect`, poslouchat eventy, nebo přistupovat k DOMu a do prohlížeče. Mezi oběma způsoby je pomyslná síťová hranice, a je jen na Vás, kde si ji při vymýšlení architektury a stavby projektu stanovíte.

Optimalizace fontů a obrázků

Pokud používáte na své stránce jakýkoliv font, co 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 kompilaci 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 />`. 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.

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^[40]. Vizualní, 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 o ty se Jest stará.

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^[41]. 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^[42]. 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)^[43]. A přesně to budete potřebovat, pokud plánujete projekt posunout do zahraničí a přidat více než jeden jazyk. I na tohle existují frameworky. A ten nejpůvodnější se jmenuje i18next^[44].

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 víc, než jen hledat v překladovém slovníku^[45]. Dvojitými složenými závorkami můžete do překladů přidat proměnné a ty za chodu měnit. Můžete také určit, jak se přeloží počet, pokud na konec klíče připíšete podtržítko a `one` (jednotné číslo), `two` (dva předměty), `other` (množné číslo), nebo dokonce `zero` (žádný předmět). Máte dokonce i možnost podle konkrétního jazyka formátovat čísla.

Veškeré překlady se ale pořád dělají někde v projektu. A zatímco programátoři nejsou moc dobří v překládání, většina překladatelů zase asi nebude schopná si naklonovat repositář, upravovat JSON (nebo i jiný formát) a pak vše nahrát zpět na server (nedej bože u toho řešit git konflikty). Proto existují různá řešení na webu. A jedno z nich, které ve Worldee používáme, je Localazy.

Tento, jak jsem později zjistil, český startup, 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 zabudované překladače, podpora plurálů ve všech světových jazycích, slovníky, nebo řešení duplicit^[46]. Zároveň se dá integrovat nejen se všemi známými frontendovými frameworky, ale i s Figma, Unity engine, GitHubem, a dokonce Excel^[47].

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^[48]. 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.

Kam to bude? Kdy jedeš?

MŮJ ÚČET

MŮJ účet

Sociální sítě

Soukromí účtu

Změna hesla

Upozornění

Premiová verze

Platby

Doporučení

Odhlásit se

NA WORLDDEE VYSTUPOVAT

Pod jménem

Tvůj profil najdeš pod odkazem: <https://www.worlddee.com/cz/dlaboja>

PŘEZDÍVKA

dlaboja

TVOJE JméNO

Jan

PŘÍJMENÍ

Dlaboja

E-MAIL

TELEFON

MĚSTO

Olomouc

STÁT

Česká republika

POHLAVÍ

muž

DATUM NAROZENÍ

DD MM RRRR

O MNĚ

B / T L I E O

Občas programuju

Obrázek 3.1: Staré nastavení účtu napsané v Nette

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 kód co nejlépe znovu využívat, shlukuje se do samostatných komponent. Takhle se dá znovu a znovu použít na několika místech v projektu. Aby byla správně napsaná a kód co nejméně komplexní, je potřeba dodržovat pravidla jako izolovanost (její funkčnost by neměla být podmíněná jinou komponentou), „indempotentnost“ (jeden vstup bude mít vždy jen jeden výstup, něco jako funkce v matematice), neměnnost propů a stavů, a několik dalších.

K izolovanosti se řadí i pravidlo nestylovat komponenty zvenku. A pokud chcete měnit vlastnosti zahrnující styl, pak komponentě posílejte parametrem například výšku a podle hodnoty jí nastavte styl až uvnitř. Tímto se zároveň dodrží i smysl CSS modulů a nebudete mít různě po webu deset variant jedné komponenty.

Div

Protože podle pravidel Reactu byste nikdy neměli stylovat komponentu přímo, je třeba si poradit jinak. Styly můžete pořád přidávat na primitivní HTML elementy, aniž byste integritu komponent jakkoliv narušili. 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.

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 sami vytváříme jako SVG obrázky ve složce custom.

Tlačítko

Tlačítek můžete na webu najít osm druhů, každé z nich používá trochu jinou barevnou paletu (modré, 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

Ačkoliv se na první pohled může zdát, že se jedná pouze o Next.js `<Link />` element, zde můžete dělat ještě několik věcí navíc. Místo psaní url se zde používají `ActionData`, objekt, kterým skládáme cesty na všechny stránky projektu. Dokonce umí otevírat i soubory, nebo stránky v novém panelu.

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

Pokud potřebujete vybrat více možností, musíte použít nějakou formu combo boxu. Ten základní, který je jen rozšířená verze toho z knihovny `react-select`, má podobné parametry

co Text box, ale je navíc obohacen o pole hodnot. Pokud je hodnot moc, dá se povolit hledání podle vstupu z klávesnice a pro přívětivou práci s mobilem se na něm roztáhne přes část obrazovky.

Combo boxů máme ještě tři typy. Zatímco ten první obsahuje všechny země světa i s jejich vlajkami, druhý má zase uložené veškeré světové měny i s jejich značkou. Třetí jsem vytvořil teprve nedávno a obsahuje všechny telefonní předvolby (opět i s vlajkami).

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 v kompozici. 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 malou komponentu, do které 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), obě k tlačítkům ještě přidaly klíč, aby je React zvládnul lépe vykreslit. 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.

Desktopovou verzi asi nemusím příliš popisovat. Horní část byla ve ScrollShadow komponentě, což poslední odkazy dokázalo skrýt, pokud by jich bylo hodně a nevešly by se na obrazovku. Zajímavější je ovšem verze pro mobil. Je totiž založená na již zmíněné SwipeableDrawer komponentě a dá se ovládat změnou proměnné open.

Pokud toto menu chcete kdekoliv použít, stačí mu předat menuItems parametr, složený z pole obsahujícího text na tlačítko, odkaz, ikonu a zda je aktivní (k tomu mi skvěle poslouží Next.js hook usePathname(), který vrací aktuální url). Změny, které se promění čistě na mobilu, zase předávám parametry isOpen, akci po zavření onClose a titulkem header.

Account Container

Protože má každá z podstránek stejnou horní část, a také, aby se mi lépe umísťovaly, vytvořil jsem pro každou z nich kontejner. Ten si podle aktuální cesty tvoří jméno stránky a v mobilní verzi se mu objevuje tlačítko, kterým se dá otevírat postranní menu. Do komponenty samotné se pak vkládají jednotlivé stránky, které Vám nyní představím.

 Můj účet

 Sociální sítě

 Změna hesla

 Upozornění

 Platby

 Obsah a profil



dlabaja
Odhlásit se

Obrázek 3.2: *Menu*

Podstránka Můj účet

Stránka s účtem se dá označit jako srdce celého nastavení. Na první pohled Vás určitě zaujal banner přes celou stránku, 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, přezdívkou, pod čím chcete vystupovat, email, telefon, a sekci O mně, ve které se můžeme představit.


Ačkoliv komponenta s názvem města Olomouc vypadá jako jen další Text box, není tomu tak. Pokud do ní totiž začnete psát, začnou se vám díky Google Autocomplete funkci zobrazovat návrhy měst (případně i míst). 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 Modal – 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

Podstránka Sociální sítě

Na této stránce si s Worldee můžete propojit svůj Facebook nebo Instagram účet, který se Vám 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 napsat URL Vašeho profilu na jedné z těchto sítí. Pokud bylo úspěšné, 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

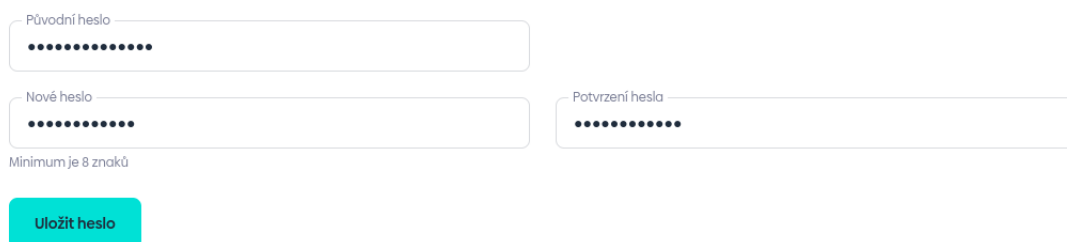
Propojit účet

Obrázek 3.5: Podstránka Sociální sítě

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 po nastavení nového.

Změna hesla



Původní heslo

Nové heslo

Potvrzení hesla

Minimum je 8 znaků

Uložit heslo

Obrázek 3.6: Podstránka Změna hesla

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

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

[Uložit změny](#)

Obrázek 3.7: Podstránka Upozornění

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 asijští 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

Variananta 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

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. Všechny className atributy jsem přesunul do <div /> 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

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 už víte. Asi vás proto nepřekvapí, že svůj vlastní store má každá z již vyjmenovaných stránek. Díky tomu nemusím používat `useState()` a kazit si tak svá data Reactem.

Store stránek je třída složená z polí obsahujících všechna data, která se budou na stránce používat, jejich náležitých getterů (a většinou i setterů) a konstruktoru. Všechna pole, která se mění, musí být označena dekorátorem `@observable` a jejich změna musí být obalená buďto v `runInAction()` funkci nebo `@action` dekorátoru.

Konstruktorem by měl správně jen dosadit základní hodnoty, spustit nějaké interní funkce a zavolat `makeObservable(this)`, čímž aktivuje MobX. Externí data z API by ho mohly zbrzdit nebo vyvolat výjimku, proto by si je měl vývojář zavolat sám. Proto má každá ze stránek svoji veřejně přístupnou metodu `init()`, která se v Reactu volá po prvním renderu. Tím ovšem vzniká další problém – když se stránka načte, ještě nemá data. Proto jsem si vytvořil vlastní funkci `load()`, 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.

Každý ze storů 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.

Pokud je na stránkách potřeba nějaké volání na API, i to naleznete zde. Kromě metody init() a save(), které jsou všude, mají některé stránky i své vlastní metody – třeba na smazání a nahrání fotky v nastavení účtu, nebo připojení Facebooku a Instagramu v sociálních sítích.

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 prošla.

Pokud bylo vráceno true, spustí se část, kterou mají všechny stránky stejnou – funkce save(). Podobně jako load() bere jako argument metodu, kterou má spustit (každý store má kromě init() i svůj save()), a také již zmíněné 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

Postupně se dostávám výš a výš v naší frontendové hierarchii. Všechna data, která jsem dostával ve storu, jsou vlastně jen datové třídy. Konkrétně se jedná o třídy s definovanými poli reprezentujícími jejich obsah, gettery, a konstruktorem, jehož jediný parametr je interface se všemi dosaditelnými hodnotami, kterým třídu můžeme naplnit. Protože nemá settery, jde to jen jednorázově, a tak veškeré změny proměnných řeší až store a já zde nemusím používat MobX a @observable dekorátory.

Zhruba na stejné úrovni jako data mohou být i konvertory. Ne vždy jsou totiž data z databáze v pro nás ideálním formátu. Například čas i enumy jsou vyjádřené jako string, s čímž se ale špatně pracuje a museli bychom ho 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 odchytn 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. Ten vrací mock datové třídy, díky čemuž nejsou potřeba žádné konvertory. 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.ts.

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 jde zjistit pomocí tvorby falešných dat, jejich dosazením a následně porovnáním všech veřejných getterů.

Testům se samozřejmě nevyhnou 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 nejkompexnější, protože musí pokrýt všechno od přidávání a získávání dat, validací a komunikací 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, že není prázdná. Email už zase musel mít správnou strukturu. A Appears, 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 jednu část jména a „Pod přezdívkou“ zase požadoval nastavenou přezdívku, která se validovala v předchozím kroku.

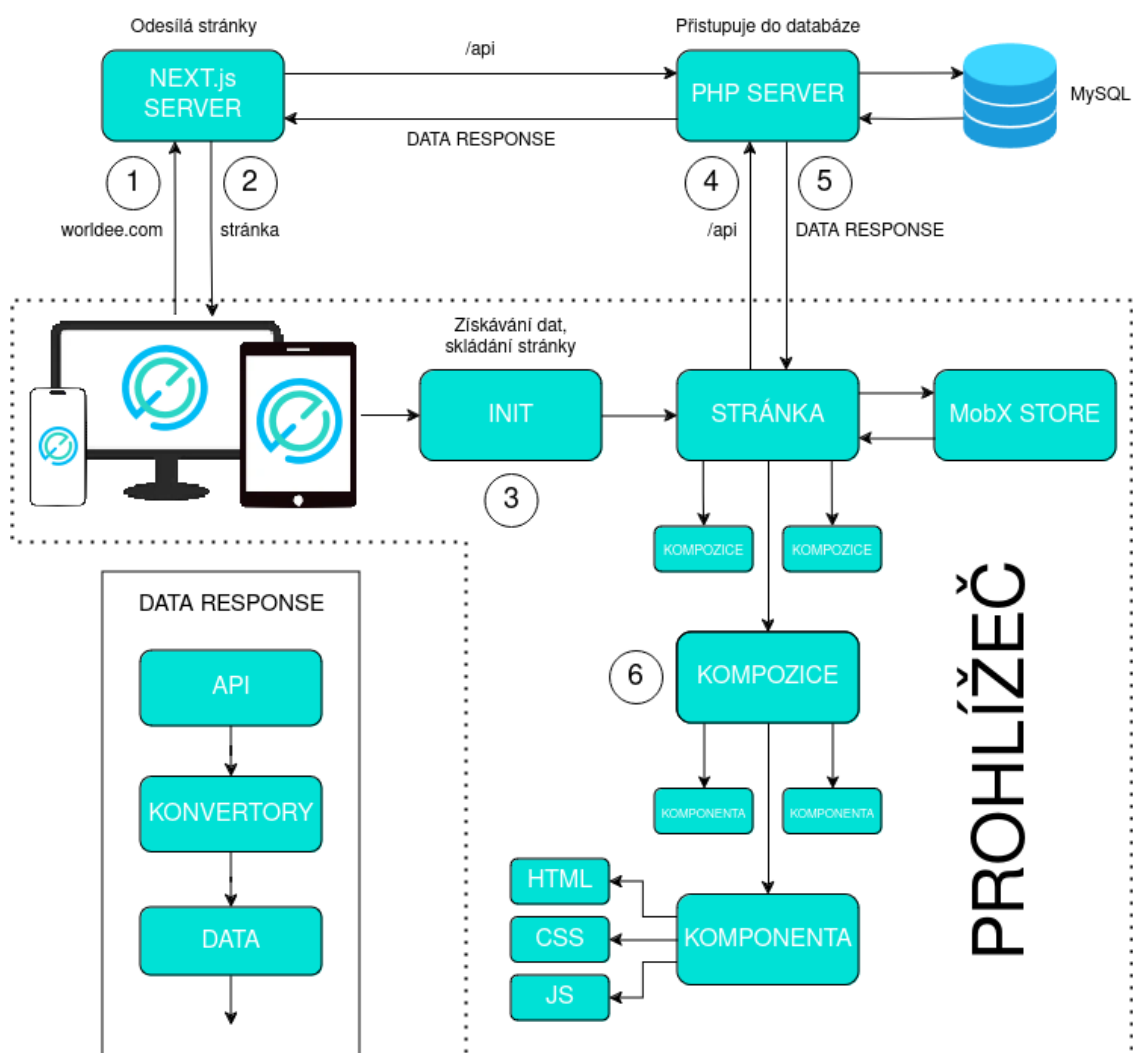
Následně se testují všechny metody integrující API volání. Ačkoliv jsou všechny z nich jen mocky a ve skutečnosti na server nevolají, můžu vyzkoušet, zda se pod nimi zabalená API metoda zavolala s těmi parametry, které jsem jejímu rodiči předal. K tomu se výborně hodí SpyObject.

3.3 Finální struktura stránky

The screenshot displays the 'Můj účet' (My Account) page. The top navigation bar includes links for 'Cesty', 'Průvodce destinacemi', 'O nás', and 'Travel Buddies'. A search bar is present with the text 'Kam to bude?' and 'Kdy jede?'. The user is logged in as 'diabaja'. The left sidebar contains links for 'Můj účet', 'Sociální sítě', 'Změna hesla', 'Upozornění', 'Platby', and 'Obsah a profil'. The main content area shows the user's profile picture, a bio section with a rich text editor, and a 'Uložit změny' button. The bio section includes fields for 'Jméno' (diabaja), 'Příjmení', 'Přezdívka' (diabaja), 'E-mail' (diabajahonza@gmail.com), and 'Telefon'. There are also dropdown menus for 'Na WorldDee vystupovat' (Pod jménem) and 'Město' (Česká republika). A section for 'Odstranění účtu' is also visible.

Obrázek 3.10: Vizuální podoba stránky, dostupná po přihlášení na www.worlddee.com/settings/account

popsat
ar-
chitek-
turu



Obrázek 3.11: Diagram architektury frontendu

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. 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, zkompilovaný 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 účastním 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, nebo mapuju data ze scraperu na naše API. Tvorba webů je něco, v čem jsem se momentálně našel, a co ještě nějakou chvíli budu dělat. A chtěl bych poděkovat celému vývojovému týmu, že mi pomáhají na cestě za jeho pochopením.

Seznam použitých zdrojů

- [1] Tomáš Zapletal. *Toto je Worldee*. Unpublished. 2024.
- [2] We travel the world s.r.o. *Vyber si svůj způsob cestování*. 2024. URL: <https://www.worldee.com> (cit. 25. 11. 2024).
- [3] Kovy. *Ultimátní roadtrip Skotskem! w/Míra | KOVY*. 2024. URL: <https://www.youtube.com/watch?v=66TMsTKaIbE> (cit. 25. 11. 2024).
- [4] 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).
- [5] spol. s r.o. AliaWeb. *We travel the world s.r.o.* 2024. URL: <https://www.podnikani.cz/08351864/we-travel-the-world-sro> (cit. 28. 09. 2024).
- [6] Chris Coyier. *CSS Flexbox Layout Guide*. 2024. URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox> (cit. 14. 09. 2024).
- [7] MDN contributors. *Introduction to the DOM*. 2024. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (cit. 11. 08. 2024).
- [8] W3schools. *JavaScript HTML DOM*. 2024. URL: https://www.w3schools.com/js/js_htmlDOM.asp (cit. 02. 11. 2024).
- [9] Ilya Kantor. *An Introduction to JavaScript*. 2024. URL: <https://javascript.info/intro> (cit. 11. 08. 2024).
- [10] 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).
- [11] aryash. *History of JavaScript*. 2024. URL: <https://www.geeksforgeeks.org/history-of-javascript> (cit. 06. 10. 2024).

- [12] MDN contributors. *JavaScript basics*. 2024. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics (cit. 06. 10. 2024).
- [13] MDN contributors. *JavaScript data types and data structures*. 2024. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures (cit. 06. 10. 2024).
- [14] David Bolton. *What Is an Enum in Programming Languages?* 2024. URL: <https://www.thoughtco.com/what-is-an-enum-958326> (cit. 14. 08. 2024).
- [15] TypeScript contributors. *Generics*. 2024. URL: <https://www.typescriptlang.org/docs/handbook/2/generics.html> (cit. 09. 10. 2024).
- [16] 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).
- [17] 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).
- [18] css-modules. *CSS Modules*. 2024. URL: <https://github.com/css-modules/css-modules> (cit. 14. 08. 2024).
- [19] Sanity. *JSX definition*. 2024. URL: <https://www.sanity.io/glossary/jsx> (cit. 04. 10. 2024).
- [20] React team a external contributors. *State: A Component's Memory – React*. 2024. URL: <https://react.dev/learn/state-a-components-memory> (cit. 14. 08. 2024).
- [21] React team a external contributors. *useRef – React*. 2024. URL: <https://react.dev/reference/react/useRef> (cit. 14. 08. 2024).
- [22] FrontStart. *How React ACTUALLY works (DEEP DIVE 2023)*. 2024. URL: <https://www.youtube.com/watch?v=za2FZ8QCE18> (cit. 02. 11. 2024).
- [23] PurelyFunctional tv. *React and the Virtual DOM*. 2015. URL: <https://www.youtube.com/watch?v=BYbgopx44vo> (cit. 02. 11. 2024).
- [24] Alex Kugell. *7 Top React State Management Libraries*. 2022. URL: <https://trio.dev/7-top-react-state-management-libraries> (cit. 18. 08. 2024).

- [25] MobX contributors. *Deriving information with computed*s. 2024. URL: <https://mobx.js.org/computed.html> (cit. 18.08.2024).
- [26] 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).
- [27] Storybook contributors. *Mocking modules*. 2024. URL: <https://storybook.js.org/docs/writing-stories/mocking-data-and-modules/mocking-modules> (cit. 18.08.2024).
- [28] Wikipedia contributors. *Webpack*. 2024. URL: <https://en.wikipedia.org/wiki/Webpack> (cit. 18.08.2024).
- [29] Fireship. *Module Bundlers Explained... Webpack, Rollup, Parcel, and Snowpack*. 2020. URL: <https://www.youtube.com/watch?v=5IG4UmULyoA> (cit. 02.11.2024).
- [30] Webpack Contributors. *Concepts*. 2024. URL: <https://webpack.js.org/concepts/> (cit. 18.08.2024).
- [31] 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).
- [32] Webpack Contributors. *Dependency Graph*. 2024. URL: <https://webpack.js.org/concepts/dependency-graph> (cit. 18.08.2024).
- [33] Benjamin Semah. *What Exactly is Node.js?* 2022. URL: <https://www.freecodecamp.org/news/what-is-node-js> (cit. 21.08.2024).
- [34] Wikipedia contributors. *Node.js*. 2024. URL: <https://en.wikipedia.org/wiki/Node.js> (cit. 06.10.2024).
- [35] nextjs.org. *Introduction*. 2024. URL: <https://nextjs.org/docs> (cit. 21.08.2024).
- [36] nextjs.org. *Learn Next.js*. 2024. URL: <https://nextjs.org/learn> (cit. 25.08.2024).

- [37] nextjs.org. *Next.js Server and client components*. 2024. URL: <https://nextjs.org/learn/react-foundations/server-and-client-components> (cit. 01.09.2024).
- [38] nextjs.org. *Next.js Server Components*. 2024. URL: <https://nextjs.org/docs/app/building-your-application/rendering/server-components> (cit. 01.09.2024).
- [39] nextjs.org. *Next.js Client Components*. 2024. URL: <https://nextjs.org/docs/app/building-your-application/rendering/client-components> (cit. 01.09.2024).
- [40] web.dev. *Types of automated testing*. 2024. URL: <https://web.dev/learn/testing/get-started/test-types> (cit. 04.09.2024).
- [41] 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).
- [42] 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).
- [43] Robert Sheldon. *What is internationalization?* 2024. URL: <https://www.techtarget.com/whatis/definition/internationalization-I18N> (cit. 25.11.2024).
- [44] i18next team. *Introduction*. 2024. URL: <https://www.i18next.com> (cit. 04.09.2024).
- [45] locize. *i18next Crash Course | the JavaScript i18n framework*. 2022. URL: https://www.youtube.com/watch?v=SA_9i4TtxLQ (cit. 04.09.2024).
- [46] Localazy. *Localization for teams of any size*. 2024. URL: <https://localazy.com/pricing> (cit. 07.09.2024).
- [47] Localazy. *Integrate Localazy*. 2024. URL: <https://localazy.com/integrations> (cit. 07.09.2024).

- [48] David Grudl. *How Do Applications Work?* 2024. URL: <https://doc.nette.org/en/application/how-it-works> (cit. 08.09.2024).

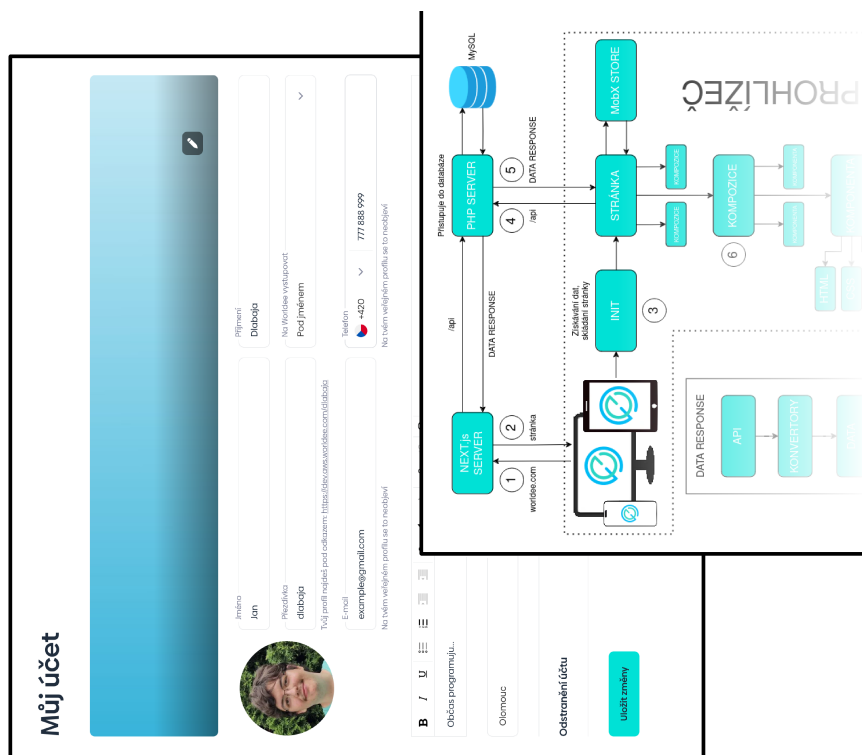
Seznam obrázků

1.1	Logo firmy Worldee	7
3.1	Staré nastavení účtu napsané v Nette	32
3.2	Menu	38
3.3	Podstránka Můj účet	40
3.4	Dialog s odstraněním účtu	41
3.5	Podstránka Sociální sítě	42
3.6	Podstránka Změna hesla	43
3.7	Podstránka Upozornění	44
3.8	Podstránka Obsah a profil	45
3.9	Podstránka Platby	46
3.10	Vizuální podoba stránky	51
3.11	Diagram architektury frontendu	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

