

UNICORN VYSOKÁ ŠKOLA s.r.o.

Softwarový vývoj



Bakalářská práce

Webové řešení na prodej vstupenek s rezervací míst

Autor bakalářské práce: Filip Ditrich

Vedoucí bakalářské práce: Ing. Marek Beránek, Ph.D.

Praha 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení	Filip Ditrich
Studijní program	Softwarový vývoj
Název práce	Webové řešení na prodej vstupenek s rezervací míst

Cíl

Tato bakalářská práce se zabývá problematikou implementačního řešení prodeje vstupenek s využitím rezervace míst. Cílem práce je v teoretické části tuto problematiku rozvést a identifikovat nejhlavnější technické výzvy a požadavky na implementaci takového systému a to zejména z pohledu frontendu.

Tyto výzvy, typu intuitivní uživatelské prostředí, dostupnost informací o prodeji v reálném čase, administrace a správa sedačkových plánů či finální rezervace a platba objednávky, budou v této práci popsány a budou uvedeny možnosti jejich řešení.

Praktická část bude zaměřena na implementaci části webové aplikace zabývající se vykreslováním interaktivního plánu sedaček a jeho interakce se zákazníkem. V této části bude představen návrh uživatelského rozhraní spolu s představením vybraných technologií k implementaci. Důraz bude kláden na optimalizované řešení plánu sedaček, specifikaci jeho datového formátu, komunikaci s API a celkové dokumentaci komponent použitých ve výsledné aplikaci.

Cílem této práce je popsat problematiku a výzvy při implementaci webové aplikace řešící sedačkový prodej vstupenek a představit část reálného řešení takovéto aplikace zejména z pohledu interaktivního plánu na výběr sedaček.

Osnova

1. Teoretická část
 1. Problematika prodeje vstupenek
 2. Hlavní výzvy a požadavky
 1. Uživatelské prostředí
 2. Dostupnost míst v reálném čase
 3. Různé typy plánů míst
 4. Administrace, back-office
 5. Rezervace a tvorba objednávky
 6. Platba a integrace platebních bran
2. Praktická část
 1. Návrh uživatelského prostředí
 2. Vybrané technologie k implementaci
 3. Backendové řešení
 4. Specifikace datového formátu
 5. Implementace interaktivního plánu míst
 6. Dokumentace komponent

Doporučená literatura

Základní literatura:

Anatomy of seating plans. Seat mapping done right – Seatmap.pro [online].

Copyright © 2022. Dostupné z: <https://seatmap.pro/blog/seating-plan-anatomy/>

Seating plans. How do we render?. Seat mapping done right – Seatmap.pro

[online]. Copyright © 2022. Dostupné z: <https://seatmap.pro/blog/seating-planrendering/>

Let's Get Graphic: A Few Ways To Draw On The Web - Stack Overflow Blog.

Stack Overflow Blog - Essays, opinions, and advice on the act of computer

programming from Stack Overflow. [online]. Copyright © 2022 All Rights

Reserved. Dostupné z: <https://stackoverflow.blog/2019/11/06/lets-get-graphic-a-few-ways-to-draw-on-the-web/>

5 Best Practices for Developing an Online Booking Ticket System | by Yevheniia

Korotia | Medium. Medium – Where good ideas find you. [online]. Dostupné z:

<https://medium.com/@Eugeniya/5-best-practices-for-developing-an-onlinebooking-ticket-system-797d777a7ed0>

HTTP and Websockets: Understanding the capabilities of today's web communication technologies | by Thilina Ashen Gamage | Platform Engineer |

Medium. Medium – Where good ideas find you. [online]. Dostupné z:

<https://medium.com/platform-engineer/web-api-design-35df8167460>

All You Need To Know About Payment Gateway Integration In Mobile Apps | by

Softensy | Medium. Softensy – Medium [online]. Dostupné z:

<https://softensydevs.medium.com/all-you-need-to-know-about-paymentgateway-integration-in-mobile-apps-dd626039d7eb>

Vedoucí bakalářské práce

Ing. Marek Beránek, Ph.D.

Adresa pracoviště

V Kapslovně 2767/2, 13000 Praha 3

Datum zadání bakalářské práce

12/2022

Termín odevzdání bakalářské práce

Podle rozhodnutí rektora

V Praze dne 28.06.2023



prof. Ing. Jan Čadil, Ph.D.
Rektor

Čestné prohlášení

Prohlašuji, že jsem svou bakalářskou práci na téma *Webové řešení na prodej vstupenek s rezervací míst* vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím výhradně odborné literatury a dalších informačních zdrojů, které jsou v práci všechny citovány a jsou také uvedeny v seznamu použitých zdrojů.

Jako autor této bakalářské práce dále prohlašuji, že v souvislosti s jejím vytvořením jsem neporušil autorská práva třetích osob a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb.

Dále prohlašuji, že odevzdaná tištěná verze bakalářské práce je shodná s verzí, která byla odevzdána elektronicky.

V dne

Filip Ditrich

TODO: Poděkování

Rád bych touto cestou srdečně poděkoval vedoucímu bakalářské práce Ing. Markovi Beránkovi, Ph.D. za čas věnovaný vedením této práce, za příjemnou spolupráci a za cenné poskytnuté rady a připomínky.



Webové řešení na prodej vstupenek s rezervací míst

Web-based ticketing solution with seat
reservation

UNICORN
— **UNIVERSITY**

TODO: Abstrakt

Tato bakalářská práce se zaměřuje na vývoj frontendové části webové aplikace pro prodej vstupenek s rezervací míst. Cílem práce je vyvinout prototyp aplikace, která potenciálnímu zákazníkovi umožní zobrazit mapu míst, vybrat si preferované místo, přidat vstupenky do nákupního košíku a následně vytvořit objednávku.

Teoretická část práce se zaměřuje na obecnou problematiku prodeje vstupenek a moderního řešení pomocí platform a služeb poskytujících online prodej vstupenek s možností rezervací míst. Tato část dále analyzuje trh současných poskytovatelů a definuje nejhlavnější technické problémy, které mohou při vývoji takového systému vzniknout a to výhradně z pohledu frontendu.

Praktická část práce definuje rozsah implementované aplikace, podrobně popisuje hlavní funkce, komponenty, datové modely i některé nezbytné backendové části. Kapitola o návrhu uživatelského rozhraní popisuje principy, vzory a osvědčené postupy návrhu uživatelského rozhraní v kontextu vyvíjené aplikace. Kapitola o vývoji frontendové části poté podrobně popisuje technologie, nástroje a knihovny použité při vývoji aplikace.

TODO: Abstract

This bachelor thesis deals with the implementation of a ticketing solution using seat reservations. The aim of the theoretical part of the thesis is to elaborate this issue and identify the main technical challenges and requirements for the implementation of such a system, especially from the frontend perspective. These challenges, such as intuitive user interface, availability of information real-time sales information, administration and management of seating plans or final booking and payment of orders, will be discussed in this thesis will be described in this thesis and options for their solution will be presented. The practical part will focus on the implementation of the part of the web application dealing with rendering of the interactive seating plan and its interaction with the customer. In this part the user interface design will be presented together with an introduction of the selected technologies to be implemented. Emphasis will be placed on the optimized design of the seating plan, the specification of its data format, communication with the API and overall documentation of the components used in the resulting application. The aim of this work is to describe the issues and challenges in implementing a web application addressing seat ticketing and present part of a real solution for such an application especially from the perspective of an interactive seat selection plan.

Keywords: interactive seating plan, seat reservations, tickets, web applications, JavaScript, React, SVG

Obsah

Úvod	12
Prodej vstupenek	12
Cíle práce	13
1 Specifikace prototypu	15
1.1 Interaktivní mapa	15
1.1.1 Rozložení a struktura	15
1.1.2 Barevné kódování	17
1.1.3 Ovládání mapy	18
1.1.4 Stav a informace o sedadlech	19
1.1.5 Data a jejich dostupnost	21
1.2 Nákupní košík	21
1.2.1 Správa dat	22
1.2.2 Přehled obsahu košíku	22
1.2.3 Rezervace míst	24
1.3 Dokončení objednávky	25
1.3.1 Osobní informace	25
1.3.2 Výběr platební metody	26
1.3.3 Souhrn objednávky	29

1.3.4	Vytvoření objednávky a potvrzení	29
2	Návrh uživatelského rozhraní	30
2.1	Principy návrhu uživatelkého rozhraní	32
2.2	Aplikovaná psychologie na UI/UX	34
2.2.1	Maslowova hierarchie potřeb	34
2.2.2	Hierarchie potřeb v návrhu uživatelského rozhraní	35
2.3	Uživatelské příběhy	38
2.3.1	Co jsou uživatelské příběhy	38
2.3.2	Psaní efektivních uživatelských příběhů	39
2.3.3	Návrh konkrétních uživatelských příběhů	40
2.4	Nástroje pro návrh	43
2.4.1	Adobe XD	43
2.4.2	Figma	45
2.4.3	Sketch	47
2.4.4	Výběr nástroje	48
2.5	Návrh produktu	51
2.5.1	Vizualizace místa konání	51
2.5.2	Výběr sedadel	52
2.5.3	Nákupní košík	55
2.5.4	Vyřízení a potvrzení objednávky	58
3	Implementace aplikace	63
3.1	Úvod do vývoje frontendu	64
3.2	Výběr technologií	65
3.2.1	React.js	65

3.2.2	TypeScript	66
3.2.3	Next.js	66
3.2.4	Mantine UI	67
3.2.5	Tailwind CSS	67
3.2.6	Ostatní technologie	68
3.2.7	Správa zdrojového kódu	70
3.3	Struktura a architektura projektu	71
3.3.1	Vytvoření projektu	71
3.3.2	Základní architektura	72
3.4	Interaktivní mapa sedadel	74
3.4.1	Struktura	74
3.4.2	Získávání a správa dat	75
3.4.3	Parsování SVG a mapování sedadel	77
3.4.4	Interaktivita s VirtualMap	78
3.4.5	Výběr sedadla a správa vstupenek	79
3.5	Správa košíku	81
3.5.1	Kontext košíku	81
3.5.2	Stav a funkce košíku	81
3.5.3	Vizualizace košíku	83
3.5.4	Integrace API	83
3.6	Rezervační systém	87
3.6.1	Expirace a vymazání rezervace	88
3.6.2	Vizualizace rezervace	89
3.7	Vyřízení a potvrzení objednávky	91

3.7.1	Metoda platby	92
3.7.2	Souhrn objednávky	92
3.7.3	Potvrzení objednávky	94
3.8	Nasazení aplikace	96
3.8.1	Proces nasazení	96
3.8.2	Výsledek nasazení	96
3.9	Závěr	98
4	Výzvy a problémy	100
4.1	Vykreslování interaktivního plánu sezení pomocí SVG	100
4.2	Absence API a jeho simulace	101
4.3	Zachování jednoduchosti v komplexní aplikaci	102
4.4	Shrnutí	102
Závěr		103
Literatura		105
Seznam obrázků		110
Seznam tabulek		112
Seznam použitých zkratek		114
Přílohy		115

Úvod

Prodej vstupenek

Prodej vstupenek na kulturní a jiné různé události je důležitou součástí zábavního průmyslu, neboť poskytuje lidem přístup na koncerty, divadelní představení, sportovní či jiné události. Prodej vstupenek umožňuje pořadatelům těchto akcí nejen kontrolovaný průběh akce, ale především generuje dostatečný finanční tok peněz před konáním jejich akce. Tyto finance zpravidla potřebují pro zajištění všech potřebných prostředků pro uspořádání akce a pro pořadatele se tedy jedná o jeden z klíčových faktorů úspěchu konání akce. Potřebují tedy pro zákazníky zajistit co nejsnadnější a nejpříjemnější možnost nákupu vstupenek.

S nástupem moderních technologií se online prodej vstupenek proměnil v atraktivní a preferovaný způsob nákupu vstupenek, nabízející zákazníkům snadný, pohodlný a hlavně rychlý způsob nákupu vstupenek, aniž by se museli kamkoliv fyzicky dostavit. Tento nový moderní způsob prodeje vstupenek však nabízí výhody nejen zákazníkům, ale také pořadatelům akcí. Systémy, které jsou na tomto způsobu prodeje vstupenek založeny, pořadatelům akcí umožňují bezproblémový prodej vstupenek, což vede k efektivnějšímu plánování a řízení akcí. Tyto systémy pořadatelům také nabízejí cenné údaje o zákaznících, jejich preferencích a chování, které mohou využít při plánování marketingových strategií, cílených reklam či propagačních akcí za účelem zvýšení zapojení zákazníků a podpoření prodeje.

Jedním z nejvýznamnějším pokrokem v této oblasti online řešení prodeje vstupenek bylo rozšíření o možnost rezervace míst v prostoru konání akce. Toto řešení nově zákazníkům umožňuje zarezervovat si místo na dané události, což opět přináší několik výhod pro zákazníky, ale také pro pořadatele akcí. Zákazníkům umožňuje rezervaci a výběr místa, které je pro ně nevhodnější. Pořadatelům akcí pak rezervace míst umožňuje předem plánovat kapacitu dané akce a také zjistit, jaké místo je pro zákazníky nejvíce preferované. Dále také značně snižuje počet

možných podvodů se vstupenkami, jelikož kapacita je jasně dána počtem míst k sezení a nelze ji snadno překročit.

Webová řešení prodeje vstupenek s rezervací míst se v posledních letech stávají stále více oblíbenými a využívanými v různých odvětvích, včetně zábavního průmyslu, sportu či cestování. Avšak s růstem vývojem v oblasti webových technologií je důležité sledovat a využívat nové trendy a technologie a přizpůsobovat jim takováto řešení, aby byla pro zákazníky stále atraktivní a relevantní. Tato práce se proto zaměřuje na vývoj frontendové části webové aplikace prodeje vstupenek s rezervací míst, která bude využívat moderní webové technologie a nástroje, které jsou v současné době nejvíce využívané a oblíbené.

Cíle práce

Cílem této práce je vyvinout prototyp responzivní webové aplikace nabízející prodej vstupenek s rezervací míst se zaměřením převážně na vývoj frontendové části. Výsledkem této práce bude webová aplikace vyvinuta moderními webovými nástroji a technologiemi, která umožní potenciálním zákazníkům zobrazit mapu areálu nějaké akce či kulturní události, vybrat si jedno či více preferovaných míst, přidat si vstupenky do nákupního košíku a vytvořit tak objednávku. Tato práce se bude zabývat vývojem takového webového řešení, ale pouze z pohledu frontendové části. Ostatní funkčnosti, jako například backendový systém či administrační řešení, nebudou součástí této práce.

Práce je strukturovaná do několika částí, počínající touto úvodní kapitolou, která stručně shrnula pozadí trhu online prodeje vstupenek a systémů rezervací míst. Následující kapitola poté takovéto online systémy na tamním trhu poskytovatelů zanalyzuje, a to převážně z pohledu jejich strategií, taktit, silných a slabých stránek v oblasti online prodeje vstupenek a systémů rezervací míst.

Dále se práce bude zabývat nejčastějšími technickými výzvami, které se mohou při vývoji takového systému vyskytnout. Opět půjde o technické výzvy převážně z pohledu frontendového řešení, jako například zajištění plné responzivity, implementace optimalizovaného plánu sedaček, zajištění dostupnosti dat v reálném čase, implementace bezpečného a spolehlivého systému rezervací míst a další. Tyto výzvy budou v podrobnosti vysvětleny a budou navržena možná řešení, která by mohla být použita při vývoji takového systému.

Praktická část práce se bude převážně zabývat vývojem frontendové části webové

aplikace umožňující nákup vstupenek s rezervací místa. V úvodní části bude vymezen rozsah funkčnosti aplikace a bude uveden podrobný popis hlavních funkcí, které bude aplikace nabízet. V této části budou také stručně uvedeny další funkčnosti, které ale nespadají do rámce této práce a bude vysvětleno proč byly záměrně vynechány. Důraz bude kláden na jasnou specifikaci a požadavky, které musí být splněny, aby bylo možné výsledný prototyp objektivně zhodnotit.

Následující kapitola bude pojednávat o grafickém návrhu uživatelského rozhraní aplikace. V této části budou také stručně shrnutý návrhové vzory a nejlepší praktiky při návrhu uživatelských rozhraní, v kontextu navrhovaného prototypu, které zajišťují efektivní a uživatelsky přívětivé zážitky. Cílem této části je vytvořit grafický návrh uživatelského rozhraní aplikace, který bude zároveň přehledný, intuitivní a přívětivý pro uživatele. A který bude zároveň použit v části implementace, jako předloha vyvíjené aplikace.

Hlavní kapitola praktické části se bude zabývat implementací frontendové části webové aplikace. V této části bude vytvořen prototyp webové aplikace, který bude splňovat všechny požadavky a specifikace, které byly v úvodní části práce vyspecifikovány. Dle analýzy těchto požadavaků budou nejprve vybrány vhodné nástroje a technologie, které budou při vývoji použity. Tyto nástroje a technologie budou také ve stručnosti představeny a bude vysvětlen důvod jejich výběru. Dále se bude tato kapitola zabývat postupem vývoje takovéto webové aplikace od začátku až do konce s důrazem na popis a implementaci hlavní komponenty zajišťující vykreslení interaktivního plánu sedaček v areálu. Tato kapitola bude zakončena technickou dokumentací jednotlivých komponent, které byly vytvořeny při vývoji aplikace, jejich komunikaci a využití.

V závěru bude výsledný prototyp vyhodnocen a porovnán s požadavky a specifikacemi, které byly v úvodní části práce vyspecifikovány. Dále budou také uvedena další možná vylepšení a rozšíření, která by mohla být v budoucnu v aplikaci implementována.

1 Specifikace prototypu

Praktická část pojednává o vývoji prototypu frontendu webové aplikace pro prodej vstupenek s důrazem na implementaci funkčnosti rezervace míst. Nutno podotknout že výsledný prototyp nebude ani není v plánu, aby byl plně funkční, nýbrž pouze ukazuje možnou implementaci konkrétních zvolených částí.

K implementaci prototypu je důležité předem vydefinovat jasnou specifikaci a požadavky na výsledný produkt. Bez těchto specifikací by nebylo možné finální výsledek objektivně zhodnotit. A právě tato kapitola se zabývá různými klíčovými funkcemi a součástmi webového řešení pro prodej vstupenek a zkoumá a podrobně popisuje každý aspekt, aby bylo možné jasně pochopit požadovaných výsledek. Tyto specifikace a požadavky později poslouží jako základ pro návrh a implementaci prototypu a následně i pro objektivní zhodnocení finálního výsledku.

1.1 Interaktivní mapa

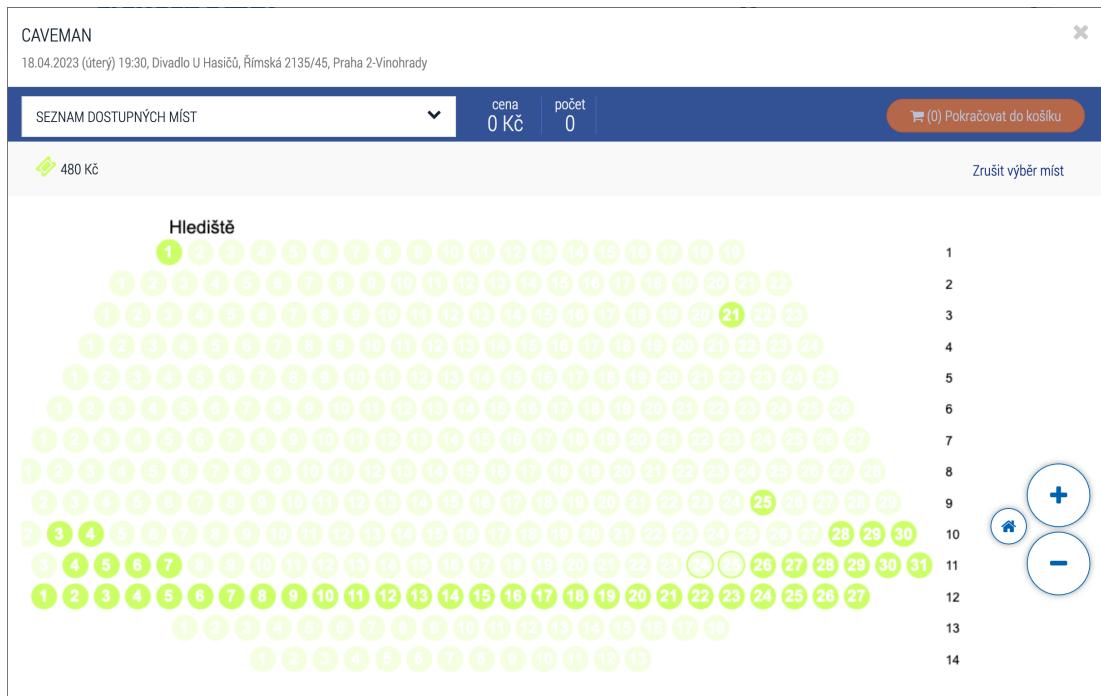
Vizualizace a celkové zobrazení interaktivní mapy sedaček místa konání akce je jednou z nejdůležitějších částí řešení webové aplikace na prodej vstupenek s rezervací míst. Tato podkapitola se zabývá nejhlavnějšími aspekty vizualizace mapy sedaček, jako je celkové rozvržení a struktura, barevné kódování prvků na mapě, ovládání mapy, zobrazení dostupných informací o zvoleném místě a také důležitost dostupnosti dat v reálném čase. V každé sekci budou tyto aspekty podrobněji rozebrány, ukázány příklady z reálného světa a vysvětlena jejich důležitost.

1.1.1 Rozložení a struktura

Pro docílení přehledného a uživatelsky přívětivého zobrazení plánu sedaček je třeba zajistit správnou vizuální strukturu a ideální rozložení prvků na obrazovce. Dobře uspořádané uživatelské rozhraní pomáhá zákazníkům lépe se v mapě ori-

entovat a vybrat tak preferovaná místa rychleji a snadněji.

Obrázek 1.1 zobrazuje příklad zasedacího pořádku portálu TICKETPORTAL v Divadle U Hasičů v Praze na Vinohradech. V tomto zobrazení nalezneme indikaci hlediště, dostupných i nedostupných míst k sezení a číslování řad sedaček. Toto zobrazení, ačkoliv poskytuje všechny důležité informace, není tolik uživatelsky přívětivé a atraktivní. Například výběr správně kontrastního barevného zobrazení sedaček by celkovému zobrazení velmi prospělo. Problematiku barevného kódování dále rozebírá následující sekce 1.1.2.



Obrázek 1.1: Plánek sedaček v síti TICKETPORTAL

Rozložení a struktura prvků na mapě by mely upřednostňovat uživatelskou přívětivost a snadnou navigaci, aby zákazníci mohli rychle najít a vybrat požadovaná místa. Aby bylo tohoto dosaženo, měla by vizualizace mapy zahrnovat zřetelné sekce, jasné popisky a intuitivní uspořádání míst k sezení či případně i místa vymezená pro stání.

Nejdůležitější prvky, které by mapa měla obsahovat jsou:

1. Místa k sezení - zvýrazněná místa k sezení, která jsou k dispozici pro výběr.
2. Místa pro stání - místa, která jsou určena pro stání, mají větší kapacitu než místa k sezení a jsou také k dispozici pro výběr.

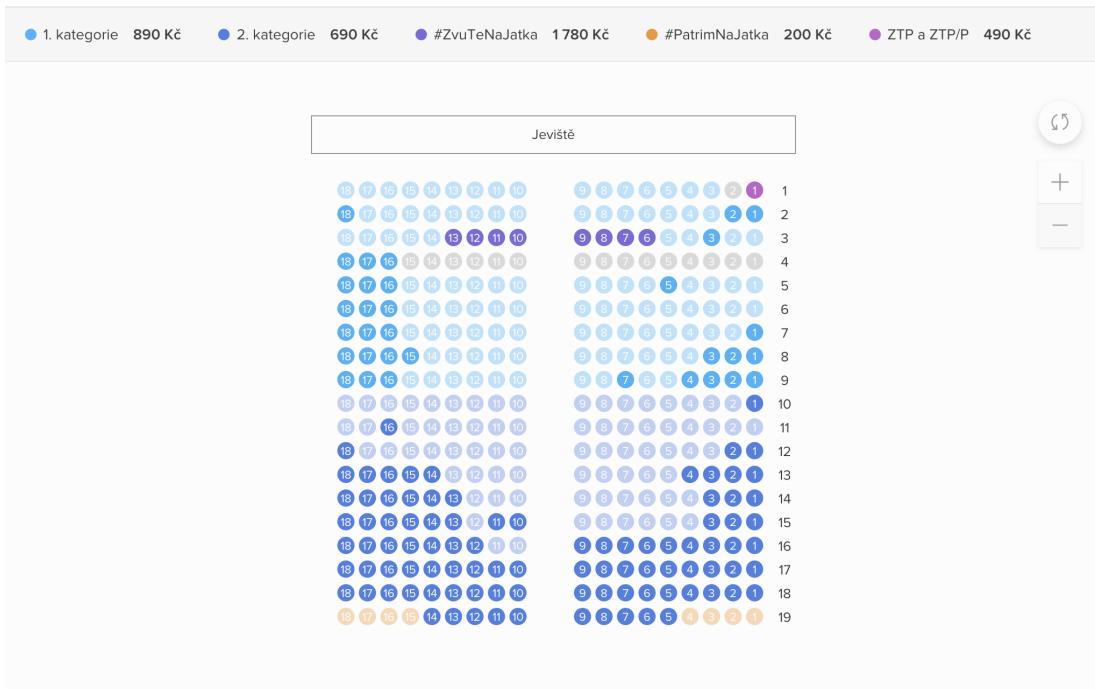
3. Sektry - rozdelení větších plánu se sedačkami do jednotlivých sektorů seskupujících místa k sezení či stání.
4. Hlediště - místo, kam budou hledět návštěvníci, důležité pro výběr místa s dobrým výhledem.
5. Popisky - popisky pro zvýraznění významných míst, jako třeba řady nebo názvy sekcí.
6. Značky pro zvýraznění dalších objektů - méně důležité, přesto informativní prvky na mapě jako třeba WC, bar, kavárna nebo zábradlí či sloupy.

Tyto prvky by měli být jasně vizuálně definované a na mapě přehledně rozložené. Záleží převážně o správnou definici a nakreslení prvků na mapě, což je úzce spojeno s administrací mapy a jejího nastavení. Tu totiž musí někdo prvně vydefinovat a uložit do nějakého datového formátu. Data si v tomto formátu poté vyžádá aplikace a na jejich základě mapu vykreslí uživateli v prohlížeči. Touto problematikou se následně více zabývá sekce ?? v implementační části práce.

1.1.2 Barevné kódování

Barevné kódování je dalším zásadním aspektem vizuálního zobrazení mapy místa konání, protože pomáhá uživatelům rychle identifikovat například kategorie míst, dostupnost a cenové úrovně sedadel. Díky použití odlišných barev pro různé typy sedadel se zákazník v mapě může snadněji a rychleji orientovat.

Obrázek 1.2 zobrazuje plánek míst sezení služby GoOut, který barevně odlišuje různé kategorie míst, dostupnost a cenové úrovně.



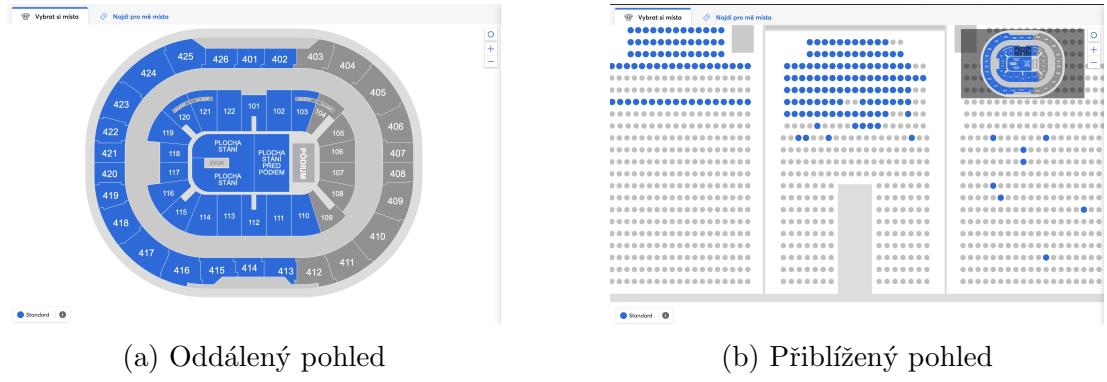
Obrázek 1.2: Mapa barevně odlišených sedadel GoOut

Efektivní implementace barevného kódování vyžaduje použití snadno rozlišitelných barev, které vyhovují zákazníkům s různými zrakovými schopnostmi. Zvolené barevné schéma by mělo zlepšit přehlednost a zvýšit uživatelský komfort tím, že zjednoduší proces výběru sedadel. Pro zachování jednotného a přehledného zobrazení je výhodné předem definovat paletu barev, které budou moci být později použity pro různé prvky na mapě. Užitím palety barev lze také jednodušeji udržet jednotnou vizuální identitu mapy a lze také předcházet zobrazenímu nekontrastních barevných kombinací, například textu na barevném podkladu.

1.1.3 Ovládání mapy

Ovládání zobrazení mapy je důležitou částí implementace, jelikož dodává větší interaktivnost tím, že umožňuje například pohyb kamery mapy po ose x a y , rotaci či přiblížení a oddálení. Tyto funkce zákazníkovi umožňují podrobněji prozkoumat celou mapu místa a zároveň zobrazit více informací na jednom zobrazení. Díky těmto funkcím získá zákazník také lepší přehled o celkovém rozpoložení míst v areálu. Takovéto funkce jsou esenciální u zobrazení map větších areálů jako jsou například arény či stadiony. Tyto mapy jsou totiž většinou ještě rozdeleny do sektorů, které seskupují místa k sezení či stání. Použitím funkce přiblížení a oddeálení lze také zobrazit různé úrovně detailu prvků na mapě v závislosti právě na hodnotě přiblížení.

Tento přístup například využívá síť *Ticketmaster*, který je možno vidět na obrázku 1.3, kde je demonstrována mapa sedadel s funkcí přiblížení a posunu a také s rozdelením na sektory. Při oddálení mohou uživatelé zobrazit celkové rozložení místa konání rozdělené do sektorů. Po přiblížení se zobrazí podrobnější zobrazení jednotlivých míst ve vybraném sektoru, což uživatelům umožňuje podrobně prozkoumat konkrétní oblasti areálu a vybrat požadované místo.



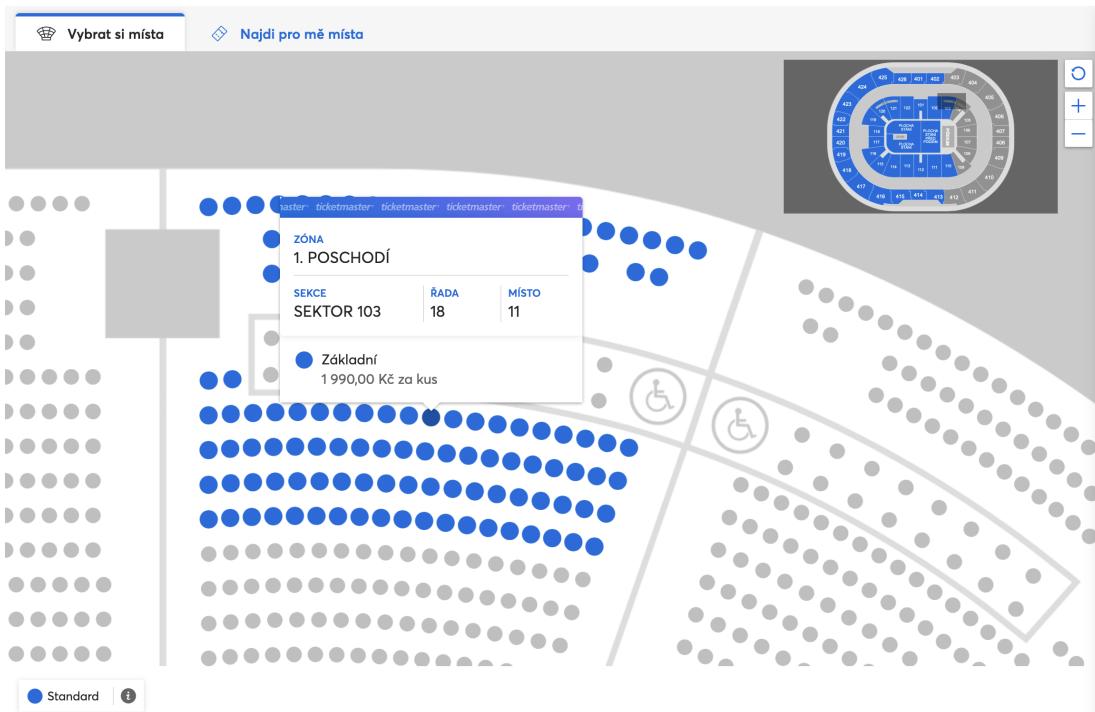
Obrázek 1.3: Různé pohledy mapy v síti Ticketmaster.

Při implementaci těchto funkčností je důležité myslit na přístupnost z pohledu koncových zařízení. Například pro nedotyková zařízení tyto funkčnosti zajišťují ukazovací zařízení jako počítačové myši. Na dotykových zařízeních jsou tyto funkčnosti docíleny pomocí dotykových gest *pinch* a *zoom* pro simulaci přiblížení a oddálení, gesta *rotate* pro rotaci a gesta *pan* pro pohyb na osách mapy. Je tedy třeba zajistit podporu pro dotyková i nedotyková zařízení. Často se k mapám zobrazuje i lišta s nástroji, které tyto funkčnosti ovládají. V těchto lištách je také vhodné umístit tlačítko na vrácení zobrazení mapy do výchozího zobrazení.

1.1.4 Stav a informace o sedadlech

Sedadla a ostatní místa k výběru o sobě nesou informace, které na mapě nemusejí být zřejmá a které zákazníkovi usnadňují výběr místa. Tyto informace mohou být pro zákazníka zásadní, jelikož poskytují podrobnosti, jako je dostupnost, cena, přístupnost či další jiné popisy. Díky těmto informacím se zákazník může lépe rozhodnout a vybrat si sedadlo, které nejlépe vyhovuje jeho preferencím a požadavkům.

Obrázek 1.4 zobrazuje mapu sedadel se stavem a informacemi o zvoleném sedadlu. Uživatelé si mohou zobrazit dostupnost, cenu a další důležité údaje o jednotlivých místech, což jim usnadňuje výběr preferovaných míst k sezení.



Obrázek 1.4: Informace o sedadle na mapě v síti Ticketmaster.

Aby bylo možné efektivně implementovat informace o stavu sedadel, měla by být mapa míst jasná a srozumitelná a všechny důležité údaje by měly být snadno dostupné. Jedním z běžných přístupů je použití vyskakovacích oken typu *popover* s informacemi, které se zobrazí například po najetí myší. Tato metoda však nemusí být vhodná pro mobilní a dotyková zařízení, kde takováto podobná gesta nefungují či fungují hodně omezeně. Pro řešení tohoto problému lze použít alternativní přístup, a to například zobrazení informací o sedadle ve vyjíždějící spodní či boční liště po označení na sedadla. Tím je zajištěno, že stav sedadla a informace o něm jsou přístupné uživatelům na všech zařízeních.

Toto okno či lišta by měla obsahovat ty nejdůležitější informace o zvoleném místě, jako:

1. Řada a místo - číselné či alfanumerické označení řady a místa sedačky.
2. Sektor - pokud je mapa rozdělena do sektorů, tak sektor, ve kterém se sedačka nachází.
3. Cenu - cenu vstupenky odpovídající dané sedačce.
4. Barevné označení - barevné označení sedačky, například dle její cenové kategorie.

5. Stav sedačky - stav, ve kterém se sedačka vůči zákazníkovi nachází (obsazená, zvolená, nedostupná, ...).
6. Další atributy - další informativní atributy, jako například informace o zhoršených podmínkách viditelnosti nebo vyhrazení místa pro osoby s postižením.

1.1.5 Data a jejich dostupnost

Pro zobrazení a fungování aplikace je nezbytné mapu a celý její stav zkonstruovat pomocí reálných a aktualizovaných dat dostupných z backendového systému pomocí aplikačního rozhraní, označovaného jako *API*. Tato data obsahují informace o vstupenkách a sedačkách jako například dostupnost, cena, umístění a další ostatní relevantní informace. Struktura těchto dat by měla být jasně definovaná a dostupná ve formátu vyhovujícímu užití aplikace.

Areály s velkým množstvím sedaček mohou být problematické a to převážně z pohledu objemu přenášených dat mezi klientem a API. Je důležité myslit na implementaci inteligentního datového přenosu, který zajistí komunikaci a přenos pouze nezbytných dat. Díky menšímu objemu přenášených dat se pak zdá aplikace rychlejší a responzivnější.

Dalším aspektem práce s daty v takovéto aplikaci je aktualizace dat o dostupnosti. Tato funkčnost zajišťuje zákazníkovi zobrazení aktuálních informací o sedačkách či vstupenkách a snižuje tak například riziko konfliktu výběru již obsazené sedačky. Průběžné aktualizace dat lze docílit technologiemi jako například *WebSockets* nebo částečnými aktualizacemi dotazovanými skrze API. Tyto metody budou později rozebrány v implementační části práce.

1.2 Nákupní košík

Velmi důležitou součástí fungování celého systému je efektivní správa a vizualizace nákupního košíku. Ta například zákazníkovi poskytuje přehledný souhrn položek, které si objednává, a umožňuje mu je případně upravit či odebrat. Nicméně ale tím nejdůležitějším aspektem nákupního košíku jsou jaká data jsou v něm uchovány a jakým způsobem jsou zpracována.

Tato podkapitola se zabývá popisem hlavních funkčností a požadavků na nákupní košík, které jsou nezbytné pro jeho efektivní fungování v rámci webového řešení

s využitím rezervace sedadel.

The screenshot shows a shopping cart interface. At the top right, it says "Čas na dokončení nákupu 12:00". Below that, there's a circular icon with the number "1" and the word "KOŠÍK" next to it. The main content area displays a single item: "LED ZEPPELIN SYMPHONIC". Below the title, it says "Pořadatel: JVS GROUP s.r.o., IČO 25865005, Slavíkova 1742/2a, 70800, Ostrava-Poruba". A table shows the details of the selected ticket:

Sektor	Řada	Sedadlo	Sleva	Cena
Balkon vpravo / VSTUP 4. p.	17	62	-	1 290,00 Kč
Balkon vpravo / VSTUP 4. p.	17	63	-	1 290,00 Kč

Total: 2 ks / 2 580,00 Kč. There is a blue button labeled "+ Přidat další".

Obrázek 1.5: Obsah nákupního košíku na portálu Ticketportal.cz

1.2.1 Správa dat

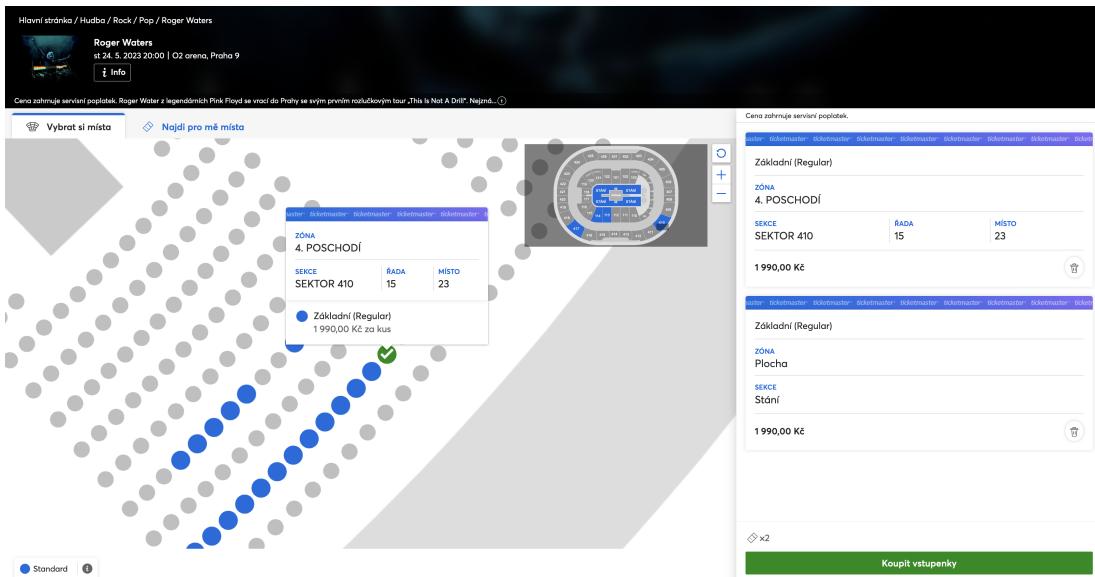
Nákupní košík by z datové perspektivy měl být implementován jako objekt, který uchovává informace o všech vybraných sedačkách a vstupenkách. Datová struktura a uchovávaná data uvnitř daného objektu o stavu nákupního košíku by měla být jasně definovaná a dostupná ve formátu vyhovujícímu užití aplikace. Díky těmto datům by mělo být vždy možné zrekonstruovat stav nákupního košíku a to i v případě, že uživatel opustí stránku nebo zavře prohlížeč.

Tuto problematikou se bude primárně zabývat kapitola 3.5, ve které bude podrobně popsána datová struktura a finální funkčnost nákupního košíku.

1.2.2 Přehled obsahu košíku

Pro zákazníka nákupní košík slouží primárně k přehlednému zobrazení všech vybraných položek, které budou tvořit jeho objednávku. V případně této webové aplikace se primárně jedná o vstupenky a zarezervované sedačky.

Zákazník by měl být schopen jednoduše zjistit, jaké položky si objednává a jaké mají parametry. Seznam všech těchto položek posléze tvoří celkový přehled nákupního košíku, který umožňuje zákazníkovi zkontolovat jeho výběr a provést případné změny před dokončením objednávky.



Obrázek 1.6: Nákupní košík na portálu Ticketmaster.com

Nákupní košík by zároveň měl zobrazit přehled cen jednotlivých položek a celkovou cenu objednávky včetně všech daní a dalších případních poplatků. Tato funkcionality zákazníkovi zajišťuje naprostou transparentnost v nabízených službách a umožňuje mu předem zkontrolovat celkovou cenu objednávky.

The screenshot displays the final order summary. On the left, a summary box says 'Vybrané vstupenky pro vás nyní držíme. Pokračujte v objednávce.' (Selected tickets are now held for you. Continue with the order.) Below it, the 'Vaše vstupenky' (Your tickets) section lists two items: 'Základní (Regular)' and 'Základní (Regular)' with detailed information like 'Plocha', 'Stání', '4. POSCHODÍ', 'SEKTOR 410', 'Řada 15', and 'Místo 23'. It also includes a note about re-entering if tickets are not available. A blue 'Pokračovat' (Continue) button is at the bottom. On the right, a large summary table titled 'Vaše objednávka' (Your order) shows the breakdown: 'Vstupenky 2' (2 tickets), 'Roger Waters' (3980,00 CZK), 'Středa, 24.05.2023 20:00', 'Sekce Plocha-Stání' (Section Floor-Standing), 'Informace Stání' (Information Standing), and a detailed breakdown of each ticket's type, category, and price. The total 'CELKEM K PLATBĚ' (Total to pay) is listed as '3995,00 CZK'.

Obrázek 1.7: Souhrn objednávky na portálu Ticketmaster.com

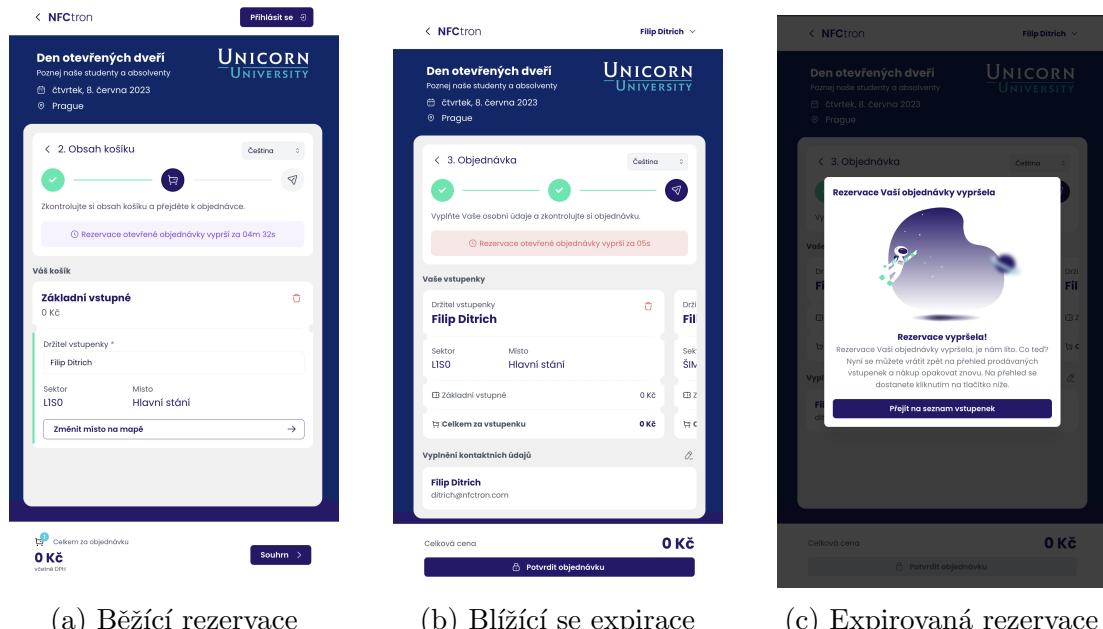
1.2.3 Rezervace míst

V rámci nákupního procesu si zákazník vybírá místa, které jsou dostupná a při jejich výběru se zákazníkovi přidají do nákupního košíku. Tato vybraná místa zákazníkem by se měla rezervovat, aby se předešlo nepříjemným situacím při dokončení objednávky, kdy zákazník zjistí, že jeho vybrané místo je již obsazené.

V případně aplikace, která se primárně zaměřuje na rezervaci míst při prodeji vstupenek je tato funkcionality naprostě esenciální. Zákazník by měl mít možnost vybrat si místa, která jsou v danou chvíli dostupná a při jejich výběru by se měla pomocí rezervačního mechanismu v rámci vytvářené objednávky zákazníkovi zarezervovat a garantovat mu v případně dokončení objednávky jejich dostupnost.

Tento rezervační systém by měl dále implementovat funkčnost časového omezení rezervace, která zajistí uvolnění rezervovaných míst po uplynutí určitého časového limitu. Tato funkčnost je velmi důležitá pro zajištění dostupnosti míst pro všechny zákazníky a zabrání zarezervování sedadel, která by nebyla zakoupena například z důvodu opuštění nákupního procesu zákazníkem.

V rámci tohoto rezervačního mechanismu by také mělo být zákazníkovi jasně zobrazeno, že má určitý časový limit, ve rámci kterého musí svou objednávku dokončit, jinak se jeho rezervace uvolní a bude si muset místa vybrat znova.



Obrázek 1.8: Rezervační mechanismus služby NFCtron Tickets na zkušební akci

1.3 Dokončení objednávky

Jakmile si zákazník vybere požadované vstupenky a rezervovaná místa, je jeho posledním úkolem dokončit a tím tak vytvořit objednávku, kterou následně zaplatí. Tento proces je většinou rozdělen do několika kroků, které zákazníka postupně provedou celým procesem dokončení objednávky.

Tato podkapitola se bude těmito kroky zabývat a bude se snažit popsat jejich jednotlivé části a funkčnosti, které by měly být v takovémto systému implementovány.

Avšak je nutné nejprve uvést, že následující funkčnosti jsou v práci uvedené pouze z důvodu kompletnosti představení celého procesu nákupu vstupenek s rezervací míst a záměrně nebudou v rámci praktické části plně implementovány nýbrž pouze orientačně využity – a to z důvodu toho, že se jedná již převážně o funkčnosti a procesy na straně backendového systému a ne frontendového, který je předmětem této práce.

1.3.1 Osobní informace

Pro účely dokončení objednávky je nutné, aby si zákazník vyplnil své osobní informace, které jsou potřebné pro vytvoření objednávky a následné doručení vstupenek. Tyto informace by měly obsahovat minimálně následující údaje:

- Jméno a příjmení
- Emailová adresa
- Telefonní číslo

Tyto informace jsou nezbytné například pro odeslání potvrzení objednávky, doručení vstupenek či poskytování případné podpory zákazníkům.

Někteří poskytovatelé stále nabízí doručení fyzických vstupenek na adresu zákazníka, nicméně tento způsob je spíše přežitek. V dnešní moderní době je výhodnější i ekologičtější využít možnost digitálních vstupenek, které zákazník obdrží v elektronické podobě například prostřednictvím emailu či SMS zprávy.

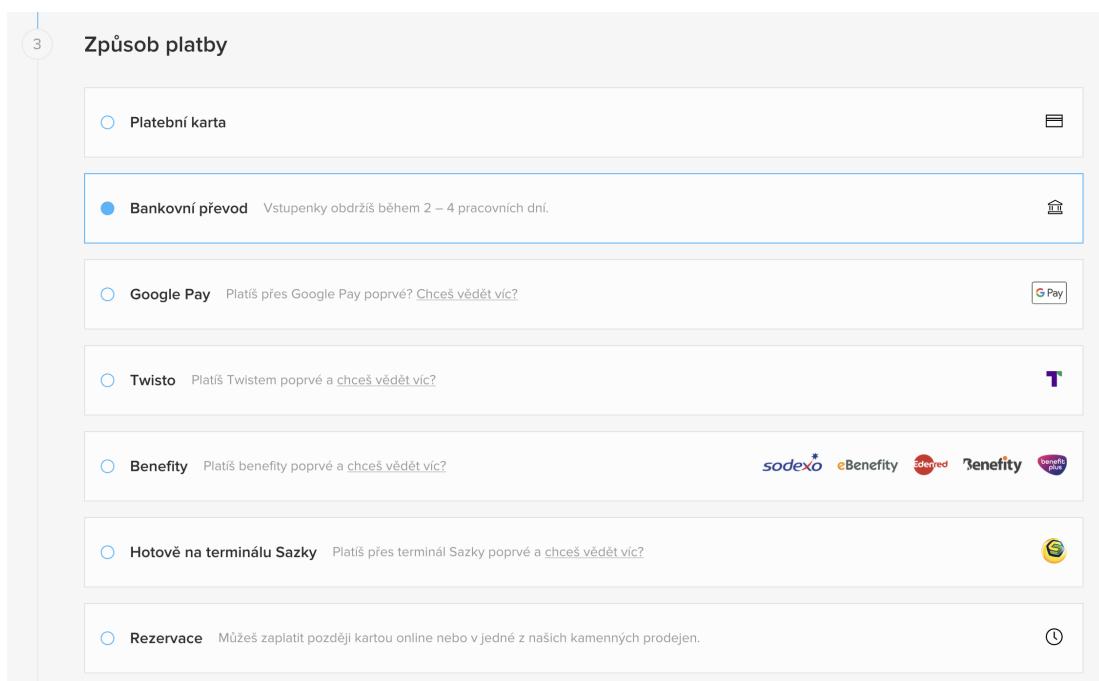
V rámci dokončení objednávky je také časté vybídnutí zákazníka k dokončení registrace na dané platformě či službě, přes kterou vstupenky nakupuje. Tato

registrace je většinou dobrovolná, ale může být i povinná, pokud chce zákazník využít některé z výhod, které jsou s touto registrací spojeny.

1.3.2 Výběr platební metody

Proces dokončení objednávky by v posledním kroku měl zákazníkovi nabídnout několik možností platby. Nejčastěji se jedná o platební metody jako:

- platba kartou
- platba přes PayPal
- platba přes Apple Pay
- platba přes Google Pay
- platba přes bankovní převod



Obrázek 1.9: Vyběr platební metody na GoOut.net

Na obrázku 1.9 je vidět výběr platební metody ve službě GoOut.net, která nabízí výše zmíněné platební metody¹ spolu i dalšími jako například platba přes službu Twisto či Sodexo a další jiné benefity.

¹platební metoda Apple Pay zde není uvedena, jelikož je dostupná pouze v rámci prohlížečů Safari: <https://support.apple.com/guide/iphone/use-apple-pay-in-apps-app-clips-and-safari-iph67e89f7c8/ios>

Pro účely platby kartou je nutné, aby byl zákazník přesměrován na platební bránu, která je schopna tuto platbu zpracovat. Většinou se jedná o platební brány třetích stran, které jsou schopny zpracovat platby z různých platebních karet a poskytovatelů. Mezi nejznámější poskytovatele platebních brán na českém trhu patří například:

- *Comgate* (<https://www.comgate.cz/platebni-brana>)
- *GoPay* (<https://www.gopay.com/cs/>)
- *PayU* (<https://czech.payu.com/payu-moderni-platebni-reseni/>)
- *GP webpay* (<https://www.gpwebpay.cz/>)
- *Stripe* (<https://stripe.com/en-cz>)

Na obrázcích 1.10 a 1.11 je vidět, že každý poskytovatel platební brány nabízí jiné výhody a poplatkové modely, nicméně všechny mají společné to, že zpracovávají platby z různých platebních karet a poskytovatelů a přímo podporují i mobilní platby přes Apple Pay a Google Pay.

Sazby a poplatky

	Comgate tarif Easy	Comgate tarif Profi	GoPay	PayU	Pays	Global Payments	Stripe	VivaWallet
Režim sazebníku	pevné sazby	pevné sazby	MIF++	pevné sazby	pevné sazby	MIF++	pevné sazby	MIF++
Poplatek z platby kartou (spotř., EU)	0,9 %	0,65 %	0,9 až 2,26 %	1,45 %	1,5 až 2,5 %	1,29 %	1,4 %	2,36 až 3,01 %
Poplatek z transakcí, Kč	1	0,7	3	1	1	0,35	6,50	10
Měsíční poplatek, Kč	0 až 100	0 až 200	0 až 190	-	-	0 až 395	-	-
Aktivační poplatek, Kč	-	-	-	999	600	1199	-	-
Převod na bankovní účet, Kč	-	-	10	-	0 až 39	3	-	15
Sazebník na webu	✓	✓	✓	✓	✓		✓	✓

Obrázek 1.10: Srovnání platebních bran dle Comgate.cz [1]

Platební metody

	Comgate	GoPay	PayU	Pays	Global Payments	Stripe	VivaWallet
Platby kartou	✓	✓	✓	✓	✓	✓	✓
Apple Pay	✓	✓	✓	✓	✓	✓	✓
Google Pay	✓	✓	✓	✓	✓	✓	✓
Bankovní převody v Česku	✓	✓	✓	-	-	-	-
Bankovní převody na Slovensku	✓	✓	✓	-	-	-	-
Bankovní převody v Polsku	✓	✓	✓	-	-	-	-
QR platby	✓	-	-	✓	-	-	-
Odložené platby Twisto	✓	-	✓	-	-	-	-
Odložené platby Skip Pay	✓	-	-	-	-	-	-
Odložené platby PlatímPak	✓	-	-	-	-	-	-
Platby na třetiny Twisto	✓	-	-	-	-	-	-
Platby na třetiny Skip Pay	✓	-	-	-	-	-	-
Splátkový prodej Cofidis	✓	-	-	-	-	-	-

Obrázek 1.11: Platební metody poskytované platebními poskytovateli dle Comgate.cz [1]

Některé e-shopy však nabízí i vlastní platební bránu, nicméně tato možnost je spíše výjimkou. Stát se provozovatelem vlastní platební brány je totiž velmi náročné na vývoj, udržbu a provoz a je k němu nutné mít i licenci platební instituce (PI) [2], kterou vydává Česká národní banka (ČNB) [3].

Platby přes bankovní převod mohou být pro poskytovatele řešení prodeje vstupenek velmi výhodné, jelikož se jedná o levnější způsob převodu peněz, než je tomu u plateb kartou. Platební brány si totiž účtují poplatky za zpracování každé z plateb a to nejčastěji v režimu pevného sazebníku či MIF++ modelu², který rozděluje poplatky mezi vydavatelskou bankou, karetním schématem (např. Visa, MasterCard) a platební bránou.

Do nedávna byla platba bankovním převodem pro zákazníky stále méně oblíbená, jelikož bylo nutné počkat na připsání peněz na účet poskytovatele, což mohlo trvat i několik dní. V dnešní době je však možné využít tzv. instantní platby, které jsou schopny převést peníze mezi účty různých bank během několika sekund. V České republice aktuálně podporuje instantní platby již několik bank, jako například Komerční banka, a.s., MONETA Money Bank, a.s. či Fio banka, a.s. a další³.

Pro účely této práce, jak bylo již zmíněno na začátku podsekce 1.3, bude backen-

²někdy také označován jako Interchange++

³kompletní seznam účastníků okamžitých plateb dle ČNB dostupný z: https://www.cnb.cz/export/sites/cnb/cs/platebni-styk/.galleries/certis/download/seznam_okamzite_platby.pdf

dový systém celou funkčnost plateb pouze simuloval a nebude se snažit o integraci s žádnou platební bránou.

1.3.3 Souhrn objednávky

Po vyplnění všech potřebných informací a výběru platební metody by zákazníkovi měl být před finálním potvrzením objednávky zobrazen její souhrn. Tento souhrn zákazníkovi umožní zkонтrolovat, zda jsou všechny údaje správně vyplněné a zda je vše v pořádku. Pokud zákazník zjistí, že je něco špatně, měl by mít možnost vrátit se zpět a údaje opravit. Pokud je vše v pořádku, měl by mít možnost objednávku potvrdit a přejít k platebnímu procesu.

Součástí potrvzení objednávky také často bývá zaškrtnutí souhlasu se zpracováním osobních údajů a obchodních podmínek či i případně dalších informací, jako například zasílání novinek a reklamních nabídek na e-mailovou adresu zákazníka.

1.3.4 Vytvoření objednávky a potvrzení

Po potvrzení objednávky by měla být vytvořena objednávka v databázi a zákazníkovi by mělo být zobrazeno potvrzení o jejím vytvoření. Na základě vybrané platbní metody by měl být dále přesměrován k jejímu zaplacení a posléze zpět na detail objednávky s potvrzením o zaplacení. Při úspěšné platbě by zákazníkovi měl systém doručit zakoupené vstupenky v elektronické podobě, například v podobě PDF souboru, který bude obsahovat vstupenky ve formátu QR kódu.

2 Návrh uživatelského rozhraní

Ve světě digitálních produktů a jejich designu jsou uživatelské rozhraní, z anglického *User Interface (UI)*, a uživatelský zážitek, z anglického *User Experience (UX)*, dva pojmy, které se často zaměňují, ačkoli se jedná o velmi odlišné aspekty procesu vývoje produktu[4]. Tato kapitola si klade za cíl představit koncepty UI a UX, prozkoumat jejich vzájemný vztah a zabývat se specifiky návrhu UI pro zkoumanou a vyvájenou aplikaci prodeje vstupenek s rezervací míst.

UI se vztahuje k vizuálním prvkům produktu, se kterými uživatel interaguje – tedy tlačítkům, textu, ikonografii, formulářům a všem vizuálním prvkům, které umožňují uživateli interagovat s produktem[4]. V kontextu aplikace pro prodej vstupenek s rezervací míst se UI vztahuje například k interaktivnímu plánu sedaček, výběru vstupenek, tlačítku pro přechod k dokončení objednávky nebo nákupnímu košíku.

UX je na druhou stranu celkový zážitek uživatele při interakci s produktem. Je ovlivněn snadností použití, hodnotou, kterou uživatel z produktu získává, a emocemi, které jsou při interakci vyvolány. UX bere v potaz celou cestu uživatele, od okamžiku, kdy uživatel do aplikace vstoupí, až po okamžik, kdy dokončí nákup[4].

Souhra UI a UX je v procesu návrhu produktu klíčová. Dobře navržené UI usnadňuje UX. Například intuitivně navržený plán sedadel (UI) může proces výběru sedadla zpříjemnit a zjednodušit (UX).

Následující sekce této kapitoly prozkoumají základní principy návrhu UI a možná použití Maslowovy hierarchie, za účelem návrhu rozhraní více zaměřeného na uživatele. Dále budou uvedeny a porovnány různé nástroje, které jsou k dispozici pro návrh UI a důvody rozhodnutí pro konkrétní nástroj. Následně budou analyzovány poznatky z kapitoly 1 z hlediska UI/UX se zaměřením na tzv. uživatelské příběhy, které tvoří základ UX designu.

Závěr této kapitoly bude věnován návrhu uživatelského rozhraní z předem defi-

novaných uživatelských příběhů a návrhu interaktivního prototypu aplikace.

2.1 Principy návrhu uživatelského rozhraní

Návrh uživatelského rozhraní je poměrně rozsáhlá disciplína, která se zaměřuje na vizuální a interaktivní aspekty produktu. Při návrhu UI je důležité dodržovat určité principy, které zajišťují optimální uživatelskou zkušenosť. Tato sekce shrnuje některé základní principy návrhu UI a posuzuje jejich implikace v kontextu aplikace pro prodej vstupenek s rezervací míst.

Jednoduchost

Návrh UI by měl směřovat k jednoduchosti. Čím méně úsilí musí uživatel vy-naložit na pochopení rozhraní, tím lepší bude celková uživatelská zkušenosť. Čisté, jednoduché rozhraní s jasným zaměřením na funkčnost snižuje kognitivní zátěž a zvyšuje použitelnost[5].

Konzistence

Tento princip prosazuje zachování jednotnosti napříč všemi prvky UI. Konzistence se projevuje v použití podobných prvků, akcí a designu napříč celým rozhraním[5]. Například pokud určitá barva značí interaktivní prvek na plánu sedadel, stejná barva by měla být použita i pro značení interaktivních prvků jinde v rámci aplikace. Tímto se zvyšuje předvídatelnost, což uživatelům usnadňuje orientaci a navigaci v rozhraní.

Pocit kontroly

Základním principem návrhu UI je umožnit uživateli cítit se vždy v kontrole nad produktem. Toho lze dosáhnout návrhem transparentního a intuitivního systému, ve kterém uživatel vždy ví, kde se nachází a jak postupovat[5]. V kontextu aplikace pro prodej vstupenek to může znamenat poskytnutí jasného a zřejmého způsobu, jak uživatelé mohou přejít k výběru sedadla, přidání do košíku a dokončení objednávky.

Zpětná vazba

Zpětná vazba je klíčovým aspektem každé interakce, protože potvrzuje nebo informuje uživatele o vykonaných akcích[5]. Vizuální indikátory, jako je zvýraznění vybraného sedadla nebo potvrzovací zpráva při přidání vstupenky do košíku, poskytují uživateli okamžitou zpětnou vazbu. Tím se snižuje nejistota a zvyšuje se důvěra uživatele v rozhraní.

Prevence a řešení chyb

Chyby jsou nevyhnutelné v jakékoli interakci, ale dobré navržené UI může zabránit většině uživatelských chyb nebo zjednodušit jejich řešení[5]. To může znamenat

například zakázání tlačítka *Pokračovat* dokud není vybráno sedadlo nebo zobrazení jasných a užitečných chybových zpráv, když něco selže.

Afordance a signifikance

Afordance se vztahuje k vlastnosti objektu, která naznačuje, jak se má používat. *Signifikance* jsou vizuálními nápodědami k témuž *afordancím*[6]. Například sedadlo na plánu sedadel může být navrženo tak, aby naznačovalo, že na něj lze kliknout (*afordance*), a změna kurzoru při najetí na sedadlo (*signifikance*) může tuto zprávu posílit.

Pochopení a aplikace těchto základních principů návrhu UI je klíčové pro vytvoření intuitivního a uživatelsky přívětivého rozhraní. Tyto principy řídí řadu rozhodnutí v rámci návrhu a pomáhají návrhu UI s celkovým cílem poskytnout uživatelům bezproblémový zážitek z rezervace vstupenek. Další sekce se zabývá tím, jak lze hierarchii Maslowa aplikovat pro další zlepšení návrhu zaměřeného na uživatele.

2.2 Aplikovaná psychologie na UI/UX

Some people say, “Give the customers what they want. “ But that’s not my approach. Our job is to figure out what they’re going to want before they do.

– Steve Jobs

Citát od Steva Jobse, zakladatele společnosti Apple, je výstižným popisem toho, co je cílem návrhu UI. Návrháři UI se snaží vytvořit takové rozhraní produktu, které bude uživatelům vyhovovat a bude je bavit používat. Aby tohoto mohlo být docíleno, je nejprve nutné aby produkt splňoval základní požadavky, jako je například funkčnost, stabilita a bezpečnost – musí splňovat nějakou minimální potřebu, jinak na ničem dalším nebude záležet[7].

Velmi důležitým aspektem návrhu uživatelského rozhraní produktu je psychologie koncového uživatele a jeho pochopení, pro které lze využít různé psychologické teorie a koncepty jako například *Maslowova hierarchie potřeb*. Tato hierarchie, obvykle vizualizovaná jako pyramidová struktura, ilustruje cestu jednotlivce k seberealizaci a naplnění, začínající od základních fyziologických potřeb až po složitější emoční a psychologické potřeby[8].

2.2.1 Maslowova hierarchie potřeb

Maslowova hierarchie potřeb je teorie psychologa Abrahama Maslowa, která se snaží definovat lidské potřeby, jejich hierarchii a vliv na lidské chování. Maslow tvrdil, že lidé mají potřeby, které se snaží uspokojit, ale některé z nich jsou naléhavější než jiné. Když jsou tyto potřeby uspokojeny, lidé se mohou cítit šťastnější, ale když nejsou, lidé mohou být frustrovaní a nespokojení.[8] Maslow rozdělil lidské potřeby do pěti základních úrovní, které jsou znázorněny na obrázku 2.1 níže.



Obrázek 2.1: Maslowova hierarchie potřeb[9]

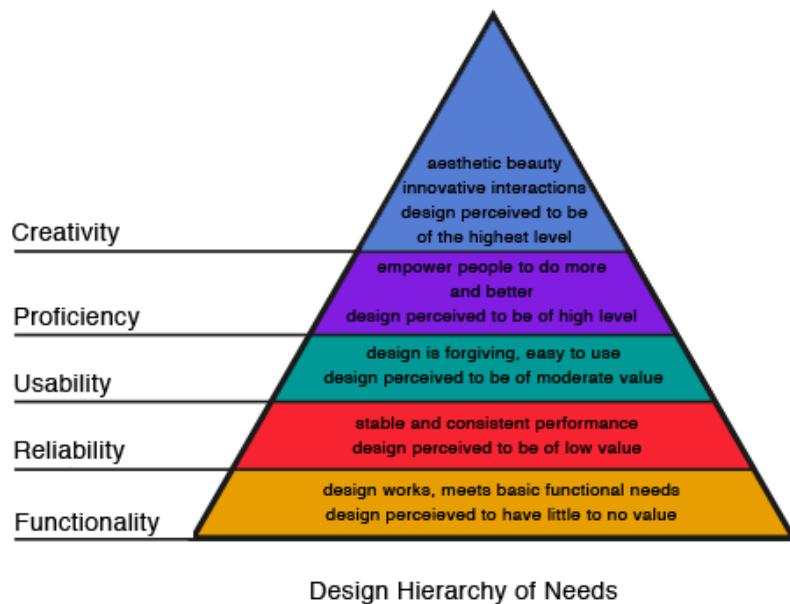
- Fyziologické potřeby:** základní potřeby pro přežití, jako je potrava, voda, teplo a spánek
- Potřeby bezpečí:** potřeby, které se týkají bezpečnosti a zabezpečení
- Sociální potřeby:** potřeby, které se týkají příslušnosti, lásky a přátelství
- Potřeby uznání:** potřeby, které se týkají úcty a sebeúcty
- Potřeby seberealizace:** potřeby, které se týkají osobního růstu a rozvoje

Jak to tedy ale souvisí s návrhem UI a zejména s návrhem aplikace pro prodej vstupenek?

2.2.2 Hierarchie potřeb v návrhu uživatelského rozhraní

Maslowova hierarchie potřeb může být aplikována na návrh UI tak, že každá úroveň hierarchie představuje jeden základní aspekt návrhu UI.

V roce 2010 navrhl Steven Bradley v článku *Designing For A Hierarchy Of Needs* podobnou hierarchii specificky pro design, se pěti odpovídajícími úrovněmi znázorněnými na obrázku 2.2.[7]



Obrázek 2.2: Hierarchie potřeb v návrhu UI dle Stevena Bradleyho[7]

Funkčnost

Na základě pyramidy jsou základní fyziologické potřeby. V kontextu návrhu UI to znamená základní funkčnost. Aplikace musí fungovat tak, jak se očekává, aby si uživatelé mohli vybrat sedadlo, přidat vstupenku do košíku a dokončit proces objednávky bez jakýchkoli problémů. Základní funkčnost musí být spolehlivá a robustní.

Spolehlivost

Další úroveň pyramidy je bezpečnost, která se v návrhu UI týká spolehlivosti. Rozhraní by mělo být navrženo tak, aby se uživatelé cítili bezpečně a sebevědomě při interakci s ním. Poskytování jasných pokynů, okamžité zpětné vazby a potvrzení o úspěšných akcích (například přidání vstupenky do košíku) přispívá k pocitu bezpečí a použitelnosti.

Použitelnost

Střední část pyramidy pokrývá sociální potřeby, které se v UI termínech rovnají uživatelské spokojenosti. Esteticky příjemné rozhraní, personalizovaný uživatelský zážitek a interaktivní prvky (jako interaktivní plán sedadel) mohou významně zvýšit uživatelskou spokojenost.

Odbornost

Potřeby sebeúcty zahrnují touhu po uznání a respektu. V kontextu aplikace pro prodej vstupenek by to mohlo znamenat přidání funkcí, které překračují očekávání uživatelů a zpríjemňují jim zážitek. Může se jednat o něco tak jednoduchého, jako je blahopřání po úspěšném nákupu, nebo vizuální animace při výběru sedadla.

Kreativita

Na vrcholu pyramidy se nachází seberealizace, která se týká realizace osobního potenciálu a hledání osobního růstu a vrcholných zážitků. Uživatelské rozhraní by mohlo přispět k této potřebě tím, že uživatelům umožní kreativně řešit problémy a dosáhnout svých cílů. Například nabízení návrhů na nejlepší dostupná sedadla nebo podobných akcí může uživatele posílit a zlepšit jejich zážitek.

Použití Maslowovy hierarchie pro návrh UI aplikace pro prodej vstupenek může pomoci zajistit, aby návrh splňoval potřeby uživatelů na různých úrovních. Z počátku je nutné zajistit základní funkčnosti a spolehlivost, aby uživatelé mohli využívat aplikaci bez jakýchkoli problémů. Dále je nutné zaměřit se na použitelnost, aby byl proces výběru sedadla a nákupu vstupenky co nejvíce zjednodušen. Při postupu v hierarchii se budou zkoumat různé metody, jak zvýšit uživatelskou spokojenosť a zlepšit jejich zážitek. Cílem na vrcholu tohoto procesu je navrhnout rozhraní, které vyvažuje praktičnost a uživatelskou přívětivost, zatímco zároveň zajišťuje vizuální přitažlivost a emoční zapojení, což povede k přínosnějšímu, uspokojivějšímu a úspěšnějšímu uživatelskému zážitku.

V předchozích částech byly prozkoumány základní principy návrhu uživatelského rozhraní (UI). Významná část tohoto průzkumu se soustředila na to, jak lze Maslowovu hierarchii potřeb použít v návrhu uživatelského rozhraní, což slouží jako cenný rámcem pro vytváření intuitivní aplikace zaměřené na uživatele. Zvolený způsob designu, zaměřený na fyziologické potřeby uživatele až po seberealizaci, zajišťuje, že aplikace slouží nejen svému funkčnímu účelu, ale také vytváří pro uživatele poutavý a uspokojující zážitek.

Kromě návrhu esteticky příjemného rozhraní je důležité vytvořit systém, který skutečně vyhovuje potřebám uživatelů. Konečným cílem je poskytnout uživatelům intuitivní, bezproblémový a příjemný zážitek při navigaci v aplikaci, výběru sedadel a nákupu vstupenek. Aby tohoto mohlo být dosaženo, se následující kapitola ponoří do efektivního nástroje v designu UI/UX známého jako *Uživatelské příběhy*.

2.3 Uživatelské příběhy

Při navrhování uživatelského rozhraní nejde pouze o estetiku nebo funkčnost; je vyžadováno pochopení potřeb a očekávání uživatele. Zahrnuje vytváření cesty, která uživatele bezproblémově provede aplikací a zároveň zajistí, aby mohli své úkony vykonávat efektivně a s potěšením. Technika, která se často používá v návrhu UI/UX k dosažení tohoto cíle, se nazývá *User Stories* (uživatelské příběhy).

Uživatelské příběhy jsou stručné, přímočaré popisy funkce nebo funkcionality, vyprávěné z pohledu uživatele. Slouží jako nástroj, který pomáhá udržovat návrh zaměřený na uživatele a zajišťuje, že konečný produkt efektivně splňuje jeho potřeby. Tyto příběhy kladou důraz na to, čeho uživatelé chtějí dosáhnout, podporují empatii a podporují návrhový proces, který se zaměřuje na uživatele[10]. Porozumění roli uživatelských příběhů při návrhu UI webového řešení pro prodej vstupenek je klíčové pro efektivní splnění potřeb koncových uživatelů.

2.3.1 Co jsou uživatelské příběhy

Uživatelské příběhy jsou součástí agilních vývojových praktik, široce používaných v návrhu UI/UX k zachycení zjednodušených popisů potenciálních funkcí aplikace z pohledu koncových uživatelů. Slouží jako rychlý a jednoduchý způsob, jak popsat uživatele, co chtějí a proč to chtějí[10]. Každá *User Story* následuje strukturovaný formát:

Formát uživatelského příběhu

“**Jako** [typ uživatele] **chci** [vykonat nějakou akci], **abych** [dosáhl nějakého cíle].“

V tomto formátu:

- **Typ uživatele** pomáhá definovat roli uživatele, který bude používat danou funkcionality.
- **Vykonat nějakou akci** umožňuje zjistit, co chce uživatel pomocí dané funkcionality udělat nebo čeho chce dosáhnout.
- **Dosáhl nějakého cíle** vysvětluje základní motivaci nebo hodnotu, kterou uživatel získá provedením akce.

Tento formát je velmi užitečný při vytváření uživatelských příběhů, jelikož pomáhá udržovat stručnost, jednoznačnost a zároveň poskytuje dostatek informací, aby bylo možné pochopit, co uživatel chce a proč to chce.

Uživatelské příběhy hrají také klíčovou roli při definování akceptačních kritérií, která dále podrobně popisují, jak by měla určitá funkce fungovat z pohledu uživatele. To pomáhá stanovit jasnou představu o účelu a očekávaném chování funkce, čímž usměrňuje její vývoj a testování[10].

V kontextu navrhovaného uživatelského rozhraní pro webové řešení prodeje vstupenek mohou tyto uživatelské příběhy pomoci přesně určit funkcionality, které jsou pro uživatele nejdůležitější. Pomáhají porozumět potencionálním uživatelům – návštěvníkům událostí, jejich potřebám (jako jsou např. prohlížení místa konání, výběr sedadel), jejich akce (přidání vstupenek do košíku, přechod k zaplacení) a jejich motivaci (užít si bezproblémový nákup vstupenek).

Následující sekce se zabývá tím, jak lze uživatelské příběhy konkrétně použít k navrhování uživatelského rozhraní.

2.3.2 Psaní efektivních uživatelských příběhů

Při psaní efektivních uživatelských příběhů aplikace je klíčové porozumět perspektivě koncového uživatele. Tento proces vyžaduje identifikaci potřeb, motivací a požadovaných výsledků uživatele při používání aplikace.

První krok je identifikace a pochopení různých *personas*, neboli **typů uživatelů**, kteří budou pravděpodobně s aplikací interagovat. V případě zkoumané aplikace je primárním uživatelem někdo, kdo má zájem o nákup vstupenek na události. Sekundární uživatelé, jako jsou organizátoři akcí nebo manažeři prostorů, však mohou také s aplikací interagovat s odlišnými požadavky na funkcionalitu, nicméně jejich potřeby nejsou v rámci této práce důležité.

Dalším krokem je zjištění, čeho uživatelé **chtějí dosáhnout**. To může zahrnovat jednoduché úkony, jako například *prohlížení mapy místa konání*, nebo složitější, jako *rezervace konkrétních sedadel*. Každý uživatelský příběh by měl zůstat stručný a zaměřený na jednu akci.

Posledním krokem je definování **požadovaného výsledku**, který uživatel získá provedením daného úkonu, neboli definování motivace uživatele. Tento krok je klíčový, jelikož dále napomáhá při prioritizaci funkcí na základě získané hodnoty, kterou poskytuje uživateli.

Při tvoření uživatelských příběhů je užitečné dodržovat princip *INVEST* (*Independent, Negotiable, Valuable, Estimable, Small, Testable*). Tento princip zajišťuje, že každý uživatelský příběh je dobře definován a má potřebné charakteristiky pro efektivní implementaci v procesu vývoje[11].

Příkladem takového uživatelského příběhu v rámci zkoumaného webového řešení prodeje vstupenek s rezervací míst může být následující:

Ukázka uživatelského příběhu

“Jako zákazník si chci být schopen vybrat konkrétní sedadlo, abych si mohl zakoupit vstupenku na akci.“

Tento uživatelský příběh je nezávislý na ostatních uživatelských příbězích, je jednoduchý a snadno pochopitelný, poskytuje hodnotu uživateli a je snadno testovatelný. Při dodržení tohoto principu mohou uživatelské příběhy poskytnout cenný náhled do toho, jak by měla výsledná aplikace fungovat z pohledu uživatele.

2.3.3 Návrh konkrétních uživatelských příběhů

Na základě pochopení získaného v předchozích sekcích se lze nyní zaměřit na konstrukci konkrétních uživatelských příběhů pro zkoumanou aplikaci zaměřenou na prodej vstupenek s rezervací míst. Nejprve je však nutné definovat hlavní typ uživatele, který bude tuto aplikaci používat.

V základu lze říci, že hlavní rolí uživatele je potencionální zákazník, který má zájem o nákup vstupenky na konkrétní událost. Pro další účely bude použito pouze záměrné označení **zákazník**. Každý příběh bude představen ve stanoveném formátu a na závěr bude diskutováno o tom, jak daný příběh ovlivňuje návrh uživatelského rozhraní.

Uživatelský příběh 1 – Vizualizace místa konání

“Jako zákazník, chci vidět, jak vypadá místo konání, abych si mohl vybrat místo, které mi bude vyhovovat.“

Tento uživatelský příběh zdůrazňuje důležitost jasné a intuitivní vizualizace místa konání. Mapa musí poskytovat přesnou reprezentaci uspořádání sedadel a nabízet dostatek detailů, aby uživatelé mohli snadno vybrat místo, které jim vyhovuje.

Uživatelský příběh 2 – Výběr sedadla

“Jako zákazník, si chci označit či odznačit konkrétní sedadla, abych si mohl vybrat místa, která mi budou vyhovovat.“

Flexibilní výběr sedadla je klíčovým aspektem pro uživatele, jelikož umožňuje volnost ve výběru sedadel. Uživatelé by měli mít možnost vybrat si konkrétní místo, které jim vychovuje, a měli by mít možnost si vybrat případně i více míst, pokud si přejí sedět například s přáteli nebo rodinou.

Uživatelské rozhraní by tedy mělo uživatelům umožnit snadno vybrat a zrušit výběr míst, aby mohli vyzkoušet různé možnosti výběru, které jsou k dispozici.

Uživatelský příběh 3 – Nákupní košík

“Jako zákazník, chci mít jasný přehled o přidaných vstupenkách do nákupního košíku, abych měl přehled o svém nákupu.“

Tento uživatelský příběh zdůrazňuje důležitost uživatelského rozhraní nákupního košíku se vstupenkami. Uživatelé by měli mít možnost snadno zobrazit, jaké vstupenky mají v nákupním košíku, a měli by mít možnost snadno upravovat jeho obsah.

To vyžaduje jednoduché a přístupné uživatelské rozhraní spravující komplexní funkci nákupního košíku.

Uživatelský příběh 4 – Vyřízení objednávky

“Jako zákazník, chci jasný a jednoduchý proces vyřízení objednávky, abych mohl svůj nákup vstupenek snadno dokončit.“

Poslední zmíněný uživatelský příběh se zaměřuje na proces vyřízení objednávky. Zdůrazňuje potřebu jednoduchého a intuitivního uživatelského rozhraní, které umožní uživatelům snadno dokončit svůj nákup vstupenek.

Uživatelské rozhraní dokončení objednávky by tedy mělo minimalizovat komplexitu a zmatečnost, čímž zajistí uživatelům důvěru při dokončování své objednávky.

Z důvodu zachování přehlednosti a jednoduchosti, ačkoliv by mohlo být vytvořeno více uživatelských příběhů, budou v této práci použity pouze tyto čtyři hlavní uživatelské příběhy. Tyto příběhy, budou v dálce použity jako základní stavební kameny pro návrh uživatelského rozhraní a jeho komponent v rámci vyvíjené webové aplikace.

Pro to bude ale nejdříve nutné vybrat vhodný návrhový nástroj, který bude schopen tyto příběhy transformovat do funkčního designu. Další sekce se nyní bude zabývat nástroji dostupnými pro návrh UI a bude diskutováno o důvodech výběru konkrétního nástroje pro tento projekt.

2.4 Nástroje pro návrh

Při navrhování efektivního a efektivního uživatelského rozhraní hraje výběr vhodného návrhového nástroje zásadní roli. Vybraný nástroj musí mít schopnosti transformovat koncepční nápady a uživatelské příběhy do funkčního designu při dodržení nejlepších postupů a standardů. Všechny tři široce uznávané nástroje v oblasti návrhu uživatelského rozhraní, jmenovitě *Adobe XD*, *Figma* a *Sketch*, nabízejí jedinečnou sadu funkcí a mají své vlastní silné a slabé stránky[12].

Následující části poskytují přehled těchto tří nástrojů s podrobným popisem jejich klíčových funkcí, silných a slabých stránek. Po srovnávací analýze bude zvolen nástroj pro tento projekt a jeho výběr bude zdůvodněn.

2.4.1 Adobe XD

Adobe XD (Experience Design) je vektorový nástroj vyvinutý a publikovaný společností *Adobe Inc.*. Primárně se používá pro návrh a prototypování UX pro webové a mobilní aplikace. *Adobe XD* usnadňuje celý proces návrhu od nápadu a návrhu po prototypování a náhled UX [13].



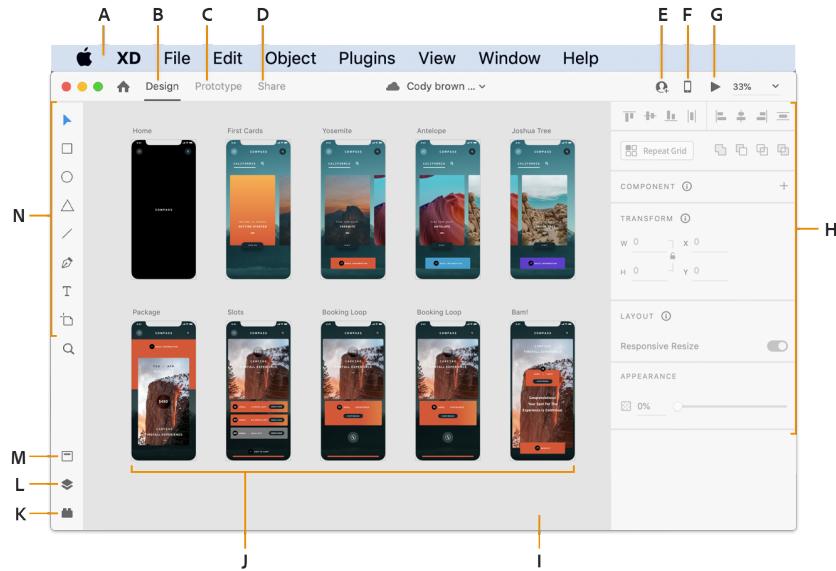
Obrázek 2.3: Logo nástroje Adobe XD[13]

Adobe XD, jehož logo je zobrazeno na obrázku 2.3, je součástí *Adobe Creative Cloud*, který se integruje s dalšími nástroji *Adobe*, jako je *Photoshop* a *Illustrator*, a nabízí bezproblémovou kompatibilitu souborů. Mezi jeho klíčové funkčnosti patří:

- **Repeat Grid:** Tato funkce umožňuje návrhářům replikovat designové prvky, což šetří čas při návrhu složitých UI.
- **Prototypování a animace:** *Adobe XD* poskytuje funkce pro vytváření interaktivních prototypů s lehkostí.
- **Voice Prototyping:** *Adobe XD* poskytuje prototypování hlasu, které lze použít pro návrh hlasových asistentů a dalších hlasem ovládaných aplikací.

- **Kolaborace:** Adobe XD podporuje kolaboraci v reálném čase, což umožňuje více členů týmu pracovat na stejném návrhu současně.
- **Integrace:** Adobe XD se integruje s dalšími nástroji v sadě *Adobe Creative Cloud* a také s pluginy třetích stran, což poskytuje širokou škálu dalších funkcí.

Na obrázku 2.4 je zobrazena ukázka rozhraní Adobe XD.



Obrázek 2.4: Ukázka rozhraní nástroje Adobe XD[13]

Silné stránky:

- Bezproblémová integrace s dalšími nástroji *Adobe*.
- Výkonné možnosti prototypování a animace.
- Schopnost navrhovat pro různé platformy včetně webu, mobilu, tabletu a dalších.
- Silná podpora komunity a zdrojů pro učení.

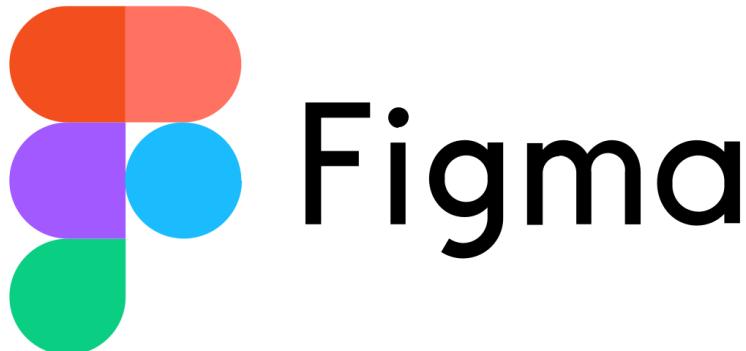
Slabé stránky:

- Omezená podpora pro složité animace ve srovnání s některými jinými nástroji.
- Je vyžadováno předplatné služby *Adobe Creative Cloud*, což může být pro některé uživatele drahé.

- Možnosti kolaborace jsou dobré, ale mohou být omezenější než v některých jiných nástrojích[14, 13].

2.4.2 Figma

Figma je webový nástroj pro návrh UI a prototypování. Získal významnou popularitu díky své pokročilé funkčnosti kolaborace a jeho webové založnosti, která umožňuje spolupráci v reálném čase, což jej činí oblíbeným nástrojem pro mnoho týmů[14]. Na obrázku 2.5 je zobrazeno logo nástroje Figma a na obrázku 2.6 níže je zobrazena ukázka jeho rozhraní.



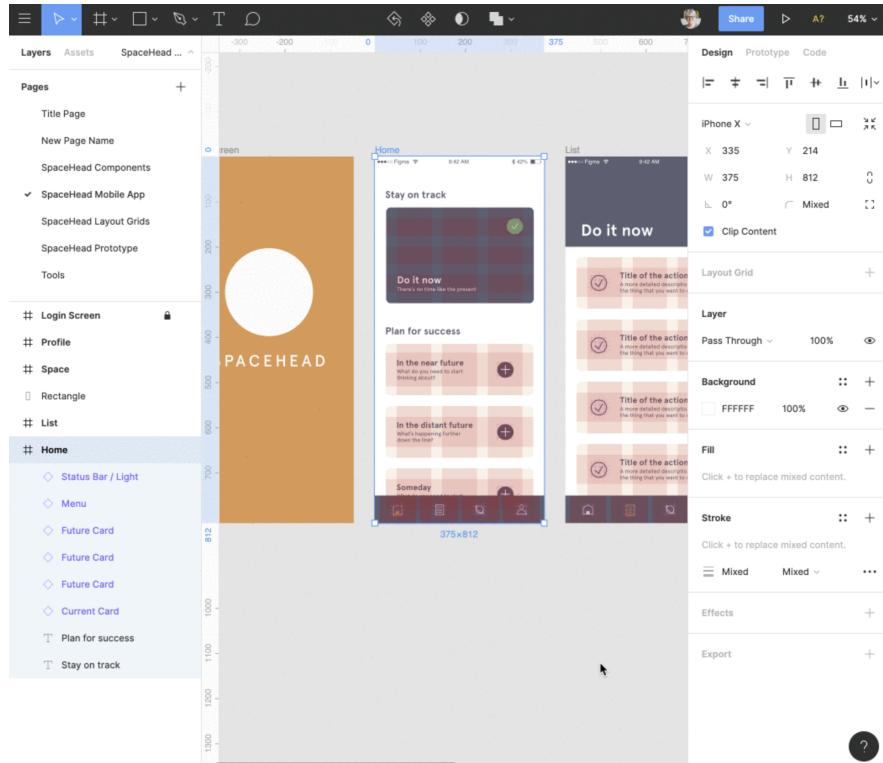
Obrázek 2.5: Logo nástroje Figma[15]

Figma umožňuje, aby celý proces návrhu probíhal v rámci nástroje: od *brainstormingu*¹ po vytváření interaktivních prototypů. Mezi některé klíčové funkce tohoto nástroje patří:

- **Kolaborace v reálném čase:** Více lidí může současně pracovat na návrhu, podobně jako v *Google Docs*, což z Figma činí skvělý nástroj pro týmové projekty.
- **Komponenty a style:** Figma podporuje komponenty (opakově použitelné designové prvky) a styly, které podporují konzistenci napříč návrhy a šetří čas.
- **Prototypování a animace:** Figma umožňuje vytvářet interaktivní prototypy s přechody. Ačkoli nemusí být tak robustní jako některé jiné nástroje, slouží pro většinu návrhových potřeb.

¹ *Brainstorming* je technika, která se používá k vytváření velkého množství nápadů na řešení problému.

- **Auto-layout:** Tato funkce umožňuje návrhářům vytvářet responzivní rozložení s lehkostí, což výrazně zrychluje proces návrhu.



Obrázek 2.6: Ukázka rozhraní nástroje Figma[15]

Silené stránky:

- Umožňuje spolupráci a společné úpravy v reálném čase.
- Není třeba stahovat software; veškerá práce je uložena a přístupná prostřednictvím cloudu.
- Zahrnuje design, prototypování a předávací nástroje na jednom místě.
- Nezávislé na platformě, funguje v prohlížeči.

Slabé stránky:

- Omezené možnosti práce v offline režimu
- Pro většinu funkcí je vyžadováno připojení k internetu.
- Na méně výkonných počítačích může běžet pomaleji[14, 16].

2.4.3 Sketch

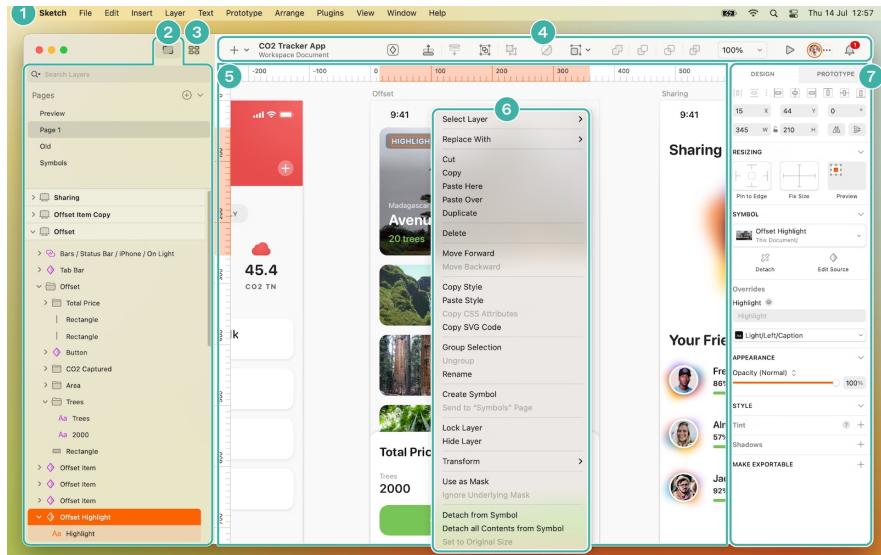
Sktech, jenž byl kdysi průmyslovým standardem pro návrh UI a UX, je vektorový nástroj pro návrh UI a UX pro macOS. Byl spuštěn v roce 2010 a od té doby získal významnou uživatelskou základnu[14]. Logo tohoto nástroje je zobrazeno na obrázku 2.7 a ukázka jeho rozhraní je zobrazena na obrázku 2.8 níže.



Obrázek 2.7: Logo nástroje Sketch[17]

Sketch je primárně určen pro návrh rozhraní a prototypování. Některé z jeho klíčových funkcí jsou:

- **Symboly a styly:** Sketch podporuje symboly (opakovatelné designové prvky) a styly, které podporují konzistenci napříč návrhy a šetří čas.
- **Pluginy:** Jednou z hlavních sil nástroje Sketch je jeho široká škála pluginů, které lze použít k rozšíření jeho funkcionality.
- **Prototypování:** Zatímco Sketch byl původně za jinými nástroji v oblasti prototypování, v této oblasti učinil významné pokroky.
- **Sdílené knihovny:** Sketch má silnou podporu pro sdílené knihovny, což umožňuje snadné sdílení zdrojů, komponent a stylů napříč projekty.



Obrázek 2.8: Ukázka rozhraní nástroje Sketch[17]

Silené stránky:

- Široká škála rozšíření/pluginů.
- Silná komunita a dostupnost zdrojů.

Slabé stránky:

- Dostupný pouze pro macOS, což omezuje jeho dostupnost.
- Bez vestavěné podpory pro spolupráci nebo společné úpravy[14, 17].

2.4.4 Výběr nástroje

Aby bylo možné učinit rozhodnutí a vybrat nejvhodnější nástroj pro návrh uživatelského rozhraní pro tento projekt, je nezbytné provést srovnávací analýzu tří nástrojů, které jsou podrobně popsány výše - Adobe XD, Figma a Sketch. Nástroje jsou porovnávány v několika klíčových aspektech, včetně platformové a cenové dostupnosti, prototypovacích schopnostech, správy komponent, stylování a knihoven, podpory pluginů, komunity a dostupných zdrojů.

Platformová dostupnost

Figma a Adobe XD vynikají svou dostupností napříč platformami. Oba nástroje lze používat na Windows a macOS a Figma je dostupná i na Linuxu, případně

jiných OS, prostřednictvím webového prohlížeče. Sketch je však omezen pouze na macOS, což může způsobit problémy s kompatibilitou a možnou spoluprací.

Cenová dostupnost

Adobe XD nabízí bezplatný startovací plán, přičemž placené plány nabízejí více funkcí. Figma také nabízí bezplatný plán s dalšími funkcemi dostupnými v profesionálních a týmových plánech. Sketch poskytuje bezplatnou zkušební verzi s jednorázovou platbou za plnou verzi a dodatečnými náklady na roční aktualizace.

Prototypování

Adobe XD, Figma i Sketch nabízejí možnosti prototypování. Adobe XD a Figma však poskytují podporu pro složitější a interaktivnější prototypy ve srovnání s aplikací Sketch, která má základní možnosti vytváření prototypů.

Komponenty

Adobe XD, Figma i Sketch nabízejí knihovny komponent pro zjednodušení procesu navrhování. Komponenty a styly Figma jsou pokročilejší, což umožňuje lepší organizaci a efektivitu.

Stylování a knihovny

Figma a Sketch mají silnou podporu pro sdílené styly a knihovny, které umožňují konzistenci v designu. Adobe XD také podporuje sdílené prostředky, ale jeho možnosti nejsou tak robustní jako u nástrojů Figma a Sketch.

Pluginy

Sketch původně vedl v podpoře pluginů, ale Adobe XD a Figma jej rychle dohnali a všechny tři nástroje nabízejí širokou škálu pluginů pro rozšíření funkčnosti.

Komunita a zdroje

Sketch je nejstarší z těchto tří nástrojů a má největší komunitu a nejširší škálu zdrojů. Nicméně Figma, přestože se jedná o relativně nový nástroj, si díky své popularitě a širokému použití vybudoval silnou komunitu. Adobe XD má také podpůrnou komunitu, ale není tak rozsáhlá jako u nástrojů Figma nebo Sketch.

Stručně řečeno, všechny tři nástroje mají své vlastní silné stránky a uspokojují různé potřeby a preference. Toto celé srovnání je souhrnně zobrazeno v tabulce 2.1.

Po rozboru této analýzy byl vybrán jako preferovaný nástroj pro návrh uživatelského rozhraní zkoumané aplikace nástroj **Figma**. Tato volba byla motivována kombinací několika přesvědčivých faktorů, včetně:

Tabulka 2.1: Srovnání nástrojů pro návrh uživatelského rozhraní

Funkčnost	Adobe XD	Figma	Sketch
Podpora Windows	Ano	Ano	Ne
Podpora macOS	Ano	Ano	Ano
Webová aplikace	Ne	Ano	Ne
Dostupné zdarma	Ano	Ano	Trial verze
Placené plány	od \$9.99/měs.	od \$12/měs.	\$99/jednorázově
Prototypování	Ano	Ano	Ano
Komponenty	Ano	Ano (pokročilé)	Ano
Stylování/knihovny	Ano	Ano (pokročilé)	Ano
Pluginy	Ano	Ano	Ano
Komunita a zdroje	Dobrá	Výborná	Výborná

- Dostupnost a podpora platformy:** Figma je platformně nezávislý nástroj a podporuje různé operační systémy, včetně Windows, macOS a dokonce i Linuxu prostřednictvím webového prohlížeče. Tato široká dostupnost je významnou výhodou oproti Sketch, který je dostupný pouze pro macOS.
- Cena:** Figma poskytuje robustní bezplatnou verzi, která umožňuje až tři aktivní projekty. Pro tento projekt jsou funkce poskytované v bezplatné verzi dostačné, což z Figma činí ekonomicky výhodnou volbu.
- Pokročilý systém komponent:** Systém komponent nástroje Figma je pokročilý a umožňuje hierarchickou organizaci komponent. To zajišťuje konzistenci v designu a zjednoduší proces, zejména při práci s komplexními designy, jako je interaktivní mapa sedadel.
- Schopnosti prototypování:** Figma značně vyniká, pokud jde o prototypování - umožňuje vytvářet interaktivní prototypy pomocí řady funkcí a přechodů.
- Komunita a zdroje:** Figma se pyšní rozsáhlou online komunitou a množstvím zdrojů, zajišťuje, že řešení potenciálních problémů během návrhového procesu jsou snadno řešitelné.
- Osobní zkušenost:** Používá různé nástroje pro návrh, volba nástroje Figma byla také motivována osobní zkušeností s tímto nástrojem.

Tento nástroj kombinuje silnou sadu funkcí s osobní preferencí pro jeho rozhraní a funkčnost, což z něj činí ideální volbu pro tento projekt.

Nyní je již možné přejít ke konkrétnímu návrhu uživatelského rozhraní rozborem jednotlivých uživatelských příběhů definovaných v přechozí kapitole 2.3.

2.5 Návrh produktu

User Interface a User Experience jsou oba úzce spjaty s pochopením potřeb uživatele. Když je produkt navrhován, není vždy okamžitě jasné, jakým způsobem by mělo být jeho uživatelské rozhraní vypadat. Uživatelské příběhy, pokud jsou správně navrženy, usnadňují spojení mezi potřebami uživatele a strukturální kompozicí produktu. Primárním cílem je v této sekci přetvořit výše navržené uživatelské příběhy do UI jednotlivých komponent uživatelského rozhraní, které nakonec utvoří interaktivní architekturu produktu.

UI komponenty jsou prvky, které tvoří interaktivní a vizuální aspekty uživatelského rozhraní produktu. Tyto komponenty mohou zahrnovat menší jednodušší designové prvky, jako jsou například tlačítka či ikony, nebo větší a složitější funkční jednotky, jako je ku příkladu navigační menu nebo interaktivní plánek sedaček.

Převedení uživatelských příběhů do UI komponent je klíčovým krokem tomto v procesu, který vyžaduje hluboké porozumění potřebám uživatele tak, jak jsou vyjádřeny v uživatelských příbězích, a schopnost vizualizovat, jak tyto potřeby mohou být zhmotněny prostřednictvím různých UI komponent. Pro produkt, jako je aplikace na prodej vstupenek, to znamená zaměřit se na komponenty, které umožňují uživatelům navigovat po interaktivním plánu sedadel, vybrat si svá požadovaná místa a dokončit nákup vstupenek s minimálním distrakcí a maximální jednoduchostí.

Tato kapitola se dále zaměří na proces převodu výše sestrojených uživatelských příběhů do konkrétních UI komponent, které budou tvořit výsledný návrh uživatelského rozhraní vyvíjené webové aplikace.

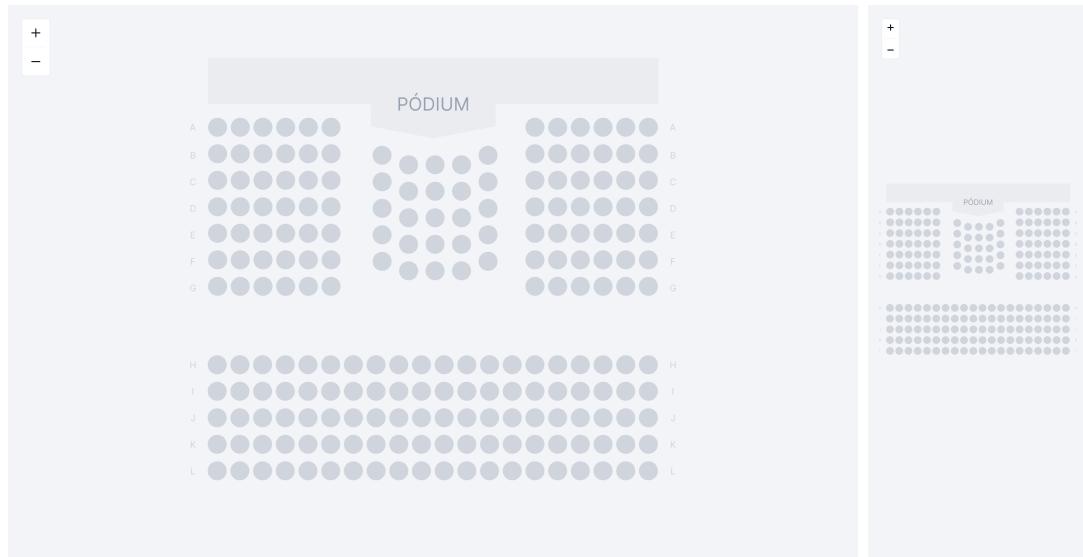
2.5.1 Vizualizace místa konání

Uživatelský příběh 1 – Vizualizace místa konání

“Jako zákazník, chci vidět, jak vypadá místo konání, abych si mohl vybrat místo, které mi bude vyhovovat.“

Základní myšlenkou tohoto příběhu je, že by uživatelé měli mít možnost zobrazit plán sedadel místa konání, aby mohli snadno vybrat svá preferovaná místa. To znamená navržení komplexní komponenty, která bude tuto funkčnost zajišťovat. Tato komponenta by měla být schopna zobrazit plán sedadel místa konání, který

bude obsahovat sedadla a případné další objekty umístěné v místě konání.



Obrázek 2.9: Návrh komponenty pro vizualizaci plánu sedadel místa konání

Na obrázku 2.9 je zobrazen návrh komponenty, která by měla být schopna zobrazit plán sedadel místa konání. Součástí této komponenty je i boční ovládací panel, které umožňuje manuální přiblížení či oddálení plánu sedadel. U této komponenty se předpokládá, že její následná ovladatelnost bude primárně zajištěna pomocí myši, případně pomocí gest na dotykové obrazovce.

2.5.2 Výběr sedadel

Uživatelský příběh 2 – Výběr sedadla

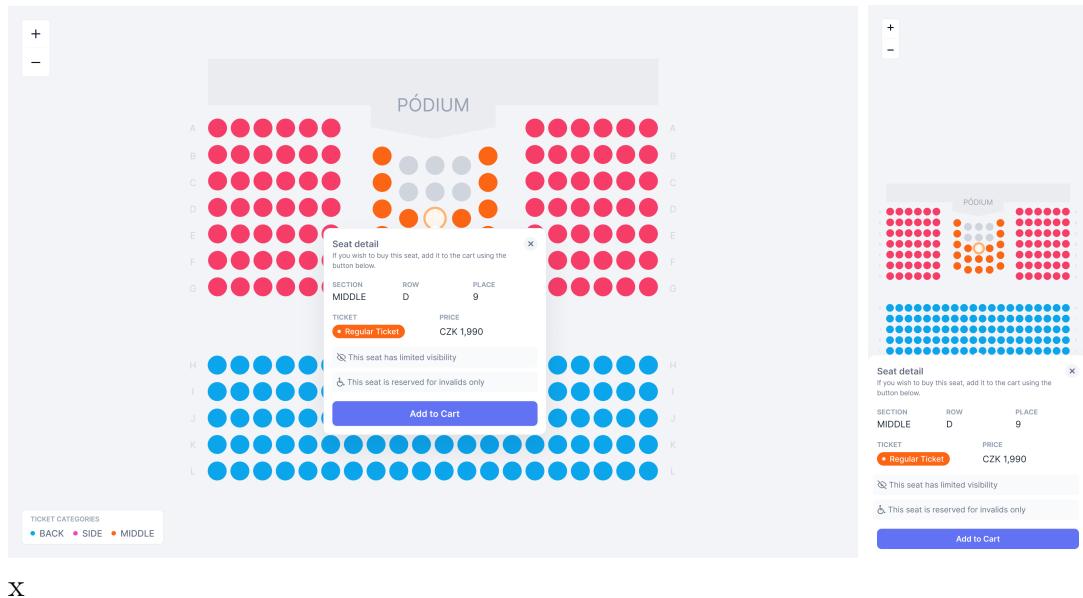
“Jako zákazník, si chci označit či odznačit konkrétní sedadla, abych si mohl vybrat místa, která mi budou vyhovovat.“

Tento uživatelský příběh se zaměřuje na možnost výběru sedadel, která uživatelé chtějí zakoupit. Uživatelé by měli mít kontrolu nad výběrem svých sedadel, což implikuje potřebu rozhraní, které nejen zobrazuje plán sedadel, ale také umožňuje interaktivní výběr a zrušení výběru jednotlivých sedadel.

Praktickým řešením tohoto uživatelského příběhu může být přidání klikacích prvků do plánu sedadel, které budou reprezentovat jednotlivá sedadla. Tyto prvky by měly po interakci jasně indikovat stav výběru či zrušení výběru, poskytující uživateli okamžitou vizuální zpětnou vazbu.

Jakmile je sedadlo vybráno, plán sedadel by měl tuto volbu odrazit změnou barvy sedadla nebo přidáním výrazného značení. Stejně tak, pokud je sedadlo zrušeno, mělo by se vrátit do svého původního stavu. Tato vizuální zpětná vazba nejen potvrzuje akci uživatele, ale také jim pomáhá sledovat jejich výběry, když se pohybují po místě konání.

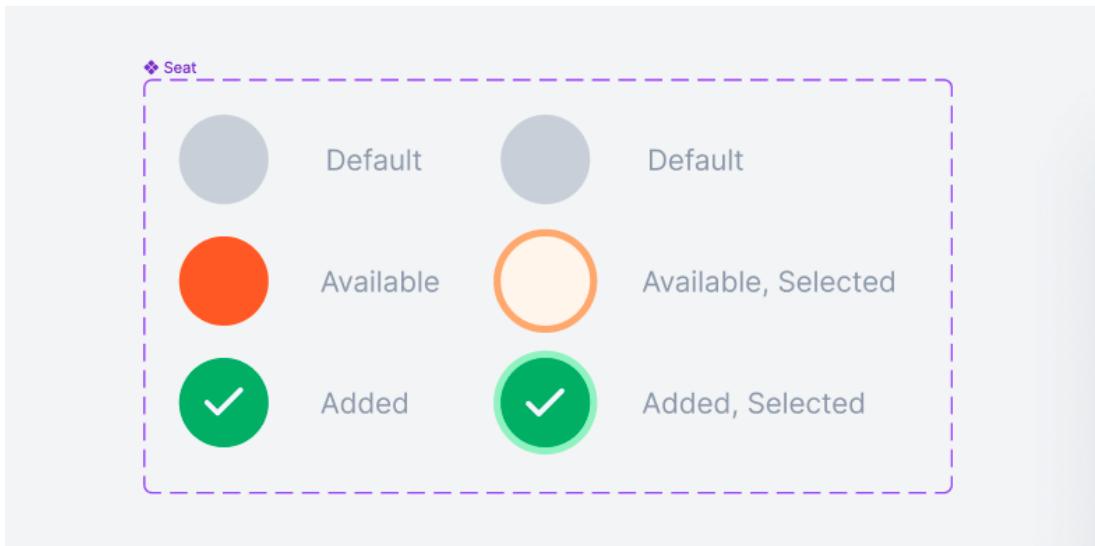
V otázce funkčnosti, je zásadní zajistit, aby proces výběru a zrušení výběru byl intuitivní a bezchybný. Pokud je sedadlo již zarezervováno či z jiného důvodu nedostupné, uživatelské rozhraní by mělo zabránit jeho výběru.



Obrázek 2.10: Návrh komponent pro výběr sedadel

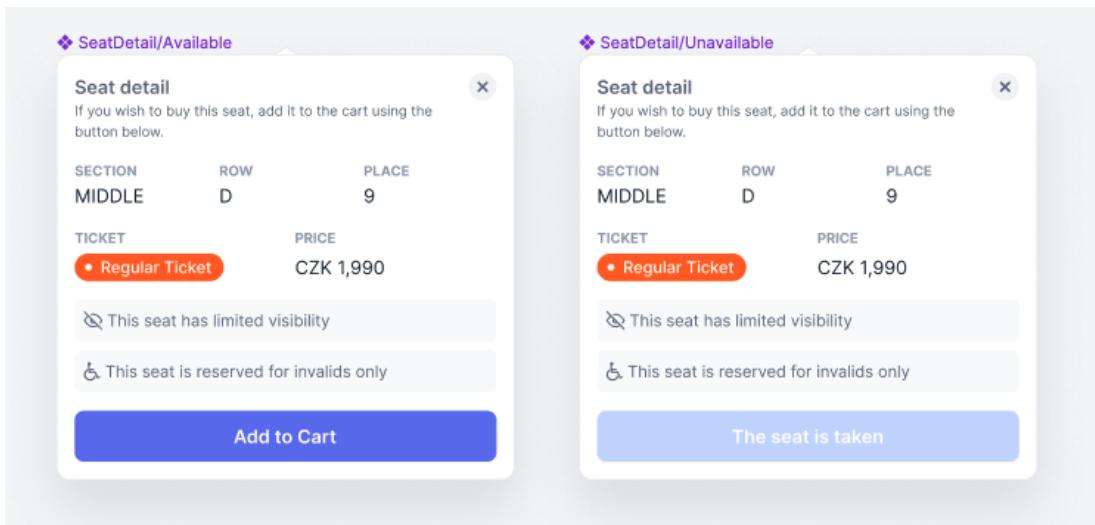
Na obrázku 2.10 je vyobrazen návrh interaktivní komponenty plánu míst, která umožňuje výběr sedadel. Volná sedadla k výběru jsou označena výraznými barvami, které odpovídají jejím cenovým kategoriím. Tyto kategorie, pro větší přehlednost, jsou zobrazeny v legendě, která je umístěna v pravém dolním rohu komponenty.

Pro tuto komponentu je zásadní jasně rozpoznat různé stavy sedadel, jako například dostupné, nedostupné, již přidané či aktuálně označené. Některé tyto stav mohou být navíc i různě kombinovány, jako například již přidané a aktuálně označené sedadlo. Tyto stavy jsou zobrazeny v rámci návrhy komponenty *Seat* na obrázku 2.11 níže.



Obrázek 2.11: Komponenta sedadla a její stavy

Součástí návrh této komplexní komponenty, ačkoliv není přímo součástí uživatelského příběhu, je i komponenta zobrazující přehledný detail právě vybraného sedadla. Sedadla totiž mohou nést různé informace, jako například číslo sedadla, jeho kategorie, cena, či informace o tom, zda je již obsazené nebo má nápríklad zhoršenou viditelnost na jeviště. Hlavní funkcí této komponenty je potvrzení výběru sedadla, tedy jeho přidání do košíku, či zrušení výběru. Na obrázku 2.12 je vyobrazen návrh této komponenty.



Obrázek 2.12: Komponenta detailu zvoleného sedadla

Tato komponenta funguje jakési modální okno, které se zobrazí po kliknutí na sedadlo. Ovšem jeho zobrazení je specifické na desktopové a mobilní verzi, jak je vyobrazeno na obrázku 2.10. Na desktopové verzi se zobrazí jako popover okno

umístěné u právě označeného sedadla. Na mobilní verzi se toto okno chová jako fixně umístěný list, který se zobrazí v dolní části obrazovky. Díky umístění v dolní části obrazovky na mobilních zařízeních se tak zajistí komfortní ovládání a interakce s tímto listem, jelikož se nachází v dosahu palce neboli v tzv. *thumb zone*.

2.5.3 Nákupní košík

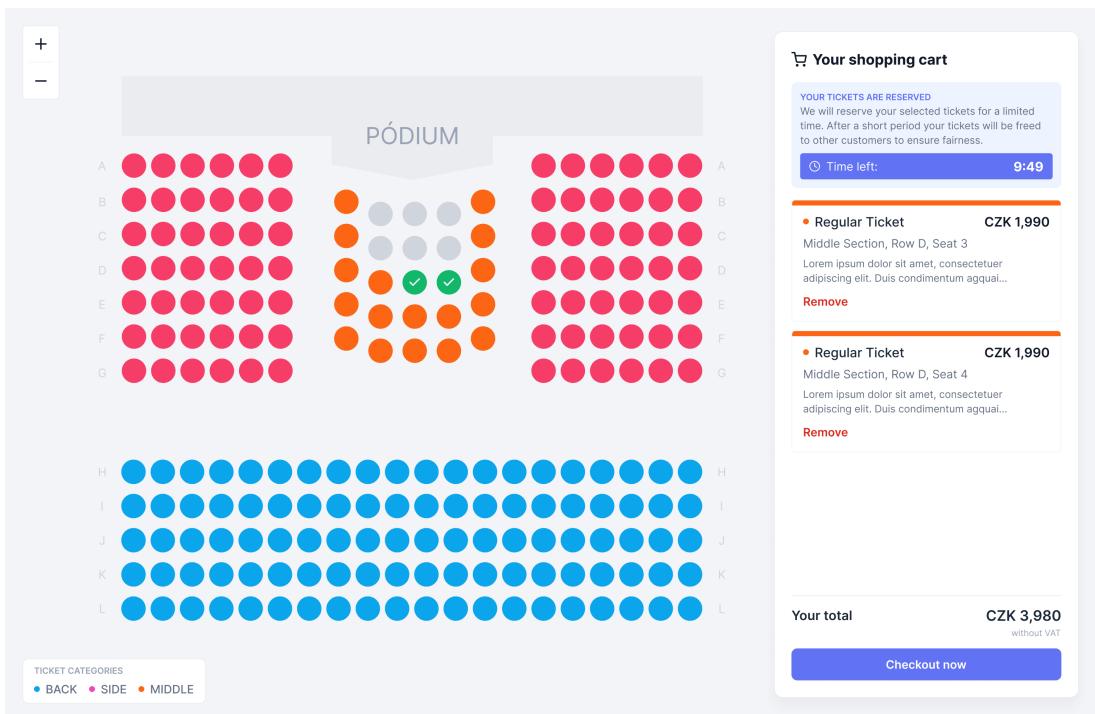
Uživatelský příběh 3 – Nákupní košík

“Jako zákazník, chci mít jasný přehled o přidaných vstupenkách do nákupního košíku, abych měl přehled o svém nákupu.“

Uživatel se nyní již umí pohybovat na mapě místa konání a vybrat si sedadlo, které chce zakoupit. Tento uživatelský příběh poukazuje na nutnost jasného přehledu aktuálního stavu nákupního košíku. Nákupní košík by měl uživateli sloužit jako přehled přidaných vstupenek s dalšími informacemi, jako například cena, kategorie, či počet vstupenek. Nákupní košík by také měl zobrazovat celkovou cenu všech vstupenek, které jsou v něm obsaženy. Poskytnutním těchto informací může zákazník potvrdit svůj výběr, sledovat celkovou cenu objednávky a rozhodnout se, zda bude pokračovat v nákupu, či provede úpravy.

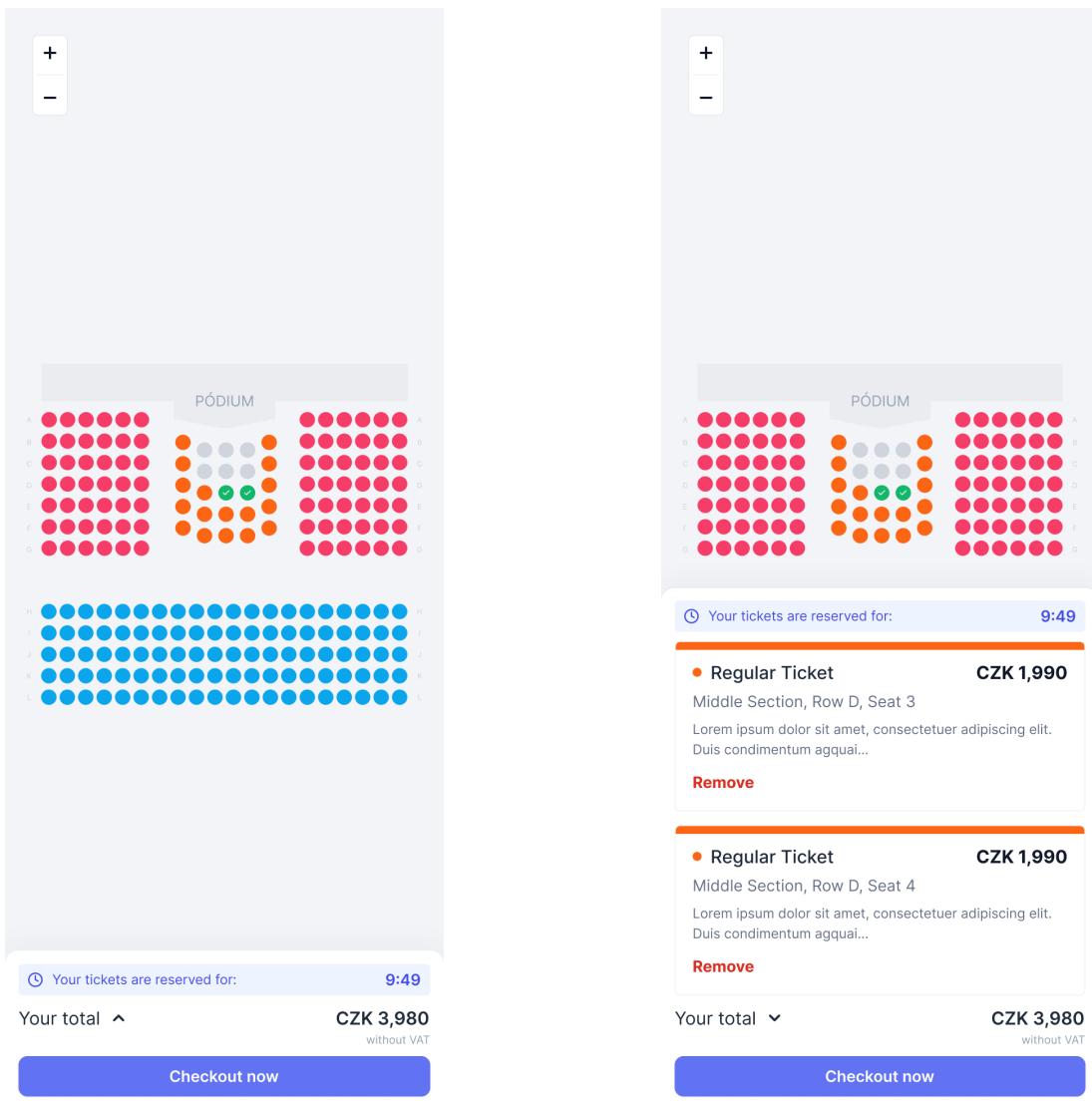
Ačkoliv není v uživatelském příběhu nijak definováno, lze z technických požadavků předpokládat, že celý proces vytváření objednávky bude probíhat v rámci rezervace pro zajištění zaručeného výběru sedadel. Tato rezervace bude mít časový limit, po jehož vypršení bude rezervace automaticky zrušena a sedadla budou opět uvolněna k prodeji. Tuto informaci je tedy také nezbytné zobrazit ideálně v rámci této komponenty nákupního košíku. Tuto komponentu nákupního košíku lze také použít jako navigační prvek, který umožní uživateli přejít k dalším krokům v procesu nákupu.

Na obrázku 2.13 je vyobrazen návrh komponenty přehledu nákupního košíku v desktopové verzi. Díky velkému prostoru, který je na desktopových zařízených k dispozici, je možné zobrazit všechny informace o vstupenkách, které jsou v košíku obsaženy. Tuto komponentu lze vyobrazit jako postranní panel, obsahující všechny tyto informace. Díky umístění této komponenty na pravou stranu celé obrazovky, kterou pokrývá komponenta mapy sedadel, je zajištěno, že uživatel bude mít vždy přehled o svém výběru, bude jasně viditelný a vyplní zbytečně nevyužitý prostor na obrazovce.



Obrázek 2.13: Návrh komponent přehledu nákupního košíku (desktopová verze)

Na mobilní verzi je situace poněkud odlišná. V této situaci je k dispozici mnohem méně prostoru, a proto je nutné zobrazit pouze nejdůležitější informace. Sekundární informace, jako například cena, či kategorie vstupenky, lze zobrazit například až po rozbalení detailu nákupního košíku.



Obrázek 2.14: Návrh komponenty přehledu nákupního košíku (mobilní verze)

Na obrázku 2.14 je návrh komponenty přehledu nákupního košíku v mobilní verzi, která je umístěna v dolní části obrazovky, aby byla v dosahu palce, a zároveň nezakrývala žádnou důležitou informaci. Ve výchozím (sbaleném) stavu obsahuje pouze ty nejdůležitější informace, jako celkovou cenu objednávky. Po rozbalení detailu nákupního košíku se zobrazí všechny informace o vstupenkách, které jsou v košíku obsaženy, podobně jako je tomu v desktopové verzi.

2.5.4 Vyřízení a potvrzení objednávky

Uživatelský příběh 4 – Vyřízení objednávky

“Jako zákazník, chci jasný a jednoduchý proces vyřízení objednávky, abych mohl svůj nákup vstupenek snadno dokončit.“

Proces vyřízení objednávky je posledním krokem v procesu nákupu vstupenek (pokud pomineme krok placení, který není řešen v rámci této práce). V tomto kroku je uživatel vyzván k zadání svých osobních údajů, které jsou nutné pro dokončení objednávky. Tyto údaje, pro jednoduchost, zahrnují pouze jméno, příjmení, e-mailovou adresu a telefonní kontakt. V reálném nasazení by bylo vhodné zahrnout i další údaje, jako například fakturační adresu, či případně další kontaktní údaje. Jako speciální údaj lze zákazníkovi nabídnout i možnost vyplnění poznámky k objednávce, která by mohla obsahovat například požadavek na zaslání faktury, či jiné informace.

Návrh celé této komplexní komponenty je zobrazen na obrázku 2.15 níže. Tato komponenta, jak je zřejmo, je složena z několika dalších menších částí.

The screenshot shows a summary of the purchase items:

Item Description	Quantity	Price
1x Regular Ticket (Middle, D4)	1	CZK 1,990
1x Regular Ticket (Middle, D3)	1	CZK 1,990
Subtotal		CZK 3,980
Service fees		CZK 0

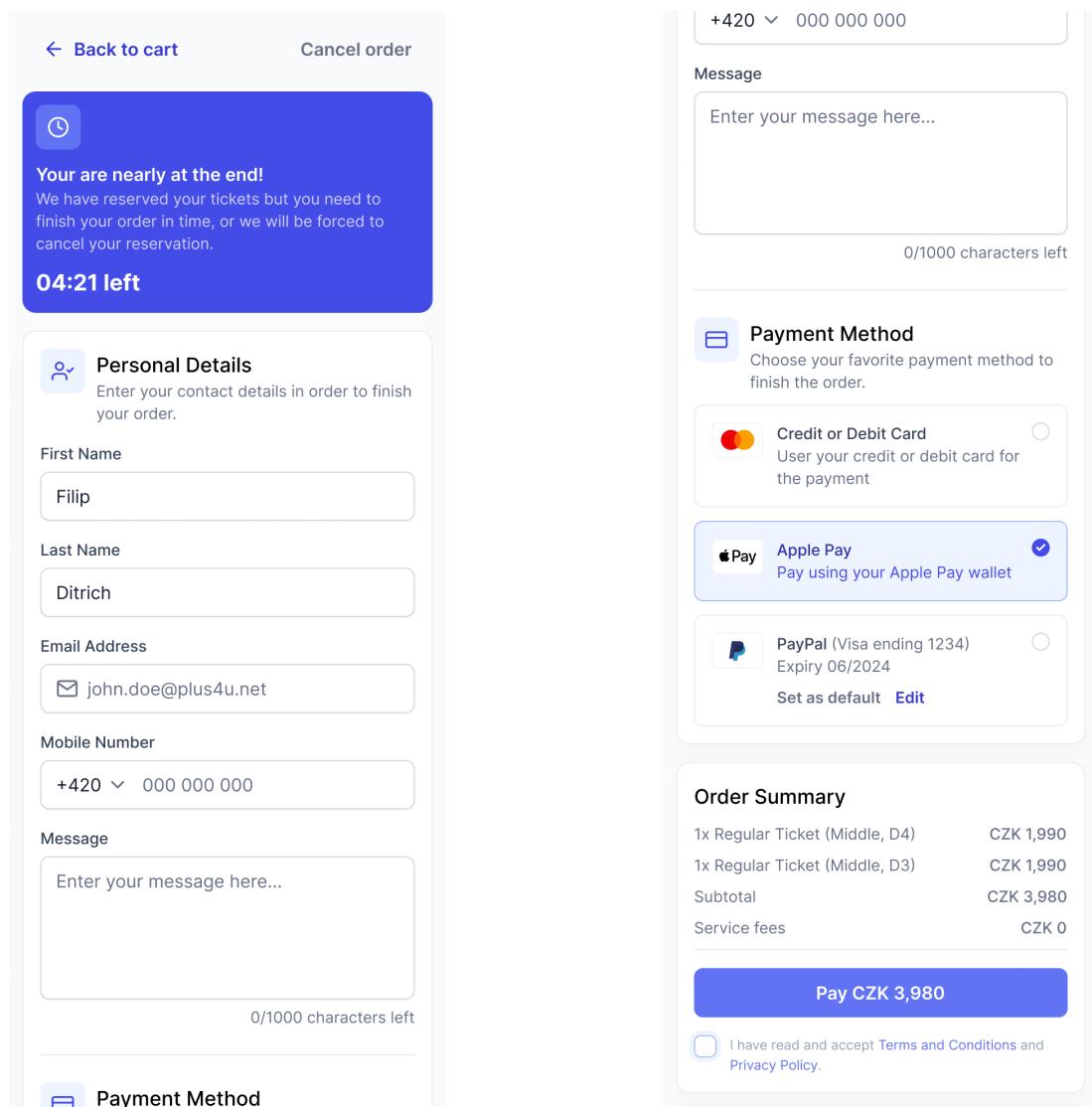
Below the summary is a blue button labeled "Pay CZK 3,980".

Obrázek 2.15: Návrh komponent dokončení objednávky (desktopová verze)

Primární částí je formulář, který obsahuje všechny potřebné údaje pro dokončení objednávky. Následuje výběr platební metody a souhrn objednávky, aby měl

zákazník stálý přehled o jeho objednávce. V neposlední řádě je zde stále zobrazena informace o probíhající rezervaci objednávky a jejím časovém limitu, v rámci kterého musí zákazník objednávku dokončit, jinak bude rezervace zrušena. Dále je zákazníkovi umožněno přejít zpět do procesu výběru vstupenek pomocí tlačítka *Back to cart* či proces vytváření objednávky kompletně zrušit pomocí tlačítka *Cancel order*.

V mobilním rozlišení je zobrazení této komponenty pouze mírně upraveno zavrháním a umístěním všech prvků do jednoho sloupce, jak je zobrazeno na obrázku 2.16 níže.



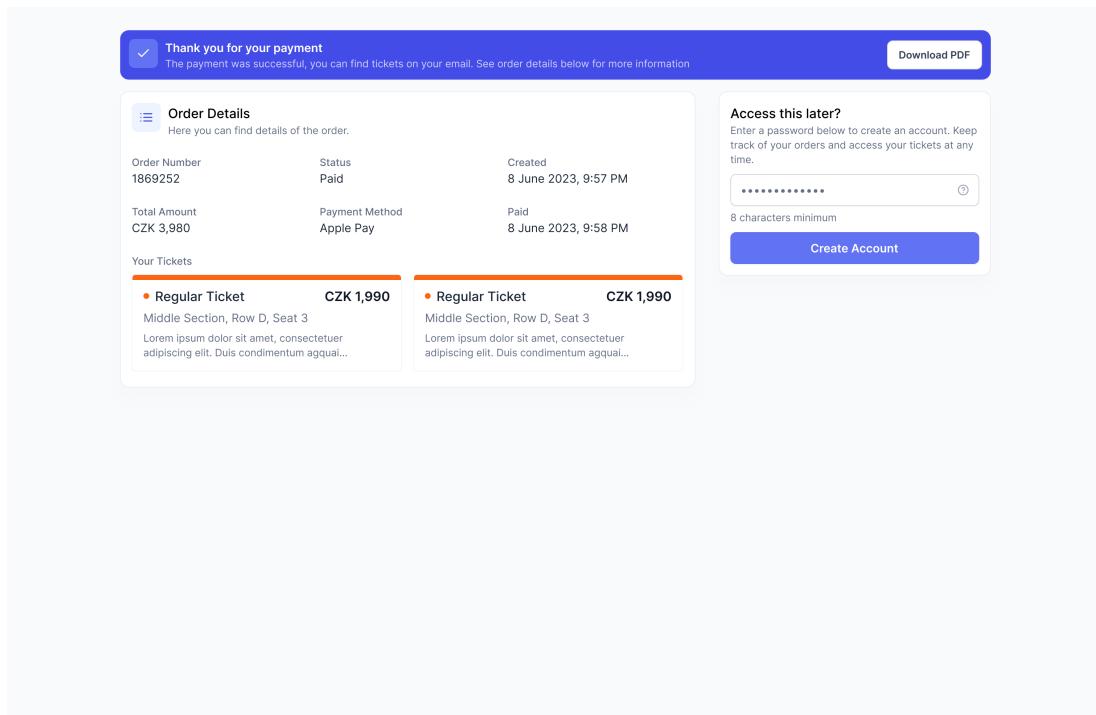
Obrázek 2.16: Návrh komponent dokončení objednávky (mobilní verze)

V obou případech je proces vyřízení objednávky zakončen kliknutím na intuitivní tlačítko *Pay*. Byť není v rámci této práce nutné, v reálném nasazení by bylo nutné

vyzvat zákazníka před dokončením objednávky k přečtení a potvrzení obchodních podmínek, případně dalších informací, které by mohly být pro zákazníka důležité. Proto je pod tlačítkem dokončení objednávky zobrazeno i povinné zaškrtávací políčko s přijetím obchodních podmínek, jak je zobrazeno na obrázku 2.15 výše či v nejspodnejší části obrázku ??.

Po kliku na tlačítko dokončení objednávky by byl v reálném světě zákazník přesměrován k zaplacení objednávky, dle jeho zvolené platební metody. V rámci této práce je však tento krok zjednodušen a zákazník je přesměrován na stránku s potvrzením dokončení objednávky.

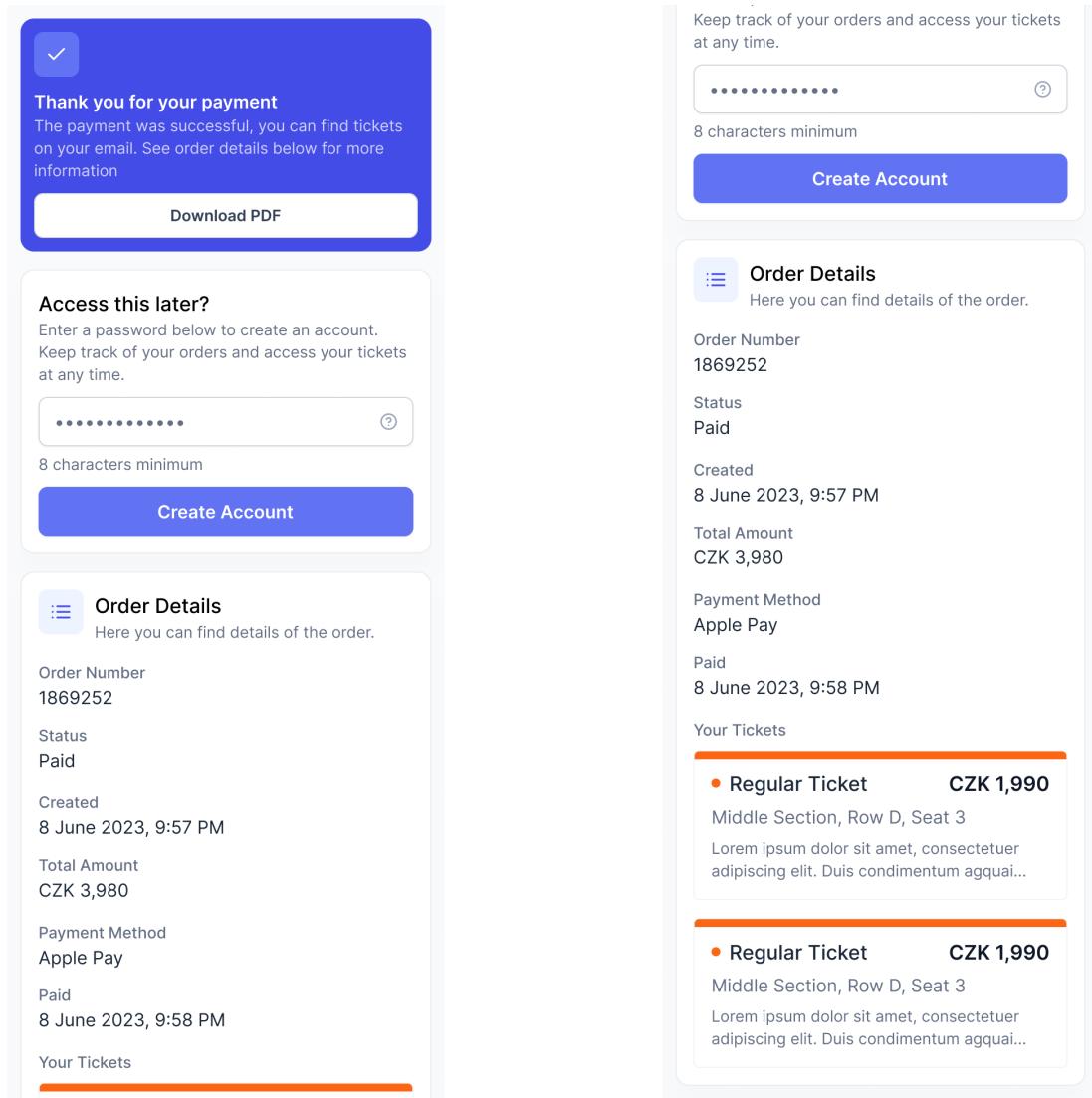
Tato stránka předpokládá zobrazení informací o zaplacené objednávce a dalšími užitečnými kroky pro zákazníka. Vrchní část této komponenty zobrazuje jasné potvrzení o dokončení objednávky a jejím zaplacení spolu s možností stažení PDF souboru s potvrzením o jejím zaplacení, jak je zobrazeno na obrázku 2.17 níže.



Obrázek 2.17: Návrh komponent potrvzení objednávky (desktopová verze)

Dále je v rámci této komponenty zobrazen přehled dostupných informací o objednávce, jako je například číslo objednávky, datum a čas jejího vytvoření, stav, částka, informace o platbě či seznam zakoupených vstupenek a jejich míst. V tomto kroku, byť opět není nutnou součástí zadání této práce, by bylo užitečné zákazníka dále vybídnout z dokončení registrace a vytvoření účtu, aby mohl v budoucnu snadněji sledovat stav svých objednávek a měl přehled o svých vstu-

penkách. Tento krok je naznačen v pravém sloupci na obrázku 2.17 výše, kde je zákazník vybídnut k vytvoření hesla pro svůj účet a tím tak vytvoření nového účtu.



Obrázek 2.18: Návrh komponent potrvzení objednávky (mobilní verze)

Mobilní rozložení této komponenty je opět pouze mírně upraveno, aby bylo možné všechny prvky zobrazit v jednom sloupci, jak je zobrazeno na obrázku 2.18 výše.

Tato kapitola vytvořila pevný základ pro pochopení návrhu uživatelského rozhraní, konkrétně v kontextu webového řešení pro prodej vstupenek s rezervací míst. To mj. zahrnovalo vytvoření uživatelských příběhů, jejich překlad do komponent uživatelského rozhraní a aplikaci efektivních principů návrhu uživatelského rozhraní. Dále bylo vyzdvihnut výběr nástroje Figma pro návrh těchto uživatelského rozhraní.

V následující kapitole se pozornost přesouvá k implementaci těchto návrhů do funkčního online rezervačního systému s výběrem míst.

3 Implementace aplikace

Vývoj frontendu je zásadním aspektem interakce uživatele s aplikací. Znamená to navrhnut vizuálně atraktivní a uživatelsky přívětivé rozhraní, které je také efektivní a přizpůsobitelné. Tato část se hluboce ponoří do specifik vývoje frontendu pro vyvýjenou aplikaci pro prodej vstupenek s rezervací míst a poskytuje komplexní informace o vybraných technologiích, architektuře projektu, klíčových funkcích a jejich implementaci.

Úvodní kapitola poskytuje vysvětlení pro volbu určitých technologií pro frontend, jako jsou React.js, TypeScript, Next.js, Mantine UI, Tailwind CSS a další nástroje. Odůvodnění začlenění těchto technologií a jejich příslušných funkcí do aplikace bude rozvedeno v části 3.2.

V části 3.3 je uveden přehled struktury a architektury projektu po vysvětlení technologického stacku¹. Tato část se bude zabývat postupy provedenými k zahájení projektu a rámcem navrženým pro usnadnění fungování této komplexní aplikace.

Oddíly 3.4 až 3.7 dokumentu se ponoří do základních funkcí aplikace, konkrétně do interaktivní mapy sedadel, správy nákupního košíku, rezervačního systému a procesu vyřízení objednávky. Tyto části poskytují podrobnou analýzu procesu implementace pro každou z těchto funkcí, včetně problémů, se kterým bylo v rámci vývoje čeleno, a strategií použitých k jejich překonání.

Část 3.9 představuje komplexní závěr, který shrnuje celý proces vývoje frontendu. Tato část zdůrazňuje zásadní poznatky, úspěchy a konečnou perspektivu aplikace.

¹Technologický stack je soubor softwarových technologií a nástrojů, které jsou používány vývojáři pro vývoj softwaru nebo webových stránek.

3.1 Úvod do vývoje frontendu

Vývoj frontendu, běžně označovaný jako vývoj na straně klienta, představuje klíčovou součást vývoje webu, která se soustředí na vizuální rozhraní a uživatelskou zkušenosť. Zahrnuje všechny aspekty webové aplikace, se kterou se uživatelé vizuálně zabývají, jako je její design, rozvržení, ovládání a obsah.

Základní komponenty používané při vytváření frontendu webových aplikací jsou Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) a JavaScript. HTML je odpovědné za strukturování obsahu webové stránky, CSS se používá k aplikaci stylů na webovou stránku a JavaScript se používá k vytváření interaktivních prvků na webové stránce[18].

Oblast vývoje frontendu přesahuje základní tři komponenty. V moderním vývoji webových aplikací využívají vývojáři frontendu sofistikované knihovny a frameworky JavaScriptu, jako jsou **React.js**, *Angular*, *Vue.js*, mimo jiné, k vytváření složitých a interaktivních uživatelských rozhraní[19]. CSS knihovny jako *Bootstrap* nebo **Tailwind CSS** navíc nabízejí předdefinované trídy pro urychlení procesu stylování[20].

Odpovědnosti frontendového vývojáře přesahují pouhou implementaci návrhu a organizaci obsahu. Zahrnují klíčové úkoly zajištění odezvy, výkonu a dostupnosti aplikace. To znamená zajistit, aby webová stránka bez problémů fungovala na řadě zařízení s různou velikostí obrazovky, a zároveň dodržovat strategie optimalizace výkonu, aby byla zajištěna bezproblémová uživatelská zkušenosť (ux). Kromě toho musí být aplikace navržena tak, aby vyhovovala uživatelům s různými schopnostmi a preferencemi[21].

Vývoj backendu se zaměřuje na server a funkčnost aplikace, zatímco vývoj frontendu se soustředí na vizuální a interaktivní aspekty aplikace, které zažívá koncový uživatel. Zahrnuje dovednou kombinaci estetiky a použitelnosti k vytvoření rozhraní, které je intuitivní a podmanivé svou odezvou na interakce uživatele[22].

V následujících částeč budou představeny a analyzovány vybrané technologie pro vývoj frontendu vyvíjené aplikace, objasní jejich jednotlivé funkce a zdůvodnění jejich použití.

3.2 Výběr technologií

Tato kapitola je zaměřena na technologický stack vybraný pro frontendovou implementaci webového řešení prodeje vstupenek s rezervací míst. Tento výběr není náhodný; každá technologie byla vybrána z určitého důvodu, ať už je to kvůli konkrétní funkčnosti, kterou poskytuje, její synergie s ostatními technologiemi ve stacku, nebo kvůli její robustnosti a spolehlivosti. Následující sekce podrobně popisují každou z těchto technologií.

3.2.1 React.js

React.js je JavaScriptová knihovna, která byla zpřístupněna veřejnosti jako *open-source*² nástroj. Byla vytvořena a v současné době je udržována společností Meta (dříve Facebook). Tato knihovna pomáhá při vývoji uživatelských rozhraní pro Single Page Application (SPA)³ tím, že umožňuje vývojářům vytvářet opakováně použitelné komponenty uživatelského rozhraní[25]. Výběr React.js pro tento konkrétní projekt je řízen řadou faktorů.

Zpočátku se knihovna může pochlubit charakteristikou virtuálního Document Object Model (DOM), která zvyšuje rychlosť a efektivitu generování složitých uživatelských rozhraní. Na rozdíl od konvenčního *full-refresh*⁴ přístupu, který je typický pro MPA, umožňuje virtuální DOM knihovně React omezit přímou manipulaci s DOM, čímž šetří životně důležité výpočetní zdroje a představuje hladší a robustnější rozhraní[26].

React.js navíc funguje na architektuře založené na komponentách, což umožňuje dekompozici složitých uživatelských rozhraní na menší, opakovaně použitelné komponenty. Tato metodika zlepšuje organizaci a údržbu kódu a také zefektivňuje proces ladění a testování[27]. Tato charakteristika se ukazuje jako obzvláště výhodná v oblasti vývoje komplexních aplikací, kde lze různé prvky uživatelského rozhraní znova použít ve více částech aplikace.

React.js těží z robustní komunitní podpory s množstvím zdrojů a nástrojů, které má k dispozici. Tato komunální podpora se ukazuje jako neocenitelná při vývoji

²Open-source software je typ software, který je zpřístupněn veřejnosti a jeho zdrojový kód je licencován tak, že umožňuje uživatelům studovat, měnit a distribuovat software kohokoli a za jakýchkoli účelů[23].

³SPA je typ webové aplikace nebo webové stránky, která interaguje s uživatelem dynamicky, nahrazuje tradiční model Multi Page Application (MPA)[24].

⁴Full-refresh je proces, kdy se celá stránka znova načte, když uživatel provede nějakou akci, která vyžaduje změnu v uživatelském rozhraní.

složitých aplikací, jelikož slouží k urychlení řešení potenciálních problémů.

3.2.2 TypeScript

TypeScript je programovací jazyk, který je staticky typovanou nadmnožinou JavaScriptu. Je vyvinut a spravován společností Microsoft a slouží k rozšíření funkčnosti JavaScriptu a zároveň usnadňuje vytváření složitých aplikací se zlepšenou spolehlivostí a sníženou náchylností k chybám[28].

Přestože je JavaScript robustní programovací jazyk, získal negativní zpětnou vazbu pro svou shovívavost ve svém dynamickém typovém systému. TypeScript se snaží tento problém vyřešit implementací statického typování, které minimalizuje chyby za běhu a poskytuje pokročilejší vývojové nástroje, včetně automatického dokončování, odvozování typu a kontroly typu[28].

V poslední době byl TypeScript velmi rozšířen a zpopularizován a stále více se stává normou pro vývoj rozsáhlých JavaScript aplikací[29]. Tento projekt preferuje TypeScript kvůli jeho vynikající spolehlivosti a kompatibilitě s React.js.

3.2.3 Next.js

Next.js je open-source React framework⁵, který zprvu zjednodušuje proces založení a nastavení nového projektu a zahrnuje spoustu dalších užitečných funkcí v dalších fázích vývoje, jako je Server Side Rendering (SSR) a Static Site Generation (SSG). Byl vytvořen společností Vercel, dříve známou jako ZEIT, a je aktivně udržován a rozvíjen rozsáhlou vývojářskou komunitou. Je navržen tak, aby zvýšil výkon a efektivitu React aplikací a vybavil vývojáře nástroji pro tvorbu aplikací s bohatými funkcemi[30].

V kontextu vyvíjené aplikace byl Next.js vybrán pro své funkce, které zjednoduší běžné úkoly. Například Next.js má vestavěnou podporu pro API endpointy, které umožňují vývojářům vytvářet backend API ve stejném repozitáři jako frontend, což je ideální například pro simulaci backendové funkcionality[30].

Next.js dále poskytuje funkce jako file-system routování⁶, automatické rozdělení kódu a mněho další, což snižuje množství potřebné konfigurace a nastavení a

⁵Framework je sada nástrojů, které usnadňují vývoj aplikací.

⁶File-system routování je funkce, která umožňuje vývojářům vytvářet strukturu souborů, která se přímo mapuje na URL adresy.

umožňuje vývojářům soustředit se na logiku aplikace[30].

V poslední řadě byl Next.js vybrán také pro svůj bezproblematický proces nasazení do cloudu pomocí platformy Vercel, která je vývojářům k dispozici zdarma a umožňuje nasadit aplikaci online během několika málo minut[30].

3.2.4 Mantine UI

Mantine je komplexní knihovna UI komponent pro React s důrazem na použitelnost, přístupnost a *developer experience*⁷. Zahrnuje řadu komponent, které lze kombinovat a přizpůsobit k vytvoření složitých a vizuálně bohatých uživatelských rozhraní. Tato knihovna nabízí jednoduší způsob tvorby uživatelských rozhraní ve srovnání s vytvářením každé komponenty od základů, což může být časově náročné a náchylné k chybám.

Použitím knihovny komponent, jako je Mantine, se zajistí, že komponenty použité v celé aplikaci jsou konzistentní čímž se nejen velmi snižuje potřeba repetitivního kódu, ale také se zvyšuje přehlednost a čitelnost kódu. Dále se díky důrazu Mantine na přístupnost (*accessibility*) zajišťuje, že aplikace je použitelná co nejširšímu publiku[31].

Díky vynikající dokumentaci a bohatému výběru komponent je Mantine preferovanou volbou pro projekt před jinými podobnými knihovnami, jako je například MaterialUI od společnosti Google[32].

3.2.5 Tailwind CSS

Tailwind CSS je CSS framework založený na *utility-first* přístupu⁸ pro rychlé vytváření vlastních designů. Na rozdíl od tradičních CSS frameworků, které poskytují hotové komponenty, umožňuje Tailwind vývojářům vytvářet vlastní designy poskytováním nízkoúrovňových utility tříd, které mohou kombinovat k vytváření jedinečných rozhraní[33].

V rámci vyvíjené aplikace je Tailwind CSS použit především pro snadné stylování rozložení prvků v aplikaci a rychlé prototypování. Jeho *utility-first* přístup zjednodušuje proces tvorby vlastních stylů a jeho responzivní modifikátory umožňují

⁷*Developer experience* označuje kvailtu nástrojů a procesů, které jsou k dispozici pro vývojáře. Jedná se o ekvivalent User Experience z pohledu vývojářů.

⁸*Utility-first* se zaměřuje na vytváření nízkoúrovňových utility tříd, které lze kombinovat k vytváření složitých rozhraní. Jedná se například o třídy jako `.text-center` nebo `.bg-red-500`.

vytvářet designy, které fungují na všech velikostech obrazovky bez nutnosti psaní vlastních CSS tříd[33], což značně zrychluje vývoj.

3.2.6 Ostatní technologie

Kromě hlavního technologického stacku diskutovaného výše bylo při vývoji aplikace klíčových několik dalších nástrojů a knihoven. Každý nástroj či knihovna slouží k jedinečnému účelu a zlepšuje vývojový proces v oblastech jako je kvalita kódu, získávání dat či správa formulářů. Následující sekce popisují každou z těchto kategorií a odpovídající nástroje nebo knihovny použité.

Kvalita kódu

Udržování kvality kódu je klíčové v jakémkoli softwarovém projektu, zejména při práci na komplexních aplikacích. Vysoká kvalita kódu usnadňuje čtení, údržbu a ladění. Pro zajištění kvality kódu v tomto projektu byly vybrány dva klíčové nástroje: *Prettier*⁹ a *ESLint*¹⁰.

Prettier je nástroj pro formátování kódu, který vynucuje konzistentní styl napříč celým *codebase*¹¹ tím, že kód analyzuje a formátuje podle vlastních pravidel. Nástroj automaticky formátuje kód pokaždé, když je například uložen, což zajišťuje, že kód je konzistentní v nastaveném formátu napříč celým projektem[34].

ESLint slouží jako nástroj pro analýzu statického kódu speciálně navržený pro detekci problematických vzorů v kódu JavaScript a TypeScript. Doplňuje Prettier, protože zatímco Prettier primárně řeší problémy s formátováním, ESLint se primárně zaměřuje na identifikaci potenciálních chyb a problémů s kvalitou kódu. Začleněním ESLint do vývojového procesu může být zachována kvalita kódu, což vede ke zvýšené bezpečnosti a předvídatelnosti v celém projektu[35].

Získávání dat

Získávání dat hraje zásadní roli v jakémoli aplikaci, která spolupracuje se serverem. V tomto projektu bylo pro účely správy komunikace se serverem využito

⁹<https://www.npmjs.com/package/prettier>

¹⁰<https://www.npmjs.com/package/eslint>

¹¹Codebase je označení pro celkový kódový základ aplikace.

knihoven *Axios*¹² a *React Query*¹³.

Axios je Hypertext Transfer Protocol (HTTP) klient, který funguje na základě *Promise*¹⁴ a je kompatibilní s prohlížečem i prostředím Node.js¹⁵. Jeho primární funkcí je usnadnit odesílání asynchronních HTTP požadavků na backend API a zároveň poskytuje užitečné funkce, jako je automatická transformace JavaScript Object Notation (JSON) struktur, ochrana na straně klienta proti Cross-Site Request Forgery (XSRF) a zpracování požadavků/odpovědí. Použití Axios může výrazně zlepšit efektivitu a snadnost získávání dat v aplikaci.

React Query je knihovna navržená pro React, která umožňuje snadnou synchronizaci dat mezi klientem a serverem. Poskytuje ukládání do mezipaměti (*cache*), aktualizace na pozadí a správu zastaralých dat automaticky bez nutnosti jakékoli konfigurace. Při použití ve spojení s Axios zjednoduší načítání dat, což v konečném důsledku snižuje potřebu dalších *state management*¹⁶ knihoven.

Formuláře

Pro správu formulářů v aplikaci byly použity knihovny *React Hook Form*¹⁷ a *Zod*¹⁸.

React Hook Form je knihovna pro správu formulářů v Reactu, která snižuje množství *boilerplate*¹⁹ kódu potřebného pro vytváření formulářů. Využitím vestavěných hooků²⁰ Reactu zjednoduší validaci formulářů a zlepšuje výkon tím, že minimalizuje počet *re-renderů*²¹.

Zod je knihovna pro deklaraci a validaci schémat pro TypeScript a JavaScript. Díky Zod lze deklarovat schémata a validovat je pomocí vestavěných funkcí. Použitím s React Hook Form zjednoduší validaci formulářů a zajišťuje, že data zadaná uživatelem odpovídají definovanému schématu.

¹²<https://www.npmjs.com/package/axios>

¹³<https://www.npmjs.com/package/@tanstack/react-query>

¹⁴Promise je JavaScriptový objekt, který slouží k reprezentaci asynchronních operací.

¹⁵Node.js je open-source, multiplatformní, JavaScriptové prostředí, které umožňuje spouštět JavaScriptový kód mimo webový prohlížeč[36].

¹⁶*State management* je proces správy stavu dat aplikace.

¹⁷<https://www.npmjs.com/package/react-hook-form>

¹⁸<https://www.npmjs.com/package/zod>

¹⁹*Boilerplate* je označení pro kód, který je nezbytný, aby bylo možné vykonat určitou akci. V tomto případě se jedná o nezbytný kód pro vytváření formulářů.

²⁰Hook je označení pro speciální React funkce, které lze využít v rámci životního cyklu komponenty[25].

²¹*Re-render* je proces znovuvykreslení komponenty, nejčastěj se děje na základě změny stavu komponenty.

3.2.7 Správa zdrojového kódu

Efektivní správa zdrojového kódu je nedílnou součástí vývoje softwaru, zejména v kolaborativních prostředích. Git, bezplatný a open-source distribuovaný systém pro správu verzí, je nejrozšířenějším nástrojem pro správu zdrojového kódu. Git umožňuje vývojářům efektivně sledovat a spravovat změny v kódu, což usnadňuje plynulý vývojový proces[37].

Nicméně Git sám o sobě nenabízí kompletní řešení pro správu projektu. **GitHub** doplňuje Git poskytováním webového rozhraní pro hostování a spolupráci na Git repozitářích. Mezi funkce GitHubu patří řízení přístupu, správa úkolů, sledování chyb, požadavků na funkce a další[38]. To z něj činí nezbytný nástroj pro správu a sledování průběhu projektů. Mimo jiné, GitHub nabízí také prémiový plán pro vzdělávací instituce, který umožňuje studentům a učitelům využívat všechny funkce GitHubu zdarma[39].

Vyvíjený prototyp byl tedy hostován prostřednictvím GitHub a je dostupný na adrese <https://github.com/filipditrich/bachelor-thesis-project>.

Kromě toho, bylo využito konvenčního zápisu *commit zpráv*²², standardizovaného stylu pro commit zprávy, který zvyšuje čitelnost a pomáhá generovat poznámky k vydání, označované jako *release notes*. Tato konvence specifikuje strukturu commit zpráv, včetně typu (feat, fix, chore, docs, atd.), volitelného rozsahu a stručného popisu. Například commit zpráva může mít podobu `feat(lib/feature): add cart feature, implement useCart hook to manage complex shopping cart state`, kde `feat` označuje novou funkcionalitu, `lib/feature` označuje oblast změny a `add cart feature, implement useCart hook to manage complex shopping cart state` je stručný popis změny. Tento příklad je z reálného commitu (3982b9a) vytvořeného v rámci vývoje projektu.

²²Commit je označení pro jednotku změny v Gitu. Commit zpráva je zpráva, která popisuje změny provedené v rámci commitu.

3.3 Struktura a architektura projektu

Zahájení a organizace nového projektu je klíčovou fází, která vytváří základy pro celou aplikaci. Tato část poskytuje přehled procesu a úvah zahrnutých do počátečního nastavení projektu a definování jeho architektury. Cílem bylo vyvinout frontendovou aplikaci, která bude udržitelná, škálovatelná a dobře strukturovaná, dodržující nejlepší postupy současného vývoje frontendu.

3.3.1 Vytvoření projektu

Projekt byl vytvořen pomocí příkazu `npx create-next-app --typescript`, nástroje pro rychlé nastavení nového projektu, který poskytuje Next.js. Příznak `--typescript` byl zahrnut, aby bylo jasné, že se má použít TypeScript místo samotného JavaScriptu. Tento příkaz vytvoří nový adresář se základní strukturou projektu Next.js a sadou přednastavených souborů[30].

Jakmile bylo počáteční nastavení projektu dokončeno, následovalo odstranění vygenerovaného boilerplate kódu a přidání vybraných knihoven a technologií pro projekt. Tento proces zahrnoval odstranění výchozích CSS souborů, jelikož projekt využívá Tailwind CSS pro stylování a přidání MantineUI, komplexní knihovny pro uživatelské rozhraní Reactu. Přidání ostatních technologií a knihoven zmíněných v sekci 3.2, jako jsou Prettier, ESLint, axios a react-query, byly také součástí této počáteční fáze nastavení.

V tomto projektu bylo pro správu balíčků upřednostněno použití `pnpm`. Vyniká svým efektivním zacházením s Node.js moduly, čímž zajišťuje rychlejší a spolehlivější *build proces*²³[40].

Dalším krokem bylo vytvoření celkové struktury projektu, která vyhovuje potřebám aplikace a podporuje osvědčené postupy. Dobře promyšlená a organizovaná struktura projektu je zásadní pro udržení čitelnosti kódu, škálovatelnosti a snadné navigace, zejména když *codebase* roste.

²³*Build proces* je proces, kdy se zdrojový kód překládá, neboli transpiluje, do spustitelného formátu, který může být spuštěn na cílovém zařízení.

3.3.2 Základní architektura

Efektivně definovaná struktura projektu hraje klíčovou roli při zajišťování udržovatelnosti, škálovatelnosti a čitelnosti kódu. Logická, jasná a konzistentně udržovaná struktura umožňuje vývojářům snadno procházet a porozumět aplikaci, i když se *codebase* časem rozrůstá a vyvíjí.

Tento projekt využívá modulární architekturu, kde je kód kategoricky organizován do odlišných modulů na základě jejich specifických funkcí. Tato metodologie slouží ke zlepšení čitelnosti a udržovatelnosti kódové základny zapouzdřením souvisejících logických komponent do samostatných modulů[41].

Podrobnější pohled na složky a soubory hlavní struktury projektu a jejich účely je zobrazen na ukázce kódu 1.

```
src/
|-- lib/
|   |-- components/
|   |-- features/
|   |-- graphics/
|   |-- hooks/
|   |-- types/
|   `-- utils/
|-- pages/
`-- styles/
next.config.js
package.json
.eslintrc.js
prettier.config.js
tsconfig.json
tailwind.config.js
```

Zdrojový kód 1: Vizualizace struktury projektu

- **src:** Toto je hlavní adresář obsahující veškerý zdrojový kód aplikace.
- **src/lib:** Tento adresář obsahuje hlavní funkce projektu, které jsou dále rozděleny do dalších podadresářů.
- **src/lib/components:** Tento adresář obsahuje znovupoužitelné komponenty, které jsou strukturovány do vlastních podadresářů, které obsahují Type-

Scriptové soubory pro jejich otypování a jejich implementaci spolu s *barrel*²⁴ souborem pro snadnější importování.

- **src/lib/features:** Tento adresář se skládá z konkrétních funkcí nebo logických částí aplikace, jako je API klient, správa košíku či mapa sedadel.
- **src/lib/graphics:** Tento adresář uchovává grafické zdroje, jako jsou Scalable Vector Graphics (SVG) ikony používané v aplikaci.
- **src/lib/hooks:** Tato složka obsahuje vlastní pomocné React hooky.
- **src/lib/types:** Tento adresář uchovává běžné typy TypeScript používané v celé aplikaci.
- **src/lib/utils:** Tento adresář obsahuje různé pomocné funkce používané v aplikaci.
- **src/pages:** Obsahuje komponenty stránek a endpointy.
- **src/styles:** Obsahuje globální styly aplikace.
- **next.config.js:** Konfigurační soubor pro Next.js, který umožňuje přizpůsobení různých aspektů frameworku.
- **package.json:** Soubor obsahuje metadata a seznam použitých knihoven v projektu.
- **.eslintrc.js, prettier.config.js, tsconfig.json:** Jsou konfigurační soubory pro ESLint, Prettier a TypeScript.
- **tailwind.config.js:** Je konfigurační soubor pro Tailwind CSS.

Každá z těchto složek a souborů hraje v aplikaci klíčovou roli a přispívá ke struktuře a architektuře projektu. V dalších sekcích budou podrobněji popsány konkrétní implementační detaily projektu, počínaje interaktivní mapou sedadel.

²⁴Barrel soubor je jednoduše soubor pojmenovaný jako *index*, který importuje ze všech souborů v rámci adresáře a znova je exportuje z jednoho místa.

3.4 Interaktivní mapa sedadel

Vývoj této webové aplikace zahrnoval několik klíčových prvků, z nichž každý představoval jedinečné výzvy a příležitosti. Tyto prvky jsou základem funkčnosti aplikace a každý z nich hraje důležitou roli v celkovém uživatelském zážitku.

Prvním z těchto nepostradatelných prvků je interaktivní mapa sedadel. Tento prvek umožňuje uživateli vizuálně procházet virtuální mapu sedadel v místnosti a vybrat si své preferované místo. Mapa sedadel uživatelům poskytuje přehled o všech dostupných sedadlech, jejich kategoriích a stavech, čímž usnadňuje proces výběru sedadla a nákupu vstupenky. Tato mapa je také zodpovědná za zobrazení informací o vstupenkách, které jsou přidruženy k vybraným sedadlům.

3.4.1 Struktura

Struktura tohoto prvku je postavena na několika komponentách, které spolu harmonicky spolupracují. Primárními komponentami jsou:

- `SeatingMapProvider`
- `SeatingMap`
- `VirtualMap`
- `Seat`
- `SeatSheet`

SeatingMapProvider je nadřazená komponenta, která získává data z API a zpřístupňuje je svým ostatním potomkům. Tato data zahrnují podrobnosti o místu konání, sedadlech, jejich stavech a vstupenkách. Více o získávání dat a jejich struktuře je popsáno v sekci 3.4.2.

SeatingMap je komponenta, která zpracovává vstupní SVG data reprezentující mapu místa konání. Přebírá SVG data z API a překládá je do formátu, který může být použit komponentou **VirtualMap**.

VirtualMap je zodpovědná za přidání interaktivnosti do komponenty **SeatingMap**. Přidává ovládací funkce mapy, jako je posouvání, přiblížování a oddalování, čímž poskytuje dynamický uživatelský zážitek.

Seat a **SeatSheet** komponenty představují jednotlivá sedadla na mapě a podrobnosti zobrazené při jejich výběru.

Souhra a struktura těchto komponent je znázorněna v zjednodušené ukázce kódu 2.

```
1 <SeatingMapProvider venue={venue}>
2   {/* seating map */}
3   <SeatingMap width={width} height={height}>
4     {/* virtual map */}
5     <VirtualMap minScaleFactor={1.1} maxScaleFactor={0.1}>
6       {/* individual seats */}
7       <Seat />
8       <Seat />
9       <Seat />
10      {/* + any other svg elements */}
11    </VirtualMap>
12
13  {/* seat sheet detail */}
14  <Sheet>
15    <Sheet.Header />
16    <Sheet.Body>
17      <SeatSheet />
18    </Sheet.Body>
19  </Sheet>
20 </SeatingMap>
21 </SeatingMapProvider>
```

Zdrojový kód 2: Struktura komponent mapy sedadel

3.4.2 Získávání a správa dat

Proces získávání a správy dat začíná komponentou **SeatingMapProvider**. Tato komponenta vytvoří požadavek na API, aby získala data týkající se uspořádání sedadel daného místa konání.

Odpověď API zahrnuje podrobnou SVG reprezentaci uspořádání sedadel a jejich jednotlivá data. Každé sedadlo obsahuje informace o jeho řadě, místě, plném názvu, dostupné kapacitě a přidružených vstupenkách a kategoriích. Každá vstupenka obsahuje informace o jejím id, názvu, volitelném popisu, ceně a kategoriích. Podrobná struktura dat je znázorněna v ukázce kódu 3.

```
1 /**
2  * Venue API response schema
3  * @export
4 */
5 export const venueApiSchema = z.object({
6   /** venue id */
7   venueId: z.string().uuid(),
8   /** venue name */
9   name: z.string(),
10  /** seating drawing (SVG) */
11  drawing: z.string(),
12  /** venue seat data */
13  seats: z.array(
14    z.object({
15      /** seat id */
16      seatId: z.string().uuid(),
17      /** seat row display label */
18      row: z.string(),
19      /** seat place number */
20      place: z.number(),
21      /** seat full display name label */
22      fullName: z.string(),
23      /** seat available capacity left */
24      capacityLeft: z.number(),
25      tickets: z.array(z.string().uuid()),
26      categoryId: z.string().uuid(),
27    }),
28  ),
29  /** venue ticket data */
30  tickets: z.array(
31    z.object({
32      /** ticket id */
33      ticketId: z.string().uuid(),
34      name: z.string(),
35      description: z.string().optional(),
36      price: z.number().int(),
37      categories: z.array(
38        z.object({
39          categoryId: z.string().uuid(),
40          price: z.number().int(),
41        }),
42      ),
43    }),
44  ),
45  /** venue ticket categories */
46  categories: z.array(
47    z.object({
48      /** category id */
49      categoryId: z.string().uuid(),
50      name: z.string(),
51      description: z.string().optional(),
52      color: z.string(),
53    }),
54  ).
```

Po získání těchto dat komponenta `SeatingMapProvider` tato data zpracuje do použitelnějšího formátu. Toto zpracování zahrnuje oddělení sedadel, vstupenek a kategorií do vlastních struktur, vytvářející hierarchii, která zjednodušuje přístup k datům pro následující komponenty.

Tento zpracovaný formát je pak ostatním komponentám zpřístupněn pomocí *Context API*²⁵, což poskytuje pohodlný způsob, jak k nim přistupovat nezávisle na jejich umístění ve stromu komponent.

3.4.3 Parsování SVG a mapování sedadel

`SeatingMap` komponenta je zodpovědná za parsování příchozích SVG dat a jejich překlad do formátu, který může být použit komponentou `VirtualMap`.

Příchozí SVG data mají specifickou strukturu, kde `<g>` elementy představují skupiny sedadel, a jsou identifikovány pomocí id atributu ve formátu "`seats:xxx`", kde `xxx` je identifikátor konkrétní skupiny sedadel. Jednotlivá sedadla v rámci těchto skupin jsou reprezentována `<circle>` či `<rect>` elementy, identifikovanými jejich jedinečnými id ve formě "`seat:row+place`", kde `row` je řada sedadla a `place` je číslo místa v rámci řady. Zjednodušené znázornění této struktury je vyobrazeno v ukázce kódu 4.

²⁵<https://react.dev/learn/passing-data-deeply-with-context>

```

1 <svg width="1287" height="1115" viewBox="0 0 1287 1115" fill="#F2F4F7"
2   ↳ xmlns="http://www.w3.org/2000/svg">
3     <!-- seats -->
4     <g id="seats:back">
5       <!-- seat H+1 -->
6       <circle id="seat:H+1" cx="48" cy="785" r="50" fill="#D0D5DD"/>
7       <!-- seat I+1 -->
8       <circle id="seat:I+1" cx="48" cy="805" r="50" fill="#D0D5DD"/>
9       <!-- ... -->
10    </g>
11    <g id="seats:right">
12      <!-- ... -->
13    </g>
14    <g id="seats:left">
15      <!-- ... -->
16    </g>
17    <g id="seats:center">
18      <!-- ... -->
19    </g>
20    <!-- details -->
21    <g id="details:patio">
22      <!-- ... -->
23    </g>
24  </svg>

```

Zdrojový kód 4: Ukázka struktury SVG dat

Parsování těchto dat probíhá za pomoci knihovny `svg-parser`²⁶ a výsledkem je upravená struktura, kde je každé sedadlo (reprezentované `<circle>` či `<rect>` elementem) nahrazeno dedikovanou komponentou `<Seat />`, jejíž funkčnost je dále rozepsána v sekci 3.4.5.

3.4.4 Interaktivita s VirtualMap

Komponenta `VirtualMap` přidává interaktivitu do mapy sedadel. Tato komponenta obaluje komponentu `SeatingMap`, přebírající parsované a namapované SVG elementy jako své potomky, a zajišťuje všechny interakce s mapou sedadel.

Tato komponenta využívá knihovny `react-spring`²⁷ a `use-gesture`²⁸ k zajištění

²⁶<https://www.npmjs.com/package/svg-parser>

²⁷<https://www.npmjs.com/package/react-spring>

²⁸<https://www.npmjs.com/package/@use-gesture/react>

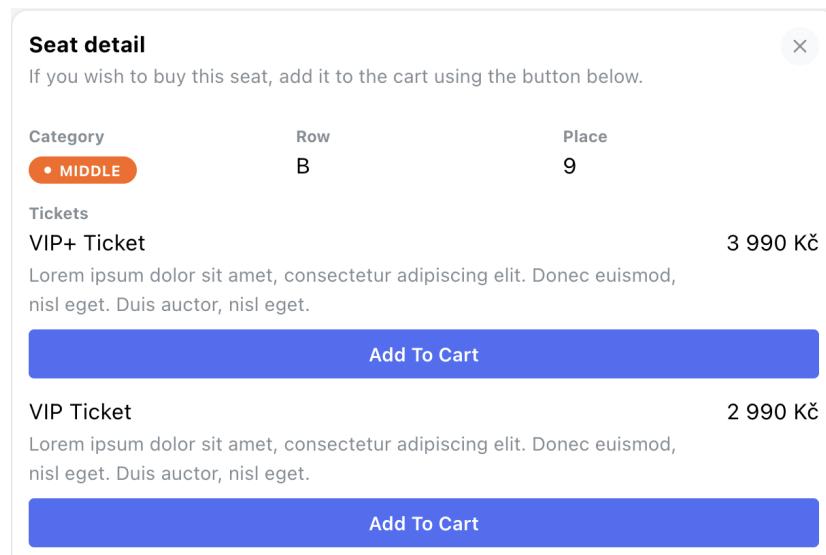
plynulé a responzivní animace pro gesta *pan*²⁹ a *pinch*³⁰, poskytující dynamický a plynulý uživatelský zážitek.

Komponenta **VirtualMap** také zajišťuje správu stavu mapy, sledováním změn, jako je úroveň *zoom*³¹ a posunutí. Tato správa stavu zajišťuje především konzistentní uživatelský zážitek při manipulaci mapy uživatelem.

3.4.5 Výběr sedadla a správa vstupenek

Komponenty **Seat** a **SeatSheet** jsou zodpovědné za výběr sedadla a správu komunikace s nákupním košíkem.

Komponenta **Seat** reprezentuje jednotlivá sedadla na mapě a spravuje všechna jeho příslušná data, jako je jeho dostupnost a aktuální stav. Kliknutím na jakékoliv sedadlo se vyvolá akce zvolení sedadla, která spustí otevření komponenty **SeatSheet**, která poskytuje uživatelsky přívětivé rozhraní pro zobrazení informací o daném sedadle a výběr z možných vstupenek.



Obrázek 3.1: Snímek obrazovky komponenty **SeatSheet**

Na obráku 3.1 je zobrazena komponenta **SeatingSheet** s detaily vybraného sedadla, včetně jeho kategorie, řady, místa a dostupných vstupenek.

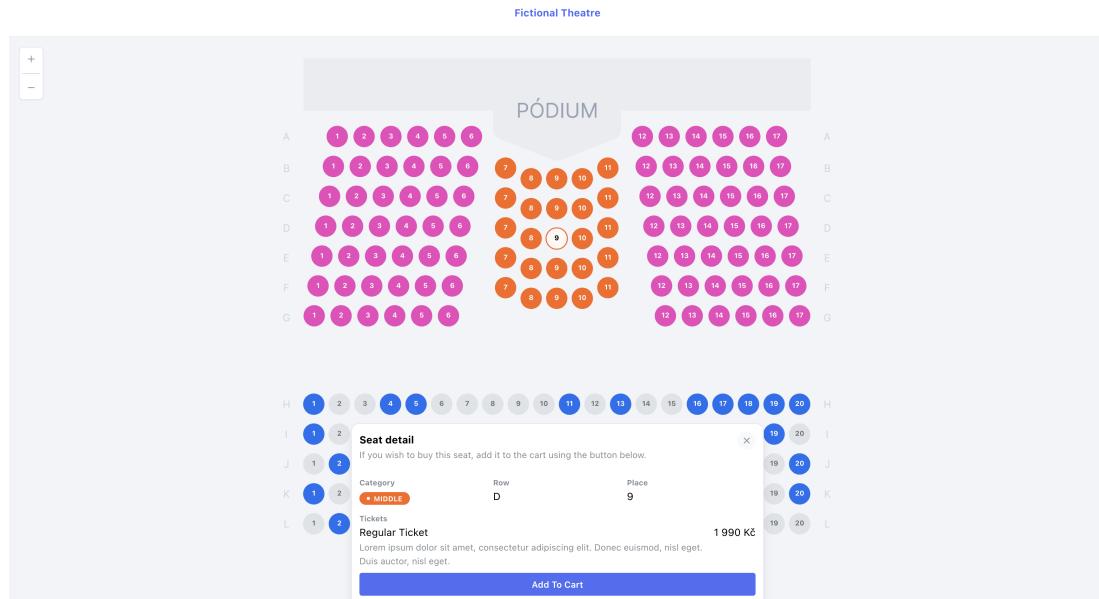
Vstupenky jsou vázány na kategorii sedadla a kategorie může mít několik příslušných vstupenek. Cena vstupenky je poté definována kombinací její kategorie a vybraného sedadla.

²⁹*Pan* je gesto, které umožňuje uživateli posunout obsah v rámci omezeného prostoru.

³⁰*Pinch* je gesto, které umožňuje uživateli přiblížit nebo oddálit obsah.

³¹*Zoom* je úroveň přiblížení nebo oddálení obsahu.

Uživatel může přidat vybranou vstupenkou do svého košíku přímo z rozhraní komponenty **SeatSheet**. Pokud je v košíku již vstupenka pro vybrané sedadlo, může uživatel vybrat, zda chce vstupenkou vyměnit nebo ji z košíku odebrat. Všechny tyto změny se okamžitě promítají v uživatelském rozhraní a poskytují tak uživateli zpětnou vazbu v reálném čase.



Obrázek 3.2: Snímek obrazovky interaktivní mapy sedadel

Obrázek 3.2 zobrazuje výsledný produkt, ilustrující uživatelsky přívětivé rozhraní a dynamickou mapu sedadel. Kombinace výše uvedených komponent umožňuje snadnou navigaci sedadel místa konání, okamžitý přístup k informacím o vstupenkách a přehlednou správu vstupenek zajišťující efektivní nákupní proces.

3.5 Správa košíku

Druhým hlavním prvkem této aplikace je správa nákupního košíku. Tento prvek umožňuje uživatelům přidávat, odebírat nebo vyměňovat vstupenky, které si vybrali z mapy sedadel. Správa košíku je pro aplikaci klíčová, jelikož slouží jako most mezi interaktivní mapou sedadel, procesem rezervace a konečné platby.

3.5.1 Kontext košíku

State, česky stav, košíku je spravován globálně v celé aplikaci pomocí kontextu. Toto zajišťuje `CartContext` komponenta, která obaluje celou aplikaci a poskytuje tak přístup a manipulaci se stavem košíku. Hlavní logika se odehrává v `useCart` hooku košíku, který vytváří instanci nákupního košíku, která obášhuje stav a nabízí dostupné akce, kterými lze stav košíku měnit.

3.5.2 Stav a funkce košíku

Z pohledu vstupenek a sedadel uchovává košík hlavně vstupenky, které jsou přidány do košíku. Tento stav je reprezentován polem objektů ve tvaru rozhraní zobrazeného v ukázce kódu 5.

```

1 /**
2  * CartedTicket type
3  * @export
4 */
5 export type CartedTicket = {
6   /** unique carted ticket id */
7   cartedTicketId: UUID;
8   /** reference to the ticket received from API */
9   ticket: VenueApiTypes.Ticket;
10  /** reference to the specific seat */
11  /** the Seat type received from API is extended with a specific category */
12  /** also tickets array from the seat type is omitted as it does not make
    → sense here */
13  seat?: Extend<
14    {
15      category: VenueApiTypes.Category;
16    },
17    Omit<VenueApiTypes.Seat, 'tickets'>
18  >;
19 };

```

Zdrojový kód 5: Rozhraní CartedTicket

Košík, jak již bylo zmíněno, nabízí akce, kterými lze stav košíku měnit. Tyto akce, reprezentované tzv. *handlery*³², zahrnují především `addToCartHandler()`, `removeFromCartHandler()` a `replaceCartedTicketHandler()`. Také poskytuje metodu `getCartedTicketPrice()`, která vrací cenu konkrétní vstupenky v košíku na základě sedadla a kategorie vstupenky, jelikož cena vstupenky je definována kombinací její kategorie a vybraného sedadla. Implementace metody `getCartedTicketPrice()` je pro ilustraci uvedena v ukázce kódu 6.

³²*Handlery* jsou v tomto případě funkce vytvořené pomocí vlastního `useHandler` hooku, který je implementován v souboru `src/lib/hooks/useHandler.ts`.

```

1  /** returns carted ticket price */
2  const getCartedTicketPrice = (cartedTicket: Types.CartedTicket) => {
3    /** ticket without seat */
4    if (!isDefined(cartedTicket.seat)) return cartedTicket.ticket.price;
5    /** get ticket category */
6    const ticketCategory = valOrThrow(
7      cartedTicket.ticket.categories.find(({ categoryId }) => categoryId ===
8        cartedTicket.seat?.categoryId),
9      `Ticket category not found for seat ${cartedTicket.seat.seatId}`,
10     );
11    /** return ticket category price */
12    return ticketCategory.price;
13  };

```

Zdrojový kód 6: Implementace metody `getCartedTicketPrice()`

V poslední řadě košík uchovává *memoizovanou*³³ hodnotu celkové ceny všech vstupenek v košíku, která je implementována jednoduše pomocí redukce pole vstupenek v košíku sečtením cen všech vstupenek prostřednictvím metody `getCartedTicketPrice()` zmíněné výše.

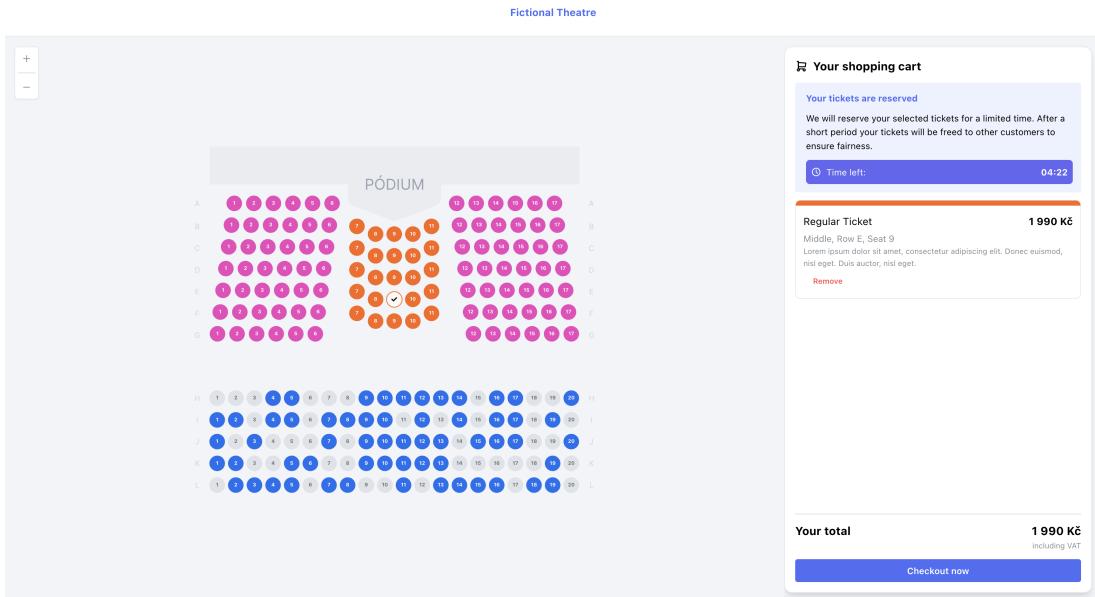
3.5.3 Vizualizace košíku

Košík je tímto již funkční, ale stále není zobrazen uživateli. Jeho zobrazení je zajištěno komponentou `Cart`, která zobrazuje stav košíku a poskytuje uživateli způsob interakce s ním. Jelikož prvním krokem celého nákupního procesu je výběr sedadla, není košík viditelný, dokud není vybráno sedadlo a vstupenka není přidána do košíku. Košík je pak zobrazen jako plovoucí prvek na pravé straně obrazovky, v případě desktopového zobrazení, a jako spodní list na mobilním zařízení, jak je navrženo v 2.5.3. Aktuální pohled na dosavadní implementaci košíku je zobrazen na obrázku 3.3.

3.5.4 Integrace API

Jelikož součástí této práce není implementace backendového systému, je stav košíku synchronizován s mock API, aby se napodobila situace v reálné aplikaci. Kdykoli uživatel provede nějakou akci v košíku, je jeho stav aktualizován s malým

³³Memoizace je optimalizační technika, která ukládá výsledky volání funkce a při jejím opakovaném volání s týmiž parametry vrací uložený výsledek místo opětovného výpočtu.



Obrázek 3.3: Snímek obrazovky košíku

zpožděním, aby se napodobila síťová latence reálného API. Toho je dosaženo voláním asynchronní funkce, která vrací *promise*, který se vyřeší po nějaké specifikované době. V tomto případě se jedná o vlastní funkci `simulatedNetworkDelay()`, která je zobrazena v ukázce kódu 7 spolu se všemi ostatními použitými funkcemi.

```

1 /**
2  * Simulates network delay
3  * @param {"normal" | "heavy"} type
4  * @return {Promise<void>}
5 */
6 export const simulatedNetworkDelay = async (type: 'normal' | 'heavy' =
7   ↪ 'normal') => {
8   const delay = type === 'normal' ? 1000 : 5000;
9   await fakePromise(DELAY_NETWORK ? randomIntBetween(delay * 0.25, delay) :
10    ↪ 0);
11 };
12 /**
13  * Resolves generated promise with optional value
14  * @param {number} delay
15  * @param {() => any} resolver
16  * @param value
17  * @returns {Promise<unknown>}
18 */
19 export const fakePromise = (delay: number, resolver?: () => any, value?: any)
20   =>
21   new Promise((resolve) =>
22     setTimeout(
23       () => {
24         resolver?.();
25         resolve(value);
26       },
27       delay,
28       value,
29     ),
30   );
31 /**
32  * Returns random integer from an interval
33  * @param {number} min
34  * @param {number} max
35  * @return {number}
36 */
37 export const randomIntBetween = (min: number, max: number): number =>
38   Math.floor(randomNumBetween(min, max));

```

Zdrojový kód 7: Implementace funkce `simulatedNetworkDelay()` a její pomocné funkce

Užití funkce `simulatedNetworkDelay()` je pak zobrazeno v ukázce kódu 8, která implementuje metodu `removeFromCartHandler`.

```

1  /** remove from cart handler */
2  const removeFromCartHandler = useHandler<Types.UseCart.RemoveFromCartCb>(
3    async (cartedTicketId) => {
4      /** clear reservation if last carted ticket */
5      if (cartedTickets.length === 1) {
6        console.log('[Cart] Removing last carted ticket, removing
7          ↵ reservation...');
8        await simulatedNetworkDelay();
9        await clearReservationHandler.handler();
10     }
11     /** TODO: API should be called instead */
12     await simulatedNetworkDelay();
13     console.log('[Cart] Updating reserved cart');
14     /** remove from cart */
15     console.log('[Cart] Removing from cart:', cartedTicketId);
16     _setCartedTickets((prev) => prev.filter(({ cartedTicketId: _cartedTicketId
17       ↵ }) => _cartedTicketId !== cartedTicketId));
18   },
19   {
20     deps: [cartedTickets],

```

Zdrojový kód 8: Ukázka kódu implementující metodu `removeFromCartHandler()`

Detailedy zmíněné výše jsou jádrem logiky košíku. Jak se může zdát, logika košíku je poměrně jednoduchá, ale je zásadní pro funkčnost aplikace. V kódu výše na řádku 9 je podmíněné volání `clearReservationHandler()`, což je první náznak integrace rezervačního mechanismu do jádra košíku. Více o rezervaci a jejím mechanismu je popsáno v následující sekci.

3.6 Rezervační systém

Rezervační systém je klíčovou součástí aplikace, jelikož se stará o rezervaci sedadel a vstupenek a také je zodpovědný za vypršení rezervace a jejího vymazání. Rezervační systém je pouze jakousi integrací, nebo spíše rozšířením jádra logiky košíku. Košík ve své podstatě pouze uchovává referenci na rezervaci ve svém stavu reprezentovanou velmi jednoduchým objektem ve tvaru rozhraní zobrazeného v ukázce kódu 9.

```
1 /**
2  * Reservation type
3 */
4 export type Reservation = {
5   reservationId: UUID;
6   reservedUntil: Date;
7 };
```

Zdrojový kód 9: Rozhraní `Reservation`

Rezervace je vytvořena v momentě, kdy je do košíku přidána první vstupenka pomocí metody `addToCartHandler()`, která je zobrazena v ukázce kódu 10.

```

1  /** add to cart handler */
2  const addToCartHandler = useHandler<Types.UseCart.AddToCartCb>(
3    async (ticket, seat) => {
4      /** create a new reservation if non-existent */
5      if (!isDefined(reservation)) {
6        console.log('[Cart] Creating reservation...');
7        await simulatedNetworkDelay();
8        /** TODO: API should be called instead */
9        const reservedUntil = seat?.place === 13 ? addSeconds(new Date(), 33) :
10          addMinutes(new Date(), 5);
11        _setReservation({ reservationId: uuidv4(), reservedUntil });
12        console.log('[Cart] Reservation created');
13      }
14      /** create a new carted ticket */
15      const newCartedTicket: Types.CartedTicket = { cartedTicketId: uuidv4(),
16        ↳ ticket, seat };
17      console.log('[Cart] Adding to cart:', newCartedTicket);
18
19      /** TODO: API should be called instead */
20      await simulatedNetworkDelay();
21      console.log('[Cart] Updating reserved cart');
22      /** add to cart */
23      _setCartedTickets([...cartedTickets, newCartedTicket]);
24      return newCartedTicket.cartedTicketId;
25    },
26    {
27      deps: [reservation, cartedTickets],
28    },
29  );

```

Zdrojový kód 10: Metoda `addToCartHandler()`

Obdobně je rezervace vymazána, když je z košíku odebrána poslední vstupenka v metodě `removeFromCartHandler()`.

3.6.1 Expirace a vymazání rezervace

Více zajímavý je proces vypršení a vymazání rezervace. Toho je docíleno pomocí vlastního hooku `useWaitUntil()`, který bere jako argumenty datum a *callback funkci*³⁴. Jednoduše čeká, dokud není dosaženo datumu, a poté zavolá tuto callback funkci, která je znázorněna v ukázce kódu 11.

³⁴Callback funkce je funkce, která je předána jako argument jiné funkci a je zavolána v rámci této funkce.

```

1 /** handle expired reservation */
2 useWaitUntil(reservation?.reservedUntil, async () => {
3   window.alert('Reservation expired, clearing cart...');
4   await clearReservationHandler.handler();
5   await options.onReservationExpired();
6 });
7
8 /* clear reservation handler */
9 const clearReservationHandler = useHandler<Types.UseCart.ClearReservationCb>(
10   async () => {
11     console.log('[Cart] Clearing reservation...');
12     await simulatedNetworkDelay();
13     _setReservation(null);
14     _setCartedTickets([]);
15   },
16   {
17     deps: [reservation],
18   },
19 );

```

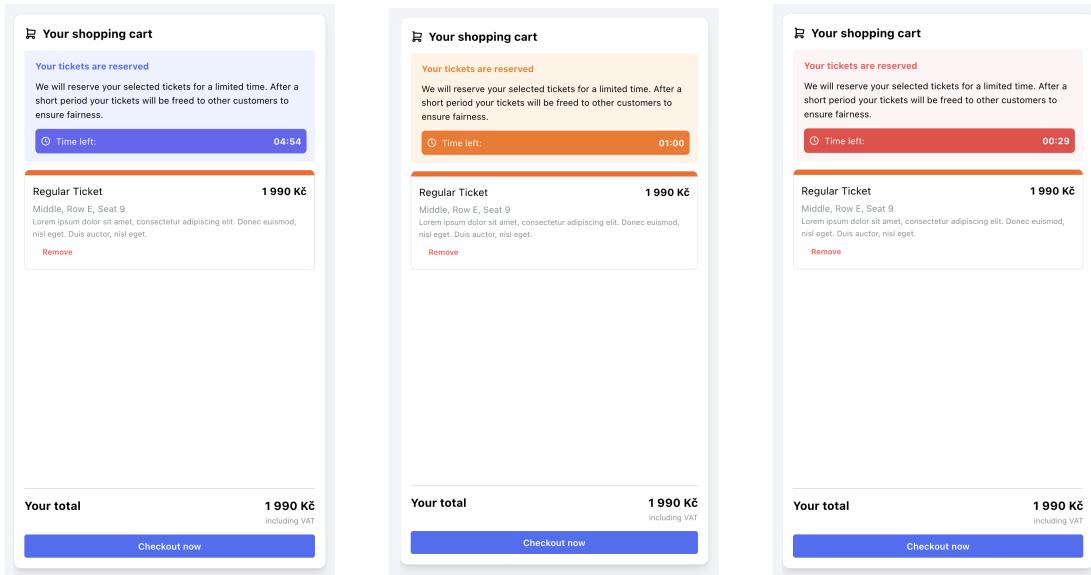
Zdrojový kód 11: Proces expirace rezervace s metodou `clearReservationHandler()`

Hlavní částí tohoto mechanismu je handler `clearReservationHandler()`, který opět jednoduše simuluje volání API a poté vymaže jak rezervaci, tak obsah košík, čímž efektivně vrací košík do výchozího stavu.

3.6.2 Vizualizace rezervace

Rezervace je vizualizována v košíku pomocí jednoduchého odpočtu, který je uživateli zobrazen. Tento odpočet je implementován s pomocí knihovny `react-countdown`³⁵, která poskytuje komponentu `Countdown`, která umožňuje vlastní vykreslování odpočtu. Komponenta přijímá datum z rezervace košíku a vykresluje upozornění na rezervaci, jenž informuje uživatele o vypršení rezervace a zbývajícím čase. Jak je znázorněno na obrázku 3.4, upozornění je zobrazeno v hlavičce košíku a liší se podle zbývajícího času do vypršení rezervace.

³⁵<https://www.npmjs.com/package/react-countdown>



Obrázek 3.4: Upozornění na rezervaci v hlavičce košíku

Po implementaci rezervačního systému je již košík plně funkční v rámci dané specifikace. Další důležitou částí celé implementace je proces dokončení objednávky, který je popsán v navazující sekci 3.7.

3.7 Vyřízení a potvrzení objednávky

Proces vyřízení objednávky je posledním hlavním krokem nákupu vstupenek, který je touto prací řešen. Je implementován jako jednoduchý formulář, který je zobrazen uživateli po naplnění košíku vstupenkami. Tento formulář a jeho správa je opět pouze dalším rozšířením logiky košíku. S pomocí hooku `useForm` z knihovny React Hook Form je implementována logika formuláře pomocí pouze několika málo řádků kódu. Formulář se skládá z několika polí reprezentovaných validačním schématem pomocí knihovny Zod v ukázce 12.

```
1 /**
2  * Cart contact details validation schema
3  * @export
4 */
5 export const useCartContactDetailsSchema = z.object({
6   firstName: z.string().nonempty(),
7   lastName: z.string().nonempty(),
8   email: z.string().email(),
9   phone: z.string().nonempty(),
10  message: z.string().optional(),
11  acceptTerms: z.boolean().refine((v) => v === true, { message: 'You must
    → accept terms and conditions' }),
12});
```

Zdrojový kód 12: Validační schéma kontaktních údajů formuláře

Formulářová políčka jsou poté vykreslena s pomocí knihovny Mantine, která kromě mnoha dalších komponent poskytuje komponentu `TextInput`, která je primárně použita pro ovládání těchto polí. Jelikož se jedná o komponentu z knihovny třetí strany, je pomocí komponenty `Controller` z knihovny `react-hook-form` připojena k formuláři, která řeší správu stavu formuláře. V ukázce kódu 13 je zobrazeno připojení komponenty `TextInput` k poli `firstName` ve formuláři.

```
1 <Controller<CartTypes.ContactDetails, 'firstName'>
2   name="firstName"
3   control={cart.contactForm.control}
4   render={({ field, fieldState }) => (
5     <TextInput label="First name" description="Enter your first name"
        → {...field} error={fieldState.error?.message} />
6   )}
7 />
```

Zdrojový kód 13: Ukázka kontrolované komponenty `TextInput`

Ostatní formulářové prvky jsou poté vykresleny obdobným způsobem.

3.7.1 Metoda platby

Kromě kontaktních údajů formulář obsahuje také výběr platební metody. I když samotný proces platby není součástí této práce, obsahuje formulář jednoduchý výběr platebních metod. Tyto metody jsou zobrazeny jako jednoduchá skupina radio tlačítek za pomoci komponenty `RadioGroup` z knihovny Mantine. Na ukázce kódu 14 je zobrazen jednoduchý číselník podporovaných platebních metod.

```
1 /**
2 * PaymentMethod enumeration
3 * @export
4 */
5 export enum PAYMENT_METHOD {
6   CREDIT_DEBIT_CARD = 'CREDIT_DEBIT_CARD',
7   APPLE_PAY = 'APPLE_PAY',
8   PAYPAL = 'PAYPAL',
9 }
```

Zdrojový kód 14: Číselník podporovaných platebních metod

Výběr podporovaných platebních metod byl zvolen na základě metod navržených v sekci 2.5.4. V reálném světě by byly metody platby implementovány například na základě integrace platební brány.

Jako mikro rozšíření obsahuje košík jednoduchý stav aktuálně vybrané platební metody, který by byl dále použit v procesu platby.

3.7.2 Souhrn objednávky

Jako požadavek na uživatelské rozhraní bylo stanoveno, že stránka dokončení objednávky musí obsahovat její souhrn. Tento souhrn je zobrazen jako karta obsahující seznam všech vstupenek v košíku. Mimo jiné slouží také jako místo pro tlačítko k dokončení objednávky, které je podmíněno platností formuláře, které zahrnuje především akceptaci obchodních podmínek. Po kliknutí na tlačítko je formulář validován oproti validačnímu schématu zobrazenému v ukázce kódu 12. Pokud validací projde, je zavolán handler `createOrderHandler()` řešící vytvoření objednávky, který simuluje volání API a poté uživatele přesměruje na stránku s potvrzením objednávky. Jeho implementace je zobrazena v ukázce kódu 15.

```

1  /** create order handler */
2 const createOrderHandler = useHandler<Types.UseCart.CreateOrderCb>(
3   async (contactDetails) => {
4     console.log('[Cart] Creating order...');
5     multiViewProvider.changeView(Types.TMultiView.ORDER_RESULT);
6     await simulatedNetworkDelay('heavy');
7     _setReservation(null);
8     _setCartedTickets([]);
9     return {
10       orderId: uuidv4(),
11       orderNumber: randomIntBetween(100000, 999999).toString(),
12       status: 'Paid',
13       created: subSeconds(new Date(), 64),
14       paid: new Date(),
15       paymentMethod,
16       amount: cartTotal,
17       tickets: cartedTickets,
18     };
19   },
20   {
21     deps: [cartedTickets, paymentMethod, cartTotal],
22   },
23 );

```

Zdrojový kód 15: Implementace handleru pro vytvoření objednávky

Jak je uvedeno v ukázce kódu 15, tento handler vytvoří falešnou novou objednávku za účelem jejího pozdějšího zobrazení na stránce s potvrzením. Souhrn objednávky spolu s již popsanými kontaktními údaji a formulářem pro výběr platební metody je zobrazeno na obrázku 3.5.

The screenshot shows a web-based ticket booking system. At the top, there's a header for "Fictional Theatre" and a timestamp "04:33". A blue banner at the top says "You are nearly at the end! We have reserved your tickets but you need to finish your order in time, or we will be forced to cancel your reservation." Below this, there are two main sections: "Personal Details" and "Order Summary".

Personal Details: Fields for First name, Last name, Email address, and Phone number. There's also a "Message" field for leaving a note.

Order Summary: Shows 1x Regular Ticket (Middle, Row F, Seat...), Subtotal 1990 Kč, Service Fees 0 Kč, and a total of 1990 Kč. It includes a checkbox for accepting terms and conditions and privacy policy, which is checked.

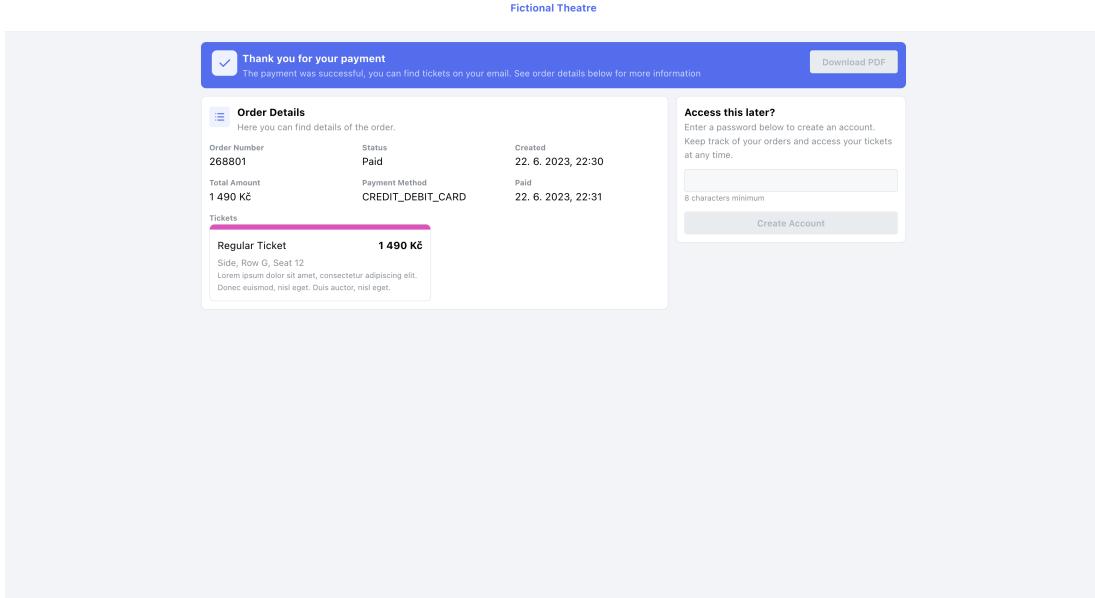
Payment Method: Options include Credit/Debit Card (selected), Apple Pay, and PayPal.

Obrázek 3.5: Potvrzení objednávky

3.7.3 Potvrzení objednávky

Jako poslední krok procesu nákupu vstupenek je implementována stránka s potvrzením objednávky. I když je tato stránka úzce propojena s procesem platby, který není součástí této práce, je implementována alespoň velmi jednoduše.

Jak je navrženo v návrhu uživatelského rozhraní, stránka s potvrzením objednávky obsahuje jednoduchou kartu s jejím přehledem a možností stáhnout si vstupenky. V rámci metody `createOrderHandler()` je vytvořena nová objednávka, která je následně zobrazena na této stránce přístupem k výsledku volání handleru `createOrderHandler`.



Obrázek 3.6: Stránka s potvrzením objednávky

Tato finální stránka, zobrazena na obrázku 3.6, je také posledním krokem implementační části vyvíjené aplikace.

3.8 Nasazení aplikace

Poslední krok vývoje tohoto prototypu je nasazení aplikace do cloudového prostředí pomocí vybrané hostingové služby. Tomuto procesu se také říká *deployment* a je to důležitý krok, který umožňuje uživatelům z celého světa přistupovat a interagovat s aplikací. Vzhledem k tomu, že aplikace byla vyvinuta pomocí Next.js, byla jako hostingová služba vybrána **Vercel** – renomovaná platforma uznávaná pro svou výjimečnou podporu Next.js aplikací[42].

3.8.1 Proces nasazení

Proces nasazení této aplikace začíná propojením Git repozitáře s novým projektem v rámci platformy Vercel, pro kterou bylo nutné vytvořit účet. Jak je v dnešní době již standardem, registrace byla rychlá a bezproblémová pomocí propojení s účtem třetí strany – v tomto případě byl použit účet GitHub. Jakmile je repozitář propojen, Vercel automaticky nastaví nasazování pro každý *push*³⁶ do zvolené produkční větve – často `main` nebo `master` větve.

V případě tohoto projektu byl nasazovací proces následující:

1. Vytvoření nového projektu na Vercel.
2. Propojení Git repozitáře obsahujícího kód aplikace.
3. Nastavení *build procesu*³⁷ aplikace, který je v tomto případě založen na příkazu `pnpm build`.

Po dokončení těchto kroků Vercel automaticky spustí build process a nasadí aplikaci vždy, když jsou do produkční větve nahrané nové změny, čímž se efektivně vytvoří Continuous Development (CD) proces – proces nasazování, který je automatický a nevyžaduje žádnou lidskou interakci.

3.8.2 Výsledek nasazení

Po úspěšném nasazení je aplikace hostována na URL, kterou poskytuje Vercel a je strukturována jako `project-name.vercel.app`.

³⁶*Push* je git příkaz, který nahraje lokální změny do vzdáleného repozitáře[43].

³⁷*Build proces* je proces, který převede zdrojový kód aplikace do spustitelné podoby.

Vyvinutá aplikace v rámci této práce je tedy nasazena a přístupná uživatelům po celém světě, hostována na URL <https://seating-map.vercel.app> skrze platformu Vercel.

Robustní funkce a integrace platformy Vercel poskytly bezproblémový přechod od vývoje k nasazení. Dodala živou, globálně přístupnou aplikaci, která je připravena k interakci uživatelů, čímž byl dosažen konečný cíl vývojového procesu.

3.9 Závěr

Tato kapitola nabídla komplexní přehled rozhodovacího procesu a strategického uvažování, které se podíleli na vývoji aplikace pro webové řešení prodeje vstupenek s rezervací míst pomocí moderních webových technologií. Objasňuje složitou cestu vytváření sofistikované frontendové aplikace od jejího počátku s pečlivými vysvětleními poskytnutými pro každou následující fázi.

Počáteční fáze zahrnovala rozhodnutí o technologickém stacku, který upřednostňuje sílu Reactu a TypeScriptu pro vytvoření škálovatelné a udržovatelné aplikace. Architektura a struktura projektu byly metodicky uspořádány podle moderních nejlepších postupů s ohledem na potenciální růst projektu.

Kapitola primárně analyzovala čtyři hlavní funkce celého řešení, konkrétně interaktivní mapu sedadel, správu košíku, rezervační systém a proces dokočení objednávky, aby bylo možné důkladně porozumět strategiím použitým při jejich implementaci. Tyto funkce se s důrazem na uživatelský zážitek snaží zpříjemnit a zjednodušit proces rezervace vstupenek.

Interaktivní mapa sedadel tvoří jádro aplikace, ilustruje dynamické použití manipulace s SVG strukturou a poskytuje uživatelům intuitivní způsob výběru sedadel. Následující sekce podrobně popisují správu nákupního košíku, rezervačního systému a procesu dokočení objednávky, které jsou všechny úzce propojeny s mapou sedadel.

Konečným výsledkem je funkční prototyp, který demonstruje všechny hlavní funkce, které byly v této kapitole analyzovány a je k dispozici na cloudové platformě Vercel.

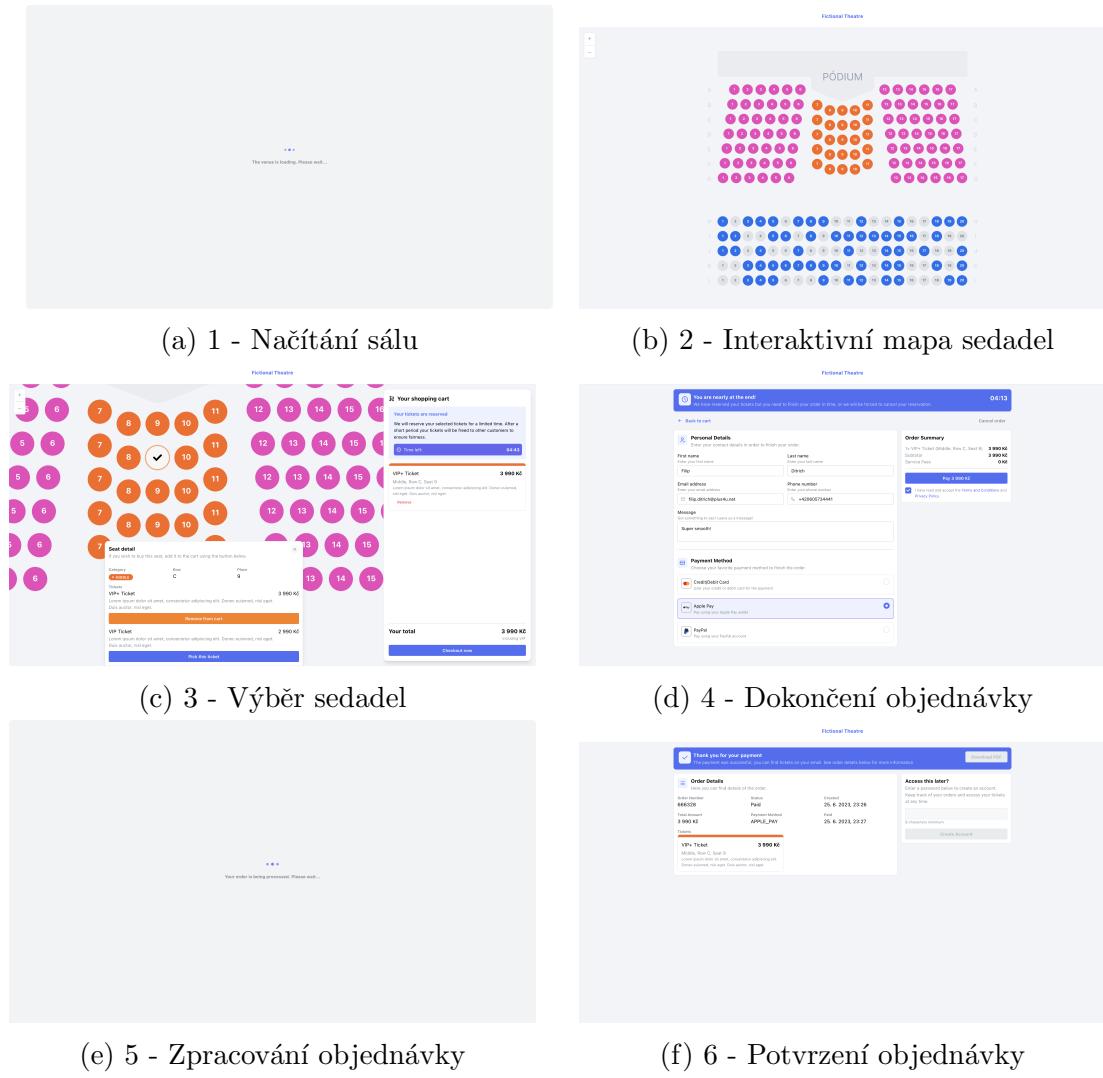
Je důležité poznamenat, že tato aplikace funguje pouze jako prototyp, který se hladce integruje se simulovaným API, a různé další praktické aspekty, jako je zabezpečení, optimalizace výkonu, důkladná správa chyb či internalizace³⁸, mohou vyžadovat další zvážení při vývoji aplikace připravené k produkci.

Souhrnně tato kapitola ilustruje, jak moderní technologie jako React a TypeScript, spolu s promyšleným plánováním, strategickým uvažováním a metodickým přístupem, mohou být úspěšně použity k vytvoření sofistikované frontendové aplikace.

Finální podoba nákupního procesu v rámci nasazené aplikace je zobrazena na

³⁸Internalizace je proces přizpůsobení aplikace pro použití v různých jazycích a regionech.

obrázích 3.7



Obrázek 3.7: Kompletní nákupní proces

Následující kapitola pojednávajá o zajímavých problémech, které se dále vyskytly během vývoje tohoto projektu, o použitých strategiích k jejich překonání a o získaných zkušenostech. Reflexe této cesty poskytuje cenné poznatky a vodítko pro budoucí projekty podobného charakteru.

4 Výzvy a problémy

Tato kapitola se zabývá zajímavými výzvami, které se vyskytly během vývoje aplikace a stojí za to je zmínit. Každá sekce popisuje konkrétní problém, který byl řešen, různé alternativy, které byly zvažovány, a konečná řešení, která byla implementována. Sdílením těchto zkušeností lze dosáhnout hlubšího porozumění vývojového procesu a složitého rozhodování, které je zapotřebí při vývoji komplexních aplikací.

4.1 Vykreslování interaktivního plánu sezení pomocí SVG

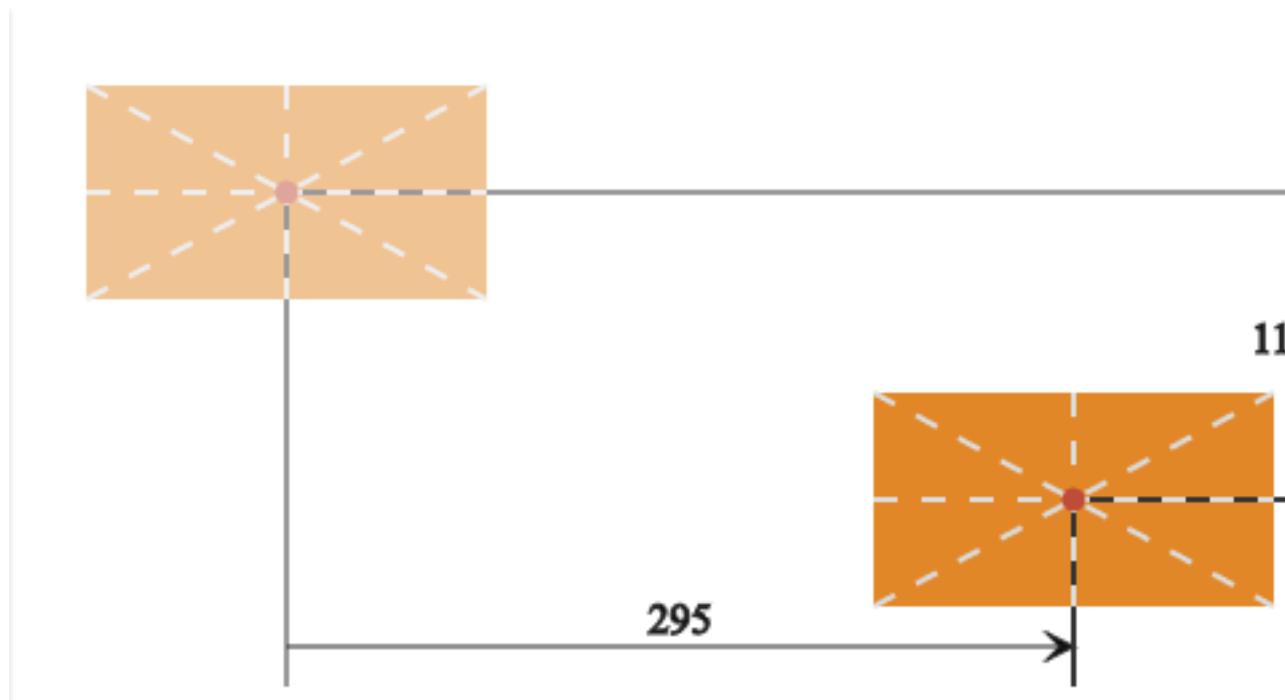
Vykreslování interaktivního plánu sezení představovalo počáteční výzvu, což bylo především způsobeno složitostí a dynamickou povahou virtuálních map. Byly zvažovány různé technologie, jako je inline SVG, *Canvas API*¹ a *Three.js*².

Po důkladném zvážení byla vybrána technologie inline SVG, jelikož poskytovala největší flexibilitu, nezávislost na externích knihovnách a nejjednodušší implementaci[46]. Tato metoda umožnila jednoduché a přímočaré interakce s DOM, což usnadnilo vývoj interaktivní mapy sedadel.

Nicméně animace a transformace SVG představovaly další problém. SVG a HTML prvky se chovají odlišně, pokud jde o jejich počáteční body, neboli tzv. *origin points*, jak je znázorněno na obrázku 4.1 níže.

¹ *Canvas API* je API pro vykreslování 2D grafiky a animací pomocí JavaScriptu a je součástí HTML5 specifikace[44].

² *Three.js* je knihovna pro JavaScript, která umožňuje vytvářet a vykreslovat 3D grafiku v prohlížeči[45].



Obrázek 4.1: Počáteční body SVG a HTML[47]

To způsobilo neočekávané chování při implementaci určitých funkcionalit transformace prvků na mapě. Tento problém byl nakonec vyřešení hlubším porozuměním transformacím SVG a jejich vztahu k HTML prvkům.

4.2 Absence API a jeho simulace

Vývoj aplikace probíhal v nepřítomnosti dedikovaného API. To znamenalo, že bylo zapotřebí definovat datové struktury a simulovat různá API volání v rámci kódu. To bylo provedeno implementací asynchronních funkcí s umělým zpožděním, které napodobovaly skutečná API volání.

Tato řešení poskytovala požadovanou funkcionalitu, ale přinášela také výzvu zajištění konzistence dat a správy potenciálních chyb, podobně jako by se očekávalo u skutečných API volání. Tato výzva byla vyřešena jednak vytvořením vlastního mock API endpointu díky Next.js API routingu[48] a jednak vytvořením vlastního API klienta, který poskytoval konzistentní rozhraní pro komunikaci s API.

4.3 Zachování jednoduchosti v komplexní aplikaci

Jednou z opakujících se výzev během vývoje byl vnitřní osobní konflikt při konstrukci komplexní aplikace produkční kvality, která by zároveň zachovávala jednoduchost a přímočarost. Například implementace komplexního routingu a správy stavu byla zvažována, ale nakonec byla zavrhnuta ve prospěch jednoduchého řešení, které bylo dostatečné pro potřeby aplikace.

K tomuto řešení byla například implementována vlastní komponenta `Multiview`, která podmíněně vykresluje zadané pohledy na základě aktivního pohledu, čímž se obešla potřeba složitého routování. Tato strategie zachovala jednoduchost aplikace, aniž by to mělo vliv na její funkčnost.

Dalším problémem bylo rozhodování, které funkce pro produkční nasazení implementovat, jako je internacionálizace, přístupnost, optimalizace pro vyhledávače - Search Engine Optimization (SEO), správa uživatelských relací či aktualizace dat v reálném čase pomocí Websockets (WS)[49]. Nakonec bylo rozhodnuto zaměřit se na základní funkčnost aplikace a implementaci těchto funkcí ponechat na možné budoucí iterace.

4.4 Shrnutí

Tato kapitola zkoumala některé z kritických výzev, které se objevily během vývoje aplikace. Od vykreslování interaktivní mapy pomocí SVG, simulace API volání, až po zachování jednoduchosti v komplexní a rostoucí aplikaci. Každá výzva přinesla své jedinečné problémy, učící příležitosti a zajímavá řešení, které ukázaly složitý proces vývoje aplikací v praxi.

Závěr

TODO: zhodnocení finálního prototypu oproti specifikacím, popis dalších možností vývoje, závěr

Literatura

- [1] Srovnání platebních bran [online]. Dostupné z:
<https://www.comgate.cz/blog/srovnani-platebnich-bran>, 2023. [cit. 2023-05-14].
- [2] SCHEJBAL&PARTNERS s.r.o. Licence platební instituce (pi) [online]. Dostupné z: <https://akschejbal.cz/platebni-instituce>, 2023. [cit. 2023-05-14].
- [3] Česká národní banka. Licencování - Česká národní banka [online]. Dostupné z: <https://www.cnb.cz/cs/dohled-financni-trh/vykon-dohledu/postaveni-dohledu/dohled-nad-uverovymi-institucemi/licencovani/>, 2023. [cit. 2023-05-14].
- [4] careerfoundry.com. Ux vs. ui design: What's the difference? [2023 guide] [online]. Dostupné z: <https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide>, 2023. [cit. 2023-06-28].
- [5] dribbble.com. resources ui-design-principles [online]. Dostupné z: <https://dribbble.com/resources/ui-design-principles>, 2023. [cit. 2023-06-28].
- [6] www.interaction design.org. What are affordances? [online]. Dostupné z: <https://www.interaction-design.org/literature/topics/affordances>, 2016. [cit. 2023-06-28].
- [7] Steven Bradley. Designing for a hierarchy of needs. Dostupné z: <https://www.smashingmagazine.com/2010/04/designing-for-a-hierarchy-of-needs/>, 4 2010. [cit. 2023-05-21].
- [8] Abraham Harold MASLOW. *Motivace a osobnost*. Portál, Praha, 2021.

- [9] Wikipedie. Potřeba [online]. Dostupné z:
<https://cs.wikipedia.org/wiki/Pot%C5%99eba>, 2023. [cit. 2023-05-21].
- [10] www.uxbooth.com. User stories: A foundation for ui design — ux booth [online]. Dostupné z: <https://www.uxbooth.com/articles/user-stories-a-foundation-for-ui-design>, 2023. [cit. 2023-06-28].
- [11] www.agilealliance.org. What does invest stand for? [online]. Dostupné z: <https://www.agilealliance.org/glossary/invest>, 2023. [cit. 2023-06-28].
- [12] www.creativebloq.com. The best ui design tools in 2023 [online]. Dostupné z: <https://www.creativebloq.com/how-to/20-best-ui-design-tools>, 2023. [cit. 2023-06-28].
- [13] Adobe Inc. Adobe xd [online]. Dostupné z:
<https://www.adobe.com/cz/products/xd.html>, 2023. [cit. 2023-05-21].
- [14] www.animaapp.com. Figma vs adobe xd vs sketch: best design tool for 2022 [online]. Dostupné z: <https://www.animaapp.com/blog/industry/the-ultimate-battle-figma-vs-sketch-vs-adobe-xd>, 2023. [cit. 2023-06-28].
- [15] Figma, Inc. Figma [online]. Dostupné z: <https://www.figma.com/>, 2023. [cit. 2023-05-21].
- [16] help.figma.com. hc en-us [online]. Dostupné z:
<https://help.figma.com/hc/en-us>, 2023. [cit. 2023-06-28].
- [17] Sketch B.V. Sketch [online]. Dostupné z: <https://www.sketch.com/>, 2023. [cit. 2023-05-21].
- [18] Mozilla Developer Network. Getting started with the web - learn web development — mdn [online]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web, 2023. [cit. 2023-06-28].
- [19] Mozilla Developer Network. Introduction to client-side frameworks - learn web development — mdn [online]. Dostupné z:
https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction, 2023. [cit. 2023-06-28].

- [20] Browser Stack. Top 5 css frameworks for developers and designers — browserstack [online]. Dostupné z: <https://www.browserstack.com/guide/top-css-frameworks>, 2023. [cit. 2023-06-28].
- [21] Mozilla Developer Network. Front-end web developer - learn web development — mdn [online]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Front-end_web_developer, 2023. [cit. 2023-06-28].
- [22] Mozilla Developer Network. Introduction to the server side - learn web development — mdn [online]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction, 2023. [cit. 2023-06-28].
- [23] Red Hat. What is open source? [online]. Dostupné z: <https://www.redhat.com/en/topics/open-source/what-is-open-source>, 2023. [cit. 2023-06-28].
- [24] www.simicart.com. Spa vs. mpa: Pros, cons how to make final choice - simicart [online]. Dostupné z: <https://www.simicart.com/blog/spa-vs-mpa>, 2023. [cit. 2023-06-28].
- [25] Facebook Open Source. React — meta open source [online]. Dostupné z: <https://opensource.fb.com/projects/react>, 2023. [cit. 2023-06-28].
- [26] React. Virtual dom and internals – react [online]. Dostupné z: <https://legacy.reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom>, 2023. [cit. 2023-06-28].
- [27] React. Describing the ui – react [online]. Dostupné z: <https://react.dev/learn/describing-the-ui>, 2023. [cit. 2023-06-28].
- [28] GeeksforGeeks. Difference between typescript and javascript - geeksforgeeks [online]. Dostupné z: <https://www.geeksforgeeks.org/difference-between-typescript-and-javascript>, 2023. [cit. 2023-06-28].
- [29] CSS Tricks. The relevance of typescript in 2022 — css-tricks [online]. Dostupné z: <https://css-tricks.com/the-relevance-of-typescript-in-2022/>, 2022. [cit. 2023-06-28].

- [30] Next.js. nextjs.org docs [online]. Dostupné z: <https://nextjs.org/docs>, 2023. [cit. 2023-06-28].
- [31] Mantine. Mantine [online]. Dostupné z: <https://mantine.dev>, 2023. [cit. 2023-06-28].
- [32] MUI. Mui: The react component library you always wanted [online]. Dostupné z: <https://mui.com>, 2023. [cit. 2023-06-28].
- [33] Tailwind CSS. Tailwind css - rapidly build modern websites without ever leaving your html. [online]. Dostupné z: <https://tailwindcss.com>, 2023. [cit. 2023-06-28].
- [34] Prettier. Prettier · opinionated code formatter [online]. Dostupné z: <https://prettier.io>, 2023. [cit. 2023-06-28].
- [35] ESLint. Find and fix problems in your javascript code - eslint - pluggable javascript linter [online]. Dostupné z: <https://eslint.org>, 2023. [cit. 2023-06-28].
- [36] nodejs.org. Node.js [online]. Dostupné z: <https://nodejs.org>, 2023. [cit. 2023-06-28].
- [37] Git. git-scm.com [online]. Dostupné z: <https://git-scm.com>, 2023. [cit. 2023-06-28].
- [38] GitHub. Build software better, together [online]. Dostupné z: <https://github.com/about>, 2023. [cit. 2023-06-28].
- [39] education.github.com. Github student developer pack [online]. Dostupné z: <https://education.github.com/pack>, 2023. [cit. 2023-06-28].
- [40] pnpm.io. Fast, disk space efficient package manager — pnpm [online]. Dostupné z: <https://pnpm.io>, 2023. [cit. 2023-06-28].
- [41] profy.dev. Popular react folder structures and screaming architecture [online]. Dostupné z: <https://profy.dev/article/react-folder-structure>, 2023. [cit. 2023-06-28].
- [42] Vercel Documentation. Vercel documentation [online]. Dostupné z: <https://vercel.com/docs>, 2023. [cit. 2023-06-28].
- [43] Git. docs git-push [online]. Dostupné z: <https://git-scm.com/docs/git-push>, 2023. [cit. 2023-06-28].

- [44] Mozilla Developer Network. Canvas api - web apis — mdn [online]. Dostupné z:
https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API, 2023. [cit. 2023-06-28].
- [45] Mozilla Developer Network. Building up a basic demo with three.js - game development — mdn [online]. Dostupné z:
https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/Building_up_a_basic_demo_with_Three.js, 2023. [cit. 2023-06-28].
- [46] seatmap.pro. Seating plans. how do we render? [online]. Dostupné z:
<https://seatmap.pro/blog/seating-plan-rendering>, 2020. [cit. 2023-06-28].
- [47] CSS Tricks. Transforms on svg elements — css-tricks [online]. Dostupné z:
<https://css-tricks.com/transforms-on-svg-elements>, 2019. [cit. 2023-06-28].
- [48] Next.js. Learn — next.js [online]. Dostupné z:
<https://nextjs.org/learn/basics/api-routes>, 2023. [cit. 2023-06-28].
- [49] Mozilla Developer Network. The websocket api (websockets) - web apis — mdn [online]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API, 2023. [cit. 2023-06-28].

Seznam obrázků

1.1	Plánek sedaček v síti TICKETPORTAL	16
1.2	Mapa barevně odlišených sedadel GoOut	18
1.3	Různé pohledy mapy v síti Ticketmaster.	19
1.4	Informace o sedadle na mapě v síti Ticketmaster.	20
1.5	Obsah nákupního košíku na portálu Ticketportal.cz	22
1.6	Nákupní košík na portálu Ticketmaster.com	23
1.7	Souhrn objednávky na portálu Ticketmaster.com	23
1.8	Rezervační mechanismus služby NFCtron Tickets na zkušební akci	24
1.9	Vyběr platební metody na GoOut.net	26
1.10	Srovnání platebních bran dle Comgate.cz [1]	27
1.11	Platební metody poskytované platebními poskytovateli dle Comgate.cz [1]	28
2.1	Maslowova hierarchie potřeb[9]	35
2.2	Hierarchie potřeb v návrhu UI dle Stevena Bradleyho[7]	36
2.3	Logo nástroje Adobe XD[13]	43
2.4	Ukázka rozhraní nástroje Adobe XD[13]	44
2.5	Logo nástroje Figma[15]	45
2.6	Ukázka rozhraní nástroje Figma[15]	46

2.7	Logo nástroje Sketch[17]	47
2.8	Ukázka rozhraní nástroje Sketch[17]	48
2.9	Návrh komponent pro vizualizaci plánu sedadel místa konání . . .	52
2.10	Návrh komponent pro výběr sedadel	53
2.11	Komponenta sedadla a její stavy	54
2.12	Komponenta detailu zvoleného sedadla	54
2.13	Návrh komponent přehledu nákupního košíku (desktopová verze)	56
2.14	Návrh komponent přehledu nákupního košíku (mobilní verze) . .	57
2.15	Návrh komponent dokončení objednávky (desktopová verze) . . .	58
2.16	Návrh komponent dokončení objednávky (mobilní verze)	59
2.17	Návrh komponent potrvzení objednávky (desktopová verze) . . .	60
2.18	Návrh komponent potrvzení objednávky (mobilní verze)	61
3.1	Snímek obrazovky komponenty SeatSheet	79
3.2	Snímek obrazovky interaktivní mapy sedadel	80
3.3	Snímek obrazovky košíku	84
3.4	Upozornění na rezervaci v hlavičce košíku	90
3.5	Potvrzení objednávky	94
3.6	Stránka s potvrzením objednávky	95
3.7	Kompletní nákupní proces	99
4.1	Počáteční body SVG a HTML[47]	101

Pozn.: Není-li uvedeno jinak, všechny obrázky jsou autorským dílem autora práce.

Seznam tabulek

2.1 Srovnání nástrojů pro návrh uživatelského rozhraní	50
--	----

Seznam zdrojových kódů

1	Vizualizace struktury projektu	72
2	Struktura komponent mapy sedadel	75
3	Struktura odpovědi API obsahující data o uspořádání sedadel . .	76
4	Ukázka struktury SVG dat	78
5	Rozhraní <code>CartedTicket</code>	82
6	Implementace metody <code>getCartedTicketPrice()</code>	83
7	Implementace funkce <code>simulatedNetworkDelay()</code> a její pomocné funkce	85
8	Ukázka kódu implementující metodu <code>removeFromCartHandler()</code> .	86
9	Rozhraní <code>Reservation</code>	87
10	Metoda <code>addToCartHandler()</code>	88
11	Proces expirace rezervace s metodou <code>clearReservationHandler()</code>	89
12	Validační schéma kontaktních údajů formuláře	91
13	Ukázka kontrolované komponenty <code>TextInput</code>	91
14	Číselník podporovaných platebních metod	92
15	Implementace handleru pro vytvoření objednávky	93

Seznam použitých zkratek

API Application Programming Interface

CD Continuous Development

CSS Cascading Style Sheets

DOM Document Object Model

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

MPA Multi Page Application

SEO Search Engine Optimization

SPA Single Page Application

SSG Static Site Generation

SSR Server Side Rendering

SVG Scalable Vector Graphics

UI User Interface

URL Uniform Resource Locator

UX User Experience

WS Websockets

XSRF Cross-Site Request Forgery

Přílohy