VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝZKUM A VÝVOJ UI/UX PRO Q ŘAZENÍ

RESEARCH AND DEVELOPMENT OF UI/UX FOR Q SORTING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VOJTĚCH PAVELKA**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

163013

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Pavelka Vojtěch**
Program:	Informační technologie
Název:	**Výzkum a vývoj UI/UX pro Q řazení**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2024/25

Zadání:

1. Seznamte se s problematikou Q řazení, zaměřte se na rozhraní pro respondenta.
2. Prostudujte existující řešení (diplomová práce M. Janů, 2023) a identifikujte jeho nedostatky a možný prostor pro vylepšení.
3. Prototypujte dílčí prvky UI, testujte je s vhodnými testery a iterativně je vylepšujte.
4. Integrujte dílčí prvky UI do aplikace, případně integrujte své řešení s řešeními dalších studentů.
5. Iterativně testujte aplikaci s vhodnými respondenty a vylepšujte ji "k dokonalosti".
6. Důkladně testujte vytvořené řešení a dobře vyhodnoťte jeho vlastnosti, přednosti a slabiny.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- JANŮ, Michal. *Webová aplikace pro podporu Q-řazení*. Online, Diplomová práce, vedoucí Adam Herout. Brno: FIT VUT, 2023.
- Elisa Păduraru, Fundamentals of Creating a Great UI/UX, Creative Tim, 2022
- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299

Při obhajobě semestrální části projektu je požadováno:
body 1 a 2, značné rozpracování bodů 3 a 4

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2024
Termín pro odevzdání:	14.5.2025
Datum schválení:	12.11.2024

Abstrakt

Cílem této bakalářské práce je výzkum, iterativní testování a následný vývoj uživatelského rozhraní aplikace pro provádění výzkumu pomocí Q–metodologie. Tato práce vychází z existujícího řešení aplikace vytvořeného v rámci diplomové práce studenta Michala Janů. Původní práce byla použitelná pro provádění výzkumů pomocí Q–metodologie, ale nebyla uživatelsky přívětivá, což může mít za následek zkreslování výsledků testování. S použitím existujícího řešení aplikace bylo prováděno iterativní testování uživatelského rozhraní s potenciálními uživateli a následné iterativní vylepšování aplikace za účelem dosažení co nejlepší použitelnosti a uživatelské přívětivosti aplikace. Výsledkem této práce je vylepšená aplikace, která uživateli poskytuje možnost jednoduchého a intuitivního řazení.

Abstract

The aim of this bachelor's thesis is the research, iterative testing, and subsequent development of a user interface for an application designed to conduct research using Q-methodology. This work builds upon an existing application developed as part of the diploma thesis of student Michal Janů. While the original work was usable for conducting research using Q-methodology, it lacked user-friendliness, which could potentially lead to skewed testing results. Utilizing the existing application, iterative user interface testing was conducted with potential users, and the application was iteratively improved to achieve the best possible usability and user-friendliness. The outcome of this thesis is an enhanced application that provides the user with the ability to perform sorting in a simple and intuitive manner.

Klíčová slova

Q-metodologie, Uživatelské rozhraní, Uživatelská přívětivost, UI/UX, Webová aplikace, Iterativní testování, Vue.js, Drag and drop, Řazení, Předřazení

Keywords

Q-methodology, User Interface, User Experience, UI/UX, Web application, Iterative testing, Vue.js, Drag and drop, Sorting, Presort

Citace

PAVELKA, Vojtěch. *Výzkum a vývoj UI/UX pro Q řazení*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Rozšířený abstrakt

V oblasti společenských věd a psychologie existuje mnoho metod pro zkoumání subjektivních postojů jednotlivců či skupiny lidí. Jednou z nich je Q–metodologie, která je specifická tím, že kombinuje kvantitativní a kvalitativní prvky výzkumu za účelem dosáhnutí výsledků, které co nejpřesněji odráží reálné postoje respondentů zapojených do výzkumu. Tato metoda je založena na principu přesouvání kartiček, na kterých jsou napsána tvrzení týkající se daného témata výzkumu. Tyto kartičky má respondent za úkol přesunout na plochu, která obsahuje kategorie reprezentující míru souhlasu či nesouhlasu s jednotlivými tvrzeními.

Kategorie jsou na této ploše seřazeny tak, že na krajích plochy se nachází kategorie reprezentující extrémní postoje, zatímco uprostřed se nachází negativní postoj k tvrzení. Každá z těchto kategorií má předem určený maximální počet polí, do kterých lze vložit tvrzení. Počet polí v jednotlivých kategoriích, do kterých může uživatel vložit tvrzení odpovídá normálnímu rozložení. To znamená, že krajní kategorie reprezentující extrémní postoje mají nejmenší počet polí pro vložení karet a kategorie s neutrálním postojem, která se nachází ve středu pole pro řazení má políček pro vložení karet nejvíce. Toto uspořádání pole pro řazení je použito z toho důvodu, aby se respondent v případě extrémního postoje vůči více tvrzením najednou musel zamyslet nad tím, které tvrzení do kategorie reprezentující extrémní postoj zařadí a které zařadí do kategorie s neutrálnějším postojem.

Prvním krokem této práce byla analýza již existujících aplikací pro provádění výzkumů na základě Q–řazení. Takovýchto aplikací existuje velké množství, ale většina z nich je zastaralá a uživatelsky nepřívětivá. Častým problémem těchto aplikací je špatná použitelnost na mobilních zařízeních. Navíc některé z aplikací používají složitý nebo zdlouhavý způsob řazení karet do tabulky s kategoriemi. Hlavním cílem při průzkumu těchto existujících aplikací bylo zjistit, které prvky a funkcionality UI/UX jsou uživatelsky nepřívětivé a vyvarovat se těmto problémům při vývoji aplikace pro Q–metodologii.

Jednou z těchto aplikací byla i aplikace pro provádění řazení pomocí Q–metodologie od Michala Janů, který ji vytvořil v rámci své diplomové práce. Mým úkolem bylo na tuto práci navázat a vylepšit její uživatelskou přívětivost a použitelnost.

Druhým krokem bylo provádění iterativní testování aplikace s potenciálními uživateli a následný návrh a implementace změn, které by mohly pomoct ke zvýšení použitelnosti. Součástí tohoto kroku byl také proveden průzkum již existujících aplikací, které používají nějaké zajímavé prvky a funkcionality uživatelského rozhraní z důvodu inspirace pro případnou implementaci těchto rozšíření do aplikace pro Q–metodologii. Po implementaci vylepšení do aplikace bylo vždy provedeno další kolo iterativního testování, aby se zjistilo, jestli změna pomohla k vylepšení uživatelského rozhraní a uživatelské přívětivosti.

Návrhy prvků uživatelského rozhraní byly vytvořeny vždy přímo ve webové aplikaci, aby bylo možno hned po implementaci nové vylepšení testovat. Pro vývoj aplikace byl použit moderní JavaScriptový framework určený pro tvorbu webových aplikací Vue 3. Pro implementaci funkcionalit uživatelského rozhraní byly použity knihovny tailwind css, vue-dragscroll, vue-draggable-plus. Spouštění aplikace je realizováno pomocí nástroje docker.

Kromě vylepšování uživatelského rozhraní aplikace vznikl také požadavek na napojení na rozhraní aplikace Q-panel pro získávání dat pro řazení ukládání postupu řazení a také na napojení na websocketový server, který umožňuje sdílení průběhu řazení mezi více zařízeními.

Výsledkem práce je webová aplikace, která umožňuje intuitivní a snadné řazení pomocí Q–metodologie.

Výzkum a vývoj UI/UX pro Q řazení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing Adama Herouta Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Pavelka
12. května 2025

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu bakalářské práce panu Ing. Adamu Heroutovi Ph.D., který mi po celou dobu poskytoval odborné konzultace k práci a sděloval mi velmi hodnotné poznatky a informace, které mi pomohly tuto práci vypracovat. Dále bych chtěl poděkovat Ing. Michalu Kapinusovi Ph.D. a Ing. Zdeňku Maternovi Ph.D. za poskytnutí zpětné vazby k prezentaci této práce v rámci předmětu ITT.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá výzkumem, iterativním testováním a následným vývojem uživatelského rozhraní aplikace určené pro provádění řazení pomocí Q-metodologie [30]. Vychází z již existujícího řešení, které vytvořil Ing. Michal Janů [11] v rámci své diplomové práce. Cílem této práce je nejen vylepšení funkcionality aplikace, ale i optimalizace její použitelnosti a přívětivosti na základě reálných zpětných vazeb od uživatelů. V průběhu vývoje jsem se zaměřil na provádění iterativního testování, při kterém jsem pravidelně vyhodnocoval výsledky a implementoval návrhy změn za účelem dosažení co nejlepší použitelnosti a celkové uživatelské přívětivosti aplikace.

Před tím, než jsem se vůbec mohl pustit do výzkumu a vývoje, bylo potřeba zjistit, co to Q-metodologie vůbec je a jakým způsobem se výzkum pomocí této metody provádí a také prozkoumat, jaké jsou již existující, dostupná řešení, analyzovat je a určit jejich silné a slabé stránky. To vše je podrobně popsáno v kapitole 2. Vzhledem k tomu, že mým úkolem bylo pracovat s rozpracovaným projektem, musel jsem také zjistit, které technologie byly k tvorbě řešení použity, naučit se s použitými technologiemi pracovat a také určit, které technologie bude potřeba k vylepšení řešení do aplikace přidat. Seznam a popis všech použitých technologií v aplikaci je v kapitole 3. V následující kapitole 4 je popsán princip iterativního testování aplikace podle Steva Kruga a také proces iterativního testování aplikace a následuje popis návrhu komponent a změn na základě výsledků testování popsaný v kapitole 5. Popis implementace změn v aplikaci včetně popsaných ukázek kódu je v kapitole 6. Nakonec jsem v kapitole 7 shrnul výsledky své práce a navrhl možné další kroky k pokračování vývoje aplikace.

Kapitola 2

Úvod do Q-metodologie

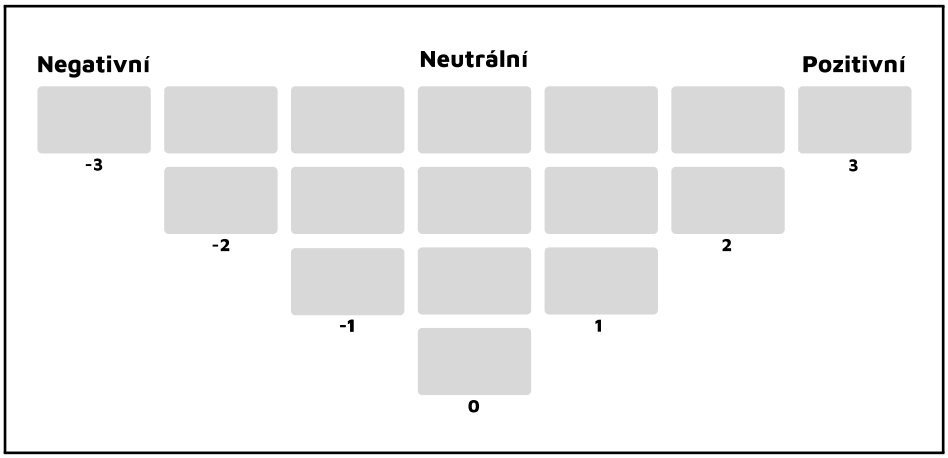
Q-metodologie [30] (někdy také označována jako Q-řazení) je výzkumná metoda, která se zaměřuje na studium subjektivních postojů jednotlivců nebo skupiny lidí k danému tématu. Je to technika, která kombinuje prvky kvantitativního a kvalitativního výzkumu a je často používána v oblastech psychologie, sociologie nebo politických věd. Metoda byla vyvinuta ve 30. letech 20. století americkým psychologem Williamem Stephensonem [24], když hledal způsob, jak zkombinovat prvky kvantitativního a kvalitativního výzkumu při studiu subjektivních postojů a preferencí lidí. Jeho práce vedla k vytvoření Q-metodologie, která se později stala účinným nástrojem pro zkoumání subjektivních postojů ve společnosti.

2.1 Popis procesu řazení u Q-metodologie

V rámci výzkumu pomocí Q-metodologie je respondentovi položena otázka například „Jak moc souhlasíte s tímto tvrzením?“, a následně mu je poskytnut balíček karet s tvrzeními, která slouží jako možné odpovědi na položenou otázku, například: „Rád se učím“. Pro umístění těchto karet má respondent k dispozici plochu (obrázek 2.1), která je rozdělena do kategorií označujících míru souhlasu, nesouhlasu a neutrálního postoje. Jednotlivá políčka na této ploše jsou organizována podle normálního rozdělení [31], což znamená, že na okrajích je políček pro umístění karet nejméně a ve středu plochy je jich nejvíce. Respondent má za úkol rozřadit karty do kategorií podle toho, jak moc s jednotlivými tvrzeními souhlasí nebo nesouhlasí. Omezením počtu karet, které lze do každé kategorie zařadit, je respondent nucen důkladněji zvážit, které tvrzení vloží do krajních kategorií označujících extrémní souhlas nebo extrémní nesouhlas. Tím je zajištěno, že odpovědi respondentů lépe odrážejí jejich skutečné postoje a musí se více zamýšlet nad tím, kterou kartu s tvrzením vloží do krajní kategorie za cenu přesunu jiné karty do mírnější kategorie.

2.2 Analýza existujících nástrojů pro Q-metodologii

Na internetu lze v dnešní době najít velké množství aplikací určených pro provádění výzkumů pomocí Q-metodologie. Problém je ale v tom, že většina aplikací je v lepším případě nevzhledná, nebo uživatelsky nepřívětivá. V horším případě je aplikace v podstatě nepoužitelná pro provedení kvalitního výzkumu. Uživatelská nepřívětivost může značně ovlivňovat výsledky výzkumu, jelikož se respondent v případě nepřívětivé aplikace musí více soustředit na to, jak s aplikací pracovat, než na samotný výzkum. Proto je v zájmu vedoucího výzkumu



Obrázek 2.1: Plocha určená pro provádění výzkumu založeného na principu Q-metodologie. Jednotlivé kategorie jsou označené hodnotami -3 až 3. Krajní a prostřední kategorie jsou navíc označeny slovně „Negativní“, „Neutrální“ a „Pozitivní“. Do negativní a pozitivní kategorie lze vložit pouze jednu kartu, zatímco do neutrální kategorie lze vložit čtyři karty.

používat takovou aplikaci, která bude pro respondenta co nejvíce přehledná, jednoduchá a intuitivní, a s pomocí této aplikace bude možné získat co nejpřesnější výsledky výzkumu.

2.2.1 Q-Assessor

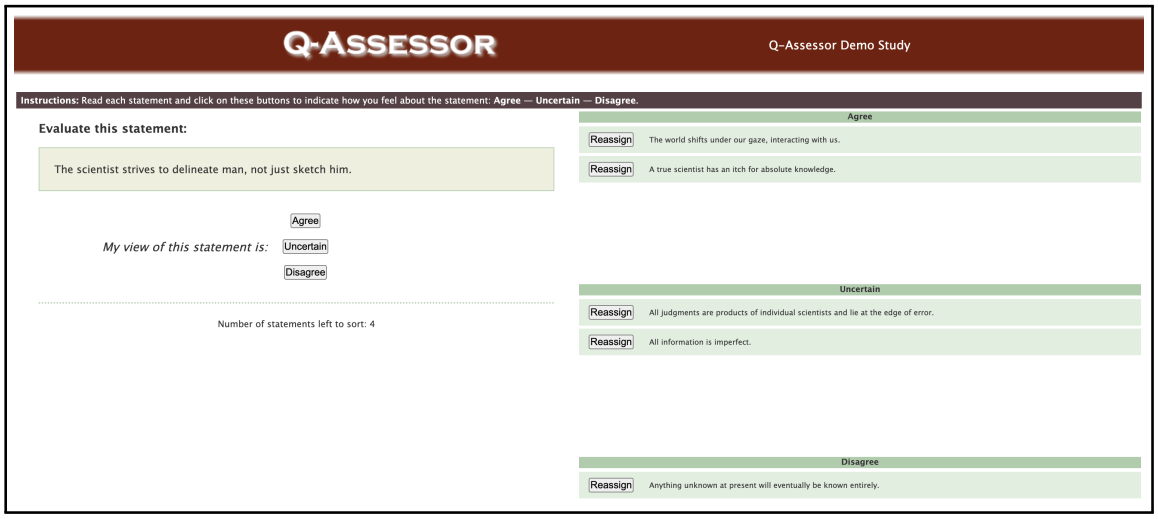
Q-Assessor[1] je webová aplikace určená pro provádění výzkumů založených na principu Q-metodologie. Aplikace nabízí dvě možnosti uživatelského rozhraní pro řazení karet do kategorií, a to standardní verzi a drag and drop verzi. Oba tyto způsoby řazení se skládají z povinné fáze předřazení tvrzení do tří hlavních kategorií (**negativní**, **neutrální**, **pozitivní**) a poté z fáze samotného řazení, při kterém uživatel používá předřazená tvrzení z prvního kroku. Aplikace je založena na předplatitelském modelu. Nabízí však možnost použití demo výzkumu pro vyzkoušení aplikace pro vytváření výzkumu a následně i samotné řazení.

Standardní verze aplikace Q-Assessor

Standardní verze aplikace je první ze dvou možností řazení, kterou aplikace Q-Assessor nabízí. V první fázi předřazení (obrázek 2.2) je uživateli v levé části obrazovky nabídnuto tvrzení. Toto tvrzení musí pomocí jednoho ze tří tlačítek předřadit do jedné ze tří hlavních kategorií. Po kliknutí na tlačítko se tvrzení přesune do vybrané kategorie v pravé části obrazovky a respondentovi se nabídne další tvrzení k zařazení, pokud fronta není prázdná. Poté, co respondent rozřadí všechna tvrzení z fronty, je mu umožněn přesun k hlavnímu řazení.

Ve fázi hlavního řazení (obrázek 2.3) respondent dostane k dispozici v pravé části obrazovky předřazené karty do tří kategorií z předchozího kroku. Přesun karty do kategorie probíhá tak, že respondent pomocí tlačítka *Select* u jednoho tvrzení vybere tvrzení jako aktivní a poté klikne v levé části aplikace na tlačítko *Move Here* u kategorie, do které chce tvrzení přesunout. Aplikace navíc nutí respondenta vyplnit ze začátku nejprve kategorie na

[1] Odkaz na aplikaci Q-Assessor: https://q-assessor.com/



Obrázek 2.2: Proces předřazení pomocí tlačítek aplikace Q-Assessor. V levé částí obrazovky se zobrazuje vždy jedno tvrzení, které uživatel zařadí do jedné z kategorií v pravé části obrazovky pomocí příslušného tlačítka u tvrzení. Tvrzení se po zařazení zobrazí v pravé části obrazovky u vybrané kategorie a v levé části se zobrazí další tvrzení.

krajích, které představují extrémní postoje, a až poté může tvrzení přesouvat do ostatních kategorií.

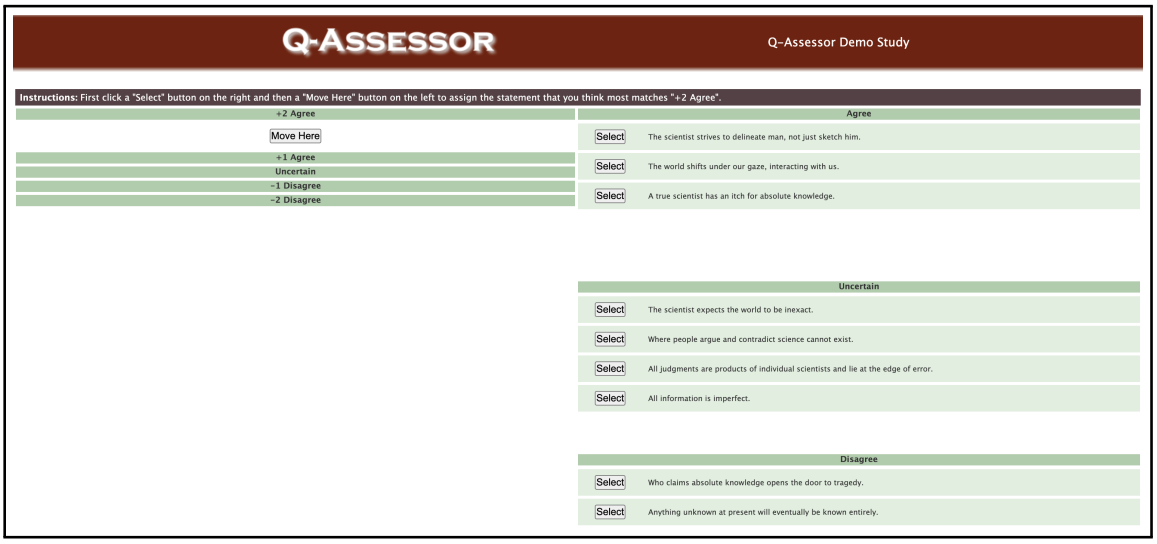
Drag and drop verze aplikace Q-Assessor

Druhou možností řazení, které aplikace Q-Assessor nabízí, je drag and drop řazení. Stejně jako ve standardní verzi aplikace je tento proces složen ze dvou fází: předřazení a hlavního řazení. Způsob, kterým se ale přesouvání položek provádí, je oproti standardní verzi jiný a uživatelsky přívětivější. Ve fázi předřazení (obrázek 2.4) je uživateli ve vrchní části obrazovky nabídnuta ve světle šedém obdélníku seznam s tvrzeními, které je potřeba rozřadit do kategorií, na které ukazují šedé šipky. Přesun tvrzení probíhá přetažením bílého prvku s tvrzením do oblasti kategorie pomocí drag and drop funkcionality. Po přesunu prvku nad jednu z kategorií a následném uvolnění levého tlačítka myši nebo prstu na mobilním zařízení se tvrzení přidá do kategorie, nad kterou se prvek zrovna nachází. Poté, co respondent všechna tvrzení předřadí do kategorií, je mu umožněn přesun k hlavnímu řazení.

Po přesunu na obrazovku hlavního řazení (obrázek 2.5) se respondentovi ve vrchní části obrazovky opět zobrazí tři kategorie s tvrzeními předřazenými tak, jak je předřadil v předchozím kroku. Pod těmito kategoriemi se nachází plocha určená pro rozřazení tvrzení do kategorií. Respondent tvrzení do dané kategorie zařadí opět tak, že ho tažením myši nebo prstu na mobilním zařízení přesune nad bílou oblast dané kategorie a uvolní. Stejně jako ve standardní verzi aplikace je respondent nucen vyplnit nejprve kategorie s extrémním postojem, aby mohl přesouvat tvrzení s mírnějším postojem k tvrzením.

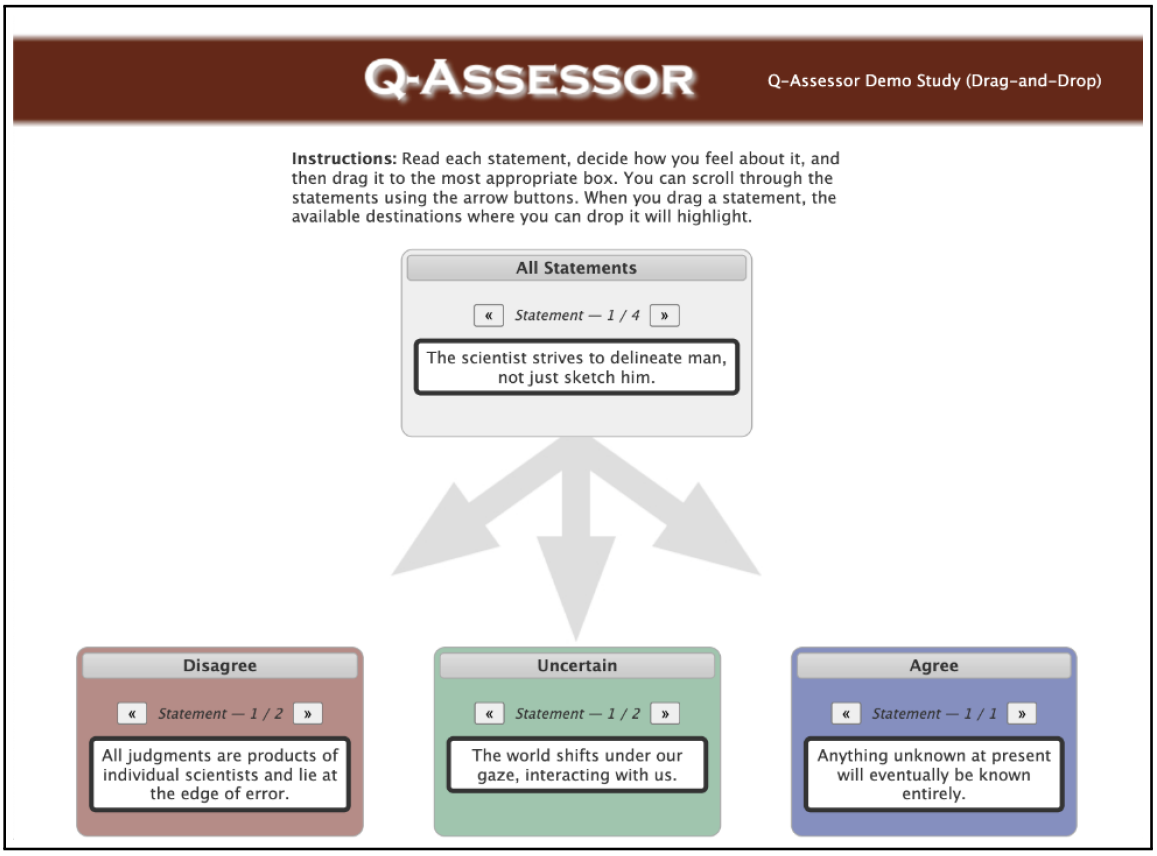
Nedostatky aplikace Q-Assessor

- Po přiřazení více tvrzení do jedné kategorie jde vidět pouze tvrzení, které bylo vloženo jako poslední. Pro zobrazení ostatních tvrzení zařazených do kategorie je potřeba překliknout pomocí tlačítka s šipkou u kategorie.



Obrázek 2.3: Proces řazení standardní verze aplikace Q-Assessor. V pravé části okna jsou předřazené tvrzení do tří kategorií (*Agree*, *Uncertain*, *Disagree*), které si uživatel předřadil v předchozím kroku. V levé části okna jsou všechny kategorie, do kterých lze tvrzení přesunout.

- Přesun tvrzení u drag and drop varianty není nijak animovaný, ale pouze v moment puštění tvrzení nad kategorií zmizí z původní pozice a přidá se na pozici novou.
- Standardní verze řazení je nepřehledná a neintuitivní.
- Aplikace nutí uživatele vždy tvrzení přesouvat nejprve do krajních kategorií a poté mu až povolí přesun do ostatních kategorií.
- V drag and drop verzi aplikace jsou v ploše pro řazení prvky obsahující přesunutá tvrzení velmi nečitelné.
- Proces předřazení je povinný. Uživatel ho nemůže přeskočit a je nucen předřadit všechny karty, než je mu povoleno přejít k hlavnímu řazení.
- Ve standardní verzi nejde přesouvat tvrzení z jedné kategorie do druhé. Pokud chce uživatel změnit své rozhodnutí, musí nejprve tvrzení vrátit zpět do fronty a poté znovu vybrat pomocí tlačítka kategorii, kam chce tvrzení nově zařadit.



Obrázek 2.4: Proces předřazení drag and drop verze aplikace Q-Assessor. Světle šedá sekce reprezentuje frontu tvrzení připravených k zařazení do jedné ze tří nabízených kategorií pod frontou.

2.2.2 QMethod Software

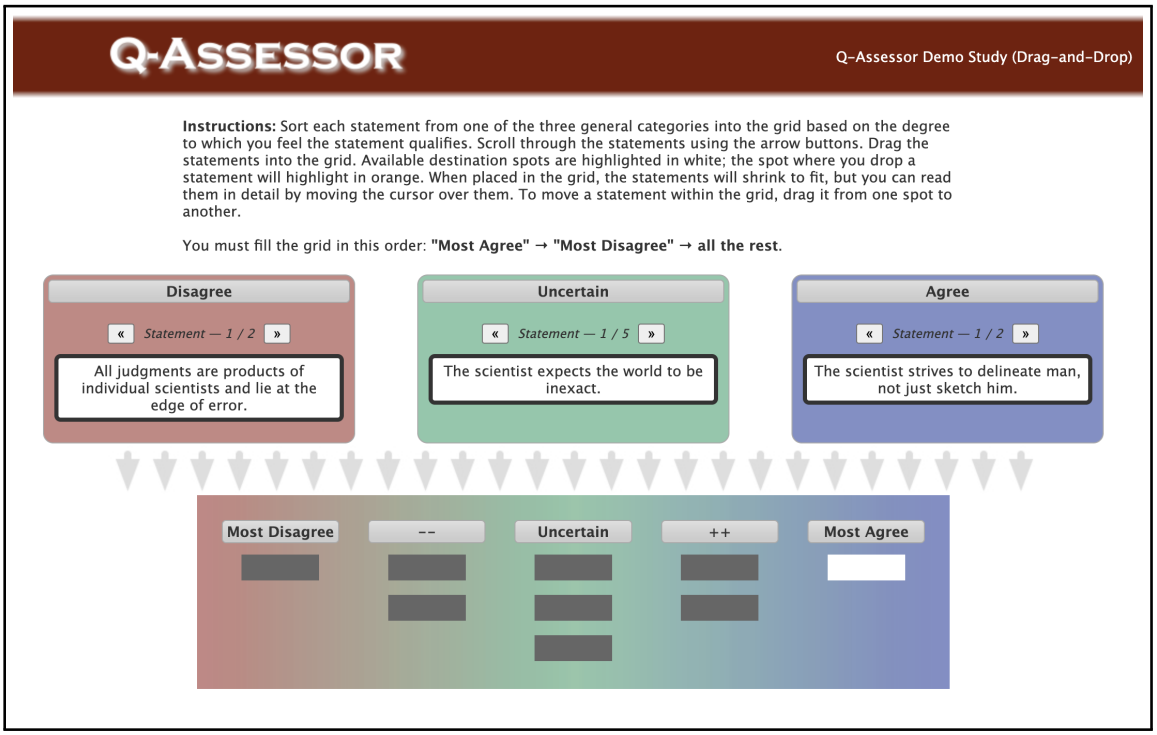
QMethod Software[2] je další z webových aplikací určených pro výzkumy založené na Q-metodologii. Tato aplikace je založena na předplatitelském modelu umožňujícím tři stupně předplatného. V bezplatné verzi aplikace nabízí možnost vytvoření jednoho výzkumu s maximálně deseti účastníky a maximálně deseti tvrzeními pro výzkum. Jiné funkce, jako analýza výsledků v reálném čase, správa účastníků, nebo možnost úpravy struktury plochy pro řazení, jsou v bezplatné verzi také k dispozici.

Vytvoření výzkumu v aplikaci QMethod Software

Vytvoření výzkumu je velice intuitivní a aplikace hezky provede uživatele celým procesem tvorby. Proces vytvoření se skládá z následujících tří kroků:

1. Vyplnění obecných informací o výzkumu, jako je název výzkumu, období platnosti výzkumu, popis výzkumu atd.
2. Vyplnění seznamu tvrzení pro výzkum
3. Nastavení rozložení plochy pro řazení

[2] Aplikace QMethod Software https://qmethodsoftware.com/



Obrázek 2.5: Proces hlavního řazení drag and drop verze aplikace Q-Assessor. Pod instrukcemi k řazení se nachází tři kategorie, do kterých respondent v předchozí fázi předřazení přesunul nabízená tvrzení. Pod těmito kategoriemi se nachází samotná plocha pro řazení pomocí Q-metodologie, do které má respondent za úkol zařadit všechna tvrzení.

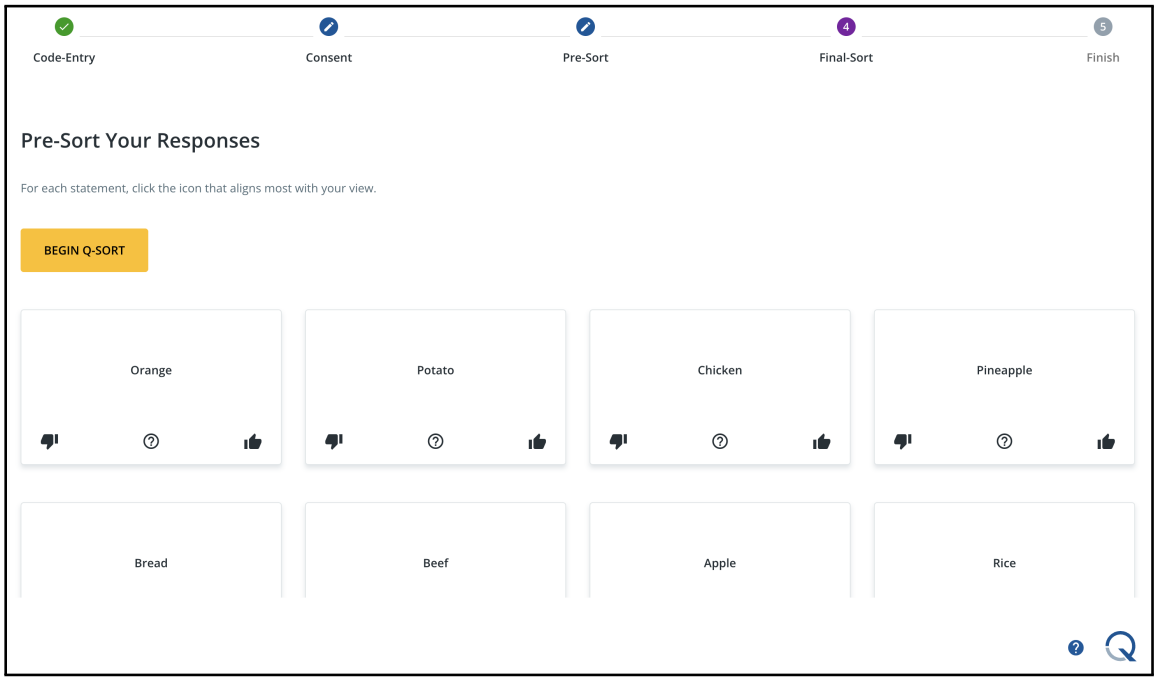
Po vytvoření výzkumu je vedoucímu výzkumu poskytnut odkaz a má možnost vygenerovat si (v závislosti na předplatném) jednorázové přístupové kódy pro účastníky výzkumu a poté jím tyto kódy zaslat e-mailem. V seznamu existujících výzkumů má vedoucí výzkumu možnost vytvořený výzkum ještě upravit. Účastníkovi pak už stačí pouze kliknout na odkaz, který mu přijde do e-mailu, zadat svůj jednorázový kód a může začít s řazením.

Proces řazení v aplikaci QMethod Software

Poté co účastník výzkumu klikne na poskytnutý odkaz a zadá svůj unikátní kód, zobrazí se mu stránka s podmínkami výzkumu, které musí před zahájením výzkumu potvrdit. Podmínky lze nastavit dodatečně v detailu výzkumu. Pokud respondent podmínky odsouhlasí, je přesunut na obrazovku určenou pro předřazení tvrzení (obrázek 2.6).

Na obrazovce pro předřazení se nachází tabulka se všemi tvrzeními a u každého z nich jsou tři tlačítka reprezentující jednotlivé kategorie pro předřazení. Ikona palce dolů znamená negativní postoj k tvrzení, ikona otazníku znamená neutrální postoj k tvrzení a ikona palce nahoru znamená pozitivní postoj k tvrzení. Po kliknutí na jednu z těchto ikonek u tvrzení je toto tvrzení přesunuto do příslušné kategorie a z původní tabulky tvrzení zmizí. Proces předřazení není povinný. Pokud se respondent rozhodne některé z tvrzení nepředřadit, je automaticky předřazeno do neutrální kategorie.

Pokud je respondent spokojen se svým předřazením, je mu umožněn přesun na stránku poslední fáze výzkumu, což je hlavní řazení (obrázek 2.7). Ve vrchní části obrazovky se nachází tři kategorie zabarvené odstíny šedé barvy, obsahující předřazená tvrzení z před-



Obrázek 2.6: Obrazovka pro předřazení tvrzení aplikace QMethod Software. Tvrzení mohou být předřazena do tří hlavních kategorií pomocí tlačítek zobrazujících palec nahoru (pozitivní postoj), otazník (neutrální postoj) a palec dolů (negativní postoj).

chozího kroku. Po kliknutí na tlačítko oboustranné šipky v pravém horním rohu kategorie se daná kategorie rozbalí a zobrazí se všechny karty v této kategorii. Přesun tvrzení z fronty do kategorie probíhá pomocí drag and drop. Po přesunutí tvrzení do pole v ploše pro řazení se místo, do kterého bylo tvrzení vloženo, na chvíli rozsvítí žlutou barvou, aby bylo uživateli jasné, že došlo ke změně v ploše. V případě přesunutí tvrzení z fronty do políčka, ve kterém se už nějaké jiné tvrzení nacházelo, dojde k přesunu původního tvrzení zpět do fronty.

Výhody aplikace QMethod Software

- Hezké a moderní uživatelské rozhraní.
- Příjemná a intuitivní drag and drop funkcionalita při řazení.
- Mapa kroků ve vrchní části obrazovky umožňuje respondentovi vyznat se ve vícekrokovém systému aplikace.

2.2.3 Nevýhody aplikace QMethod Software

- Při automatickém přesunu tvrzení z políčka z důvodu nahrazení novým tvrzením dojde vždy k přesunu do levé kategorie. To může být pro respondenta matoucí, jelikož není respektováno jeho rozřazení z procesu předřazení.

Obrázek 2.7: Obrazovka pro provádění hlavního řazení v aplikaci QMethod Software. Ve vrchní části aplikace se nachází tři kategorie obsahující tvrzení, která uživatel rozřadil v rámci procesu předřazení. Pod těmito kategoriemi se nachází samotné plocha pro řazení. Jednotlivé kategorie jsou barevně odlišeny odstíny šedé barvy.

2.2.4 HTMLQ

HTMLQ[3] je nástroj pro provádění řazení pomocí Q–metodologie. Tato aplikace, na rozdíl od ostatních zmíněných aplikací, není dostupná z internetu, ale je potřeba ji stáhnout a zprovoznit lokálně pomocí webového serveru Apache [6]. Pro vytvoření a konfiguraci výzkumu tato aplikace neobsahuje žádné uživatelské rozhraní. Nastavení výzkumů probíhá pomocí konfiguračních XML souborů uložených přímo v adresáři aplikace. Proces řazení se v této aplikaci skládá z následujících pěti fází:

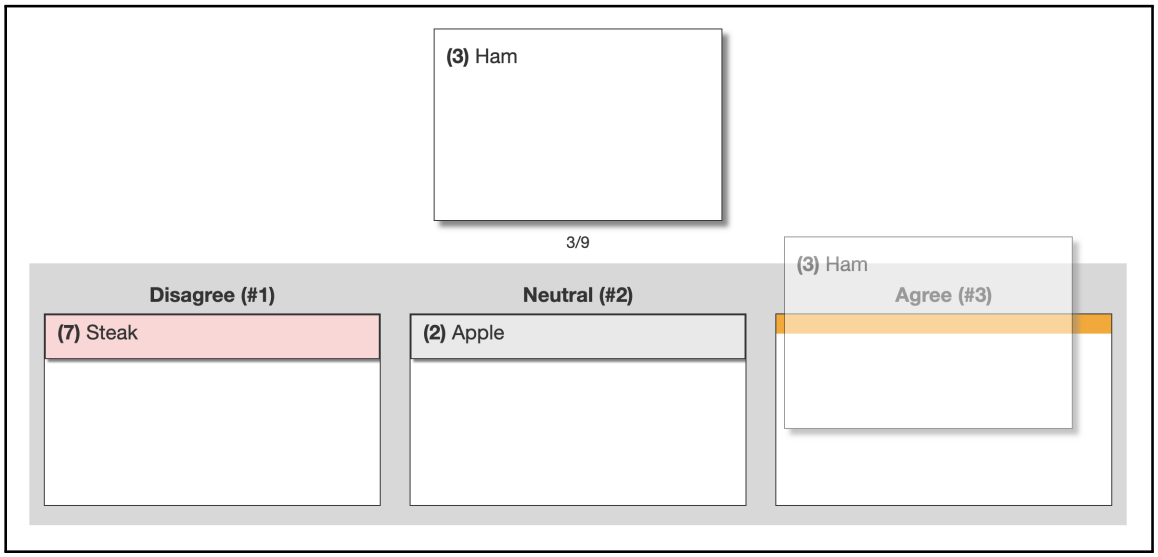
1. Předřazení fronty tvrzení (povinný krok)
2. Řazení tvrzení do tabulky s kategoriemi (povinný krok)
3. Kontrola a případné upravení výsledku řazení (nepovinný krok)
4. Vyjádření k volbě tvrzení zařazených do krajních kategorií (nepovinný krok)
5. Dotazník pro získání informací o respondentovi (nepovinný krok)

Proces předřazení

Po otevření aplikace se jako první uživateli zobrazí plocha pro předřazení tvrzení ve frontě (obrázek 2.8).

Ve vrchní části obrazovky se nachází fronta neseřazených tvrzení. Pod touto frontou jsou tři hlavní kategorie (Disagree, Neutral, Agree). Kategorie nejsou samy o sobě nijak

[3]Odkaz na GitHub repozitář aplikace HTMLQ: https://github.com/aproxima/htmlq



Obrázek 2.8: Obrazovka pro předřazení tvrzení aplikace HTMLQ. Ve vrchní části obrazovky se nachází fronta tvrzení k rozřazení. Ve spodní části jsou tři hlavní kategorie, do kterých může respondent tvrzení přesunout. Přesun probíhá pomocí drag and drop funkcionality.

barevně rozlišeny. Barevné rozlišení se zobrazí až po přiřazení tvrzení do jedné z kategorií přímo na elementu tvrzení. To tak může snižovat uživatelskou přívětivost.

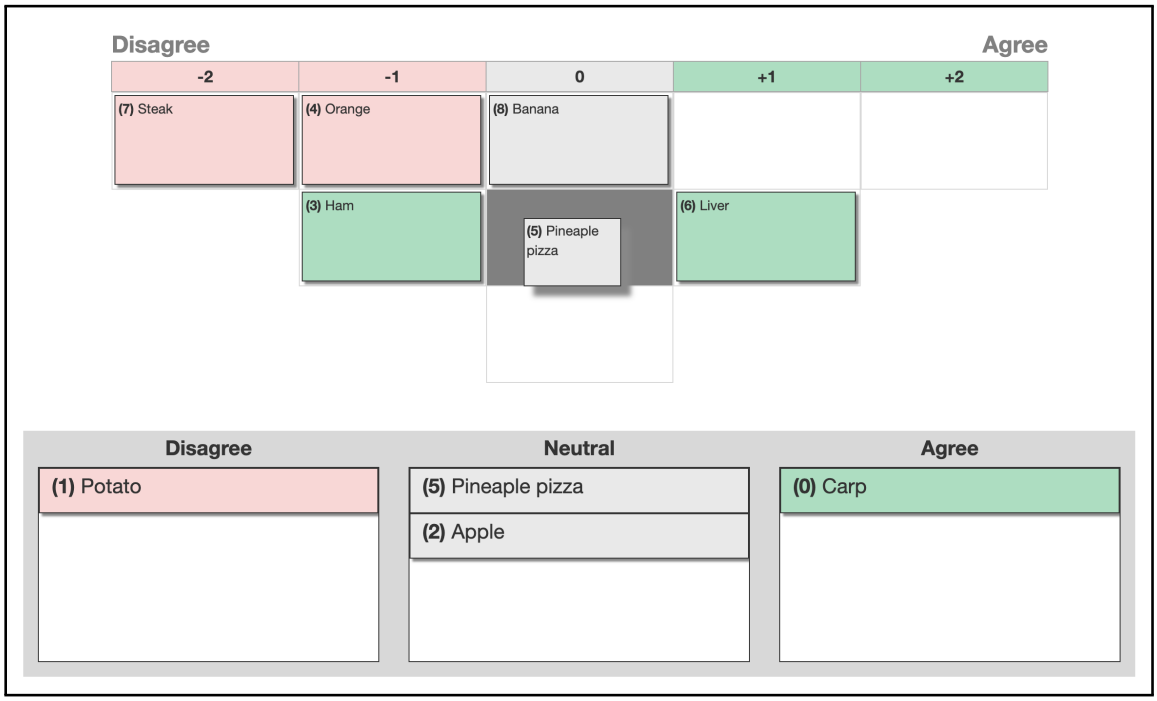
Pro přesun tvrzení do kategorií je použita drag and drop funkcionalita. Při zahájení tažení tvrzení se prvek zprůhlední, aby měl uživatel přehled o tom, že je s elementem manipulováno. Po přesunutí tvrzení nad oblast, kde může být prvek vložen, se zobrazí pouze malý oranžový obdélník značící místo vložení tvrzení, pokud ho uživatel upustí. Z fronty se přesunutý prvek odstraní až po přiřazení do některé z kategorií. To může být pro uživatele matoucí, jelikož by si mohl myslet, že se přesouvané tvrzení duplikovalo. Po přesunutí posledního tvrzení do plochy pro řazení dojde k automatickému přesunu uživatele k procesu samotného řazení. Uživatel nemá tedy možnost zkontrolovat svá rozhodnutí a případně je upravit.

Proces řazení v aplikaci HTMLQ

Obrazovka procesu řazení (obrázek 2.9) obsahuje ve vrchní části plochu se sloupci reprezentujícími jednotlivé kategorie. Do těch může respondent přesunout předřazená tvrzení z předchozího kroku, která se zobrazují ve spodní části obrazovky. Po přesunu tvrzení nad jednu z políček na ploše se pozadí políčka vybarví tmavě šedou barvou pro znázornění místa vložení tvrzení. Pokud by se uživatel na této obrazovce rozhodl, že chce změnit své rozhodnutí a přesunout tvrzení z jedné kategorie do druhé, není mu to umožněno přímo. Musí nejprve vybrané tvrzení přesunout zpět do jedné ze tří kategorií na spodku obrazovky a poté ho přesunout do nové kategorie.

Nepovinné kroky řazení

Po dokončení řazení má uživatel možnost přesunu na obrazovku rekapitulace. Na této obrazovce je uživatel upozorněn, aby si zkontroloval své předchozí rozhodnutí a případně může



Obrázek 2.9: Obrazovka pro provádění hlavního řazení. Ve vrchní části obrazovky se nachází plocha pro rozřazení tvrzení. Ve spodní části se zobrazují tvrzení předřazená do tří hlavních kategorií. Přesun tvrzení je prováděn pomocí drag and drop funkcionality.

tvrzení ještě přesunout jinam. Na této obrazovce už je uživateli umožněno přesouvat tvrzení mezi jednotlivými kategoriemi.

Pokud je uživatel se svým výsledkem řazení spokojený, může se přesunout na další obrazovku. Na této obrazovce může uživatel vysvětlit, proč při řazení (kapitola 2.9) přesunul konkrétní tvrzení právě do krajních kategorií označujících extrémní postoje. Pokud ale uživatel nechce vysvětlovat svá rozhodnutí, může tento krok přeskočit a přesunout se na poslední krok.

Posledním krokem výzkumu je osobnostní formulář. Tento formulář může být nakonfigurován v konfiguračním XML souboru aplikace.

Nedostatky aplikace HTMLQ

Po bližším zkoumání aplikace vyšly najevo tyto nedostatky:

- Nepřehlednost a špatná použitelnost uživatelského rozhraní
- Povinný krok předřazení tvrzení
- Funkcionalita drag and drop je nepřehledná a přesun prvku není nijak animovaný
- Na malých zařízeních je aplikace nepoužitelná

2.3 Verze aplikace pro Q-metodologii od Michala Janů

Součástí své diplomové práce [11] v akademickém roce 2022/2023 bývalý student inženýrského studia na FIT VUT v Brně Ing. Michal Janů vytvořil aplikaci, která je určena pro

provádění výzkumů pomocí Q-metodologie. I přesto, že aplikace splňuje požadavky pro provádění výzkumů, její používání není úplně intuitivní a chybí v ní některé funkcionality, které by mohly při provádění výzkumu ulehčit uživateli proces řazení. Toto je u aplikace pro provádění výzkumu pomocí Q-metodologie velmi důležité, protože při používání aplikace je potřeba, aby se uživatel plně soustředil na to, do kterých kategorií zařadí jednotlivá tvrzení a nemusel řešit problémy spojené se špatnou použitelností nebo uživatelskou nepřívětivostí aplikace. Pokud je totiž aplikace pro tyto účely špatně použitelná, může to značně zkreslovat výsledky výzkumu a tím se výzkum stává nepřesným.

Přesouvání tvrzení pomocí `click-select` funkcionality

Jedním z takovýchto problémů je způsob přesouvání tvrzení. Přesun tvrzení po ploše pro řazení je v této aplikaci prováděn metodou `click-select`. Proces přesunu tvrzení pomocí této metody se skládá ze dvou kroků.

V prvním kroku metody uživatel kliknutím na některé z tvrzení v poli pro řazení vybere řazení. Toto tvrzení je označeno jako aktivní a je připraveno k zařazení na plochu. Aktivní tvrzení je v této aplikaci zvýrazněno žlutým okrajem, aby měl uživatel přehled o tom, se kterým tvrzením momentálně pracuje. Poté, co má uživatel vybrané některé z tvrzení jako aktivní, má možnost kliknout na některé z volných polí v ploše pro řazení. Tím se provede druhý krok metody `click-select`, což je samotný přesun tvrzení do pole.

Pokud by chtěl uživatel vložit tvrzení z fronty na místo, ve kterém už se nějaké jiné tvrzení nachází, žádná akce se neprovede. Místo toho je mu nabízena možnost přesunu původního tvrzení z pole zpět do fronty a tím i uvolnění místa pro nové tvrzení.

Pokud uživatel nemá vybranou žádné tvrzení z plochy jako aktivní, je automaticky vybráno jako aktivní tvrzení v ploše, které je zrovna na řadě. Uživatel tak nemusí klikat na jednotlivá tvrzení ve frontě za účelem výběru, ale může rovnou kliknout na místo na ploše, kam chce tvrzení z fronty přesunout.

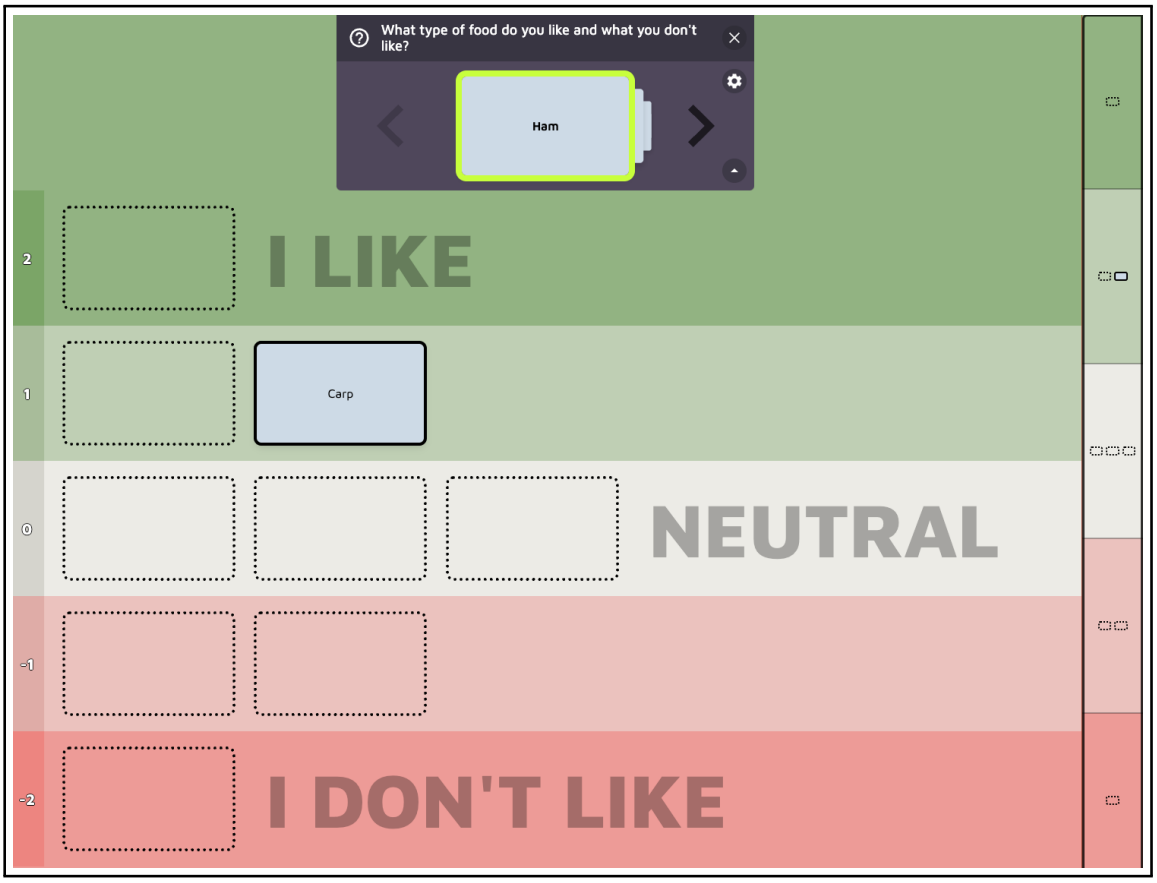
Přesun tvrzení mezi kategoriemi

Dalším problémem aplikace je přesun tvrzení v poli mezi jednotlivými kategoriemi. Pokud se uživatel během řazení rozhodne, že chce tvrzení přesunout do jiné kategorie, musí nejprve původní tvrzení z pole přesunout zpět do fronty a poté do pole přesunout nové tvrzení. Tvrzení, které uživatel vrátil do fronty, se navíc nezařadilo na začátek fronty, ale na její konec. Tím je uživatel donucen nejprve přesunout všechna tvrzení, která se nacházejí před tvrzením vráceným zpět do fronty za účelem přesunu do jiné kategorie. Může tak nastat situace, kdy uživatel nakonec musí tvrzení vložit úplně do jiné kategorie, než do které ho měl v plánu původně vložit, protože už může být kategorie plná.

Celkově je tedy proces přehodnocování rozhodnutí uživatele velice složitý a uživatelsky nepřívětivý a může ovlivňovat výsledky řazení.

Absence funkcionality předřazení

Všechny prozkoumané existující aplikace pro provádění výzkumů pomocí Q-metodologie popsané v kapitole 2.2 měly v rámci procesu řazení zabudovaný krok předřazení. Všechny tyto aplikace měly tento krok povinný. Pokud aplikace možnost předřazení nabízí, je pro respondenta proces řazení přívětivější. Předřazení totiž umožňuje uživateli roztřídit tvrzení do pomyslných hlavních kategorií pouze podle toho, jaký pocit z tvrzení má, nehledě na to, kolik tvrzení do jednotlivých kategorií může vložit.



Obrázek 2.10: Ukázka původní aplikace vytvořené studentem Michalem Janů. Ve vrchní části se nachází plocha s kartičkami s tvrzeními, které uživatel bude přesouvat do plochy v prostřední části aplikace. U pravého okraje aplikace se nachází minimapa zjednodušující orientaci v aplikaci při používání datových sad s mnoha kategoriemi.

Omezení maximálního počtu tvrzení v kategorii

Každá kategorie v aplikaci má omezený počet tvrzení, které lze do kategorie vložit. Toto omezení je v pořádku ve fázi, kdy chce uživatel dokončit řazení. V tu chvíli je totiž potřeba, aby počet tvrzení v jednotlivých kategoriích odpovídal tomu, jak je definován v ploše. Během řazení by ale mělo být umožněno uživateli dočasně vložit do kategorie více tvrzení, než je povoleno.

Toto chování totiž vychází z řazení pomocí fyzických papírových kartiček, kdy uživatelé mají tendenci si kartičky dočasně pokládat různě po ploše za účelem lepší orientace ve všech kartičkách v balíčku.

Nemožnost vložit kartu mezi kategorie

V aplikaci je plocha pevně definovaná a není možno ji nijak upravovat. Při fyzickém řazení s papírovými kartičkami ale uživatelé často vkládají kartičky mezi dvě kategorie. Dělají to převážně kvůli tomu, že v momentě, kdy se jim na řadu dostane kartička, u které se rozhodují mezi dvěma kategoriemi, chtějí toto rozhodnutí nechat až na později, až do těchto kategorií zařadí kartičky, u kterých si jsou jistí.

Kapitola 3

Technologie použité při vývoji webové aplikace pro Q-metodologii

Při vývoji aplikace byly zvoleny technologie, které odpovídají požadavkům na moderní webový vývoj a udržitelnost kódu. Některé z technologií byly použity už v původní aplikaci od Michala Janů, jako například framework pro tvorbu uživatelského rozhraní `Vue.js`, nebo nástroj pro tvorbu reaktivních úložišť `Pinia`. Nástroje jako `vue-draggable-plus`, `Tailwind CSS` nebo `TypeScript` byly do aplikace přidány za účelem implementace nových funkcionalit, zvýšení přehlednosti a udržitelnosti kódu.

3.1 Framework pro tvorbu uživatelských rozhraní: `Vue.js`

Vue.js [27] je JavaScriptový framework, který se používá pro tvorbu uživatelských rozhraní a jednostránkových aplikací. Při vývoji aplikací se lze setkat se dvěma přístupy k psaní kódu [5].

Prvním přístupem je tradiční `Options API`, který je starší a všechny části komponenty jsou zapsány v rámci jednoho objektu. Tento přístup ale u větších aplikací způsobuje nepřehlednost kódu a nepodporuje úplnou znovupoužitelnost komponent.

Jako reakce na tyto nedostatky vyšel v rámci `Vue3` (třetí verze frameworku `Vue.js`) přístup zvaný `Composition API`, který rozděluje komponenty na tři samostatné části, které jsou propojené reaktivním datovým modelem (viz sekci 3.1.1). Tento přístup značně zvyšuje přehlednost a znovupoužitelnost kódu a je doporučován při tvorbě nových `Vue.js` aplikací.

Většina komponent v aplikaci je napsaná pomocí jazyka `TypeScript` [10], který přidává do klasického `JavaScriptu` typovou kontrolu proměnných, možnost tvorby generických datových typů a mnoho dalšího.

3.1.1 Klíčové principy Vue.js

Reaktivita

Jedním z klíčových principů `Vue.js` je reaktivní datový model, který automaticky synchronizuje změny ve stavech aplikace s DOM, čímž zjednodušuje práci s dynamickými daty, zvyšuje efektivitu vykreslování a zvyšuje udržitelnost a čitelnost kódu. `Document Object Model` (zkráceně DOM) je rozhraní umožňující manipulaci s obsahem, strukturou a stylem webových dokumentů. Dokument je zde reprezentován jako strom elementů, přičemž každý element je reprezentován jako objekt.

`Vue.js` obsahuje následující pomocné funkce, které umožňují jednoduše a efektivně sledovat změny ve stavu komponent a pracovat s nimi:

- **ref** – Používá se pro vytváření jednoduchých reaktivních hodnot (například čísla nebo řetězce). Každá reaktivní hodnota je objekt s vlastností `value`, ve kterém je hodnot uložena.
- **reactive** – Používá se pro vytváření reaktivních objektů nebo polí. Je to hlavní způsob, jak pracovat se složitějšími strukturami v rámci reaktivního datového modelu.
- **computed** – Vytváří odvozené hodnoty, které se automaticky přepočítávají, pokud se změní data, na kterých hodnota závisí. Umožňuje snadno pracovat s reaktivními hodnotami, které jsou odvozeny z jiných hodnot.
- **watch** – Používá se pro sledování změn v reaktivních hodnotách. Když dojde ke změně v hodnotě, vykoná se specifikovaná funkce (vedlejší efekt).

Komponenty

`Vue.js` je založen na komponentovém modelu. Tento přístup rozděluje strukturu aplikace na malé, opakovaně použitelné, izolované části, zvané komponenty. Každá komponenta má svůj vlastní stav, metody a nachází se v daný moment v určité fázi životního cyklu. Tento přístup umožňuje psaní přehledného a rozšiřitelného kódu. To je obzvlášť důležité u větších aplikací, ale je dobré dodržovat tento princip i u malých aplikací.

Props a emit

Komponenty mohou přijímat data od své nadřazené komponenty prostřednictvím `props`. Tyto `props` slouží jako vstupní parametry pro komponentu, která s nimi může dále pracovat.

Pro interakci komponenty s její rodičovskou komponentou se používá vestavěná metoda `emit`. Tento mechanismus komponentě umožňuje odeslat událost do rodičovské komponenty, která ji zachytí a může na ní příslušně zareagovat.

Zatímco `props` fungují pro komunikaci od rodičovské komponenty směrem k její dceřiné komponentě, metoda `emit` slouží ke komunikaci opačným směrem od dceřiné komponenty k rodičovské komponentě.

Životní cyklus komponenty

Každá komponenta ve `Vue.js` má svůj vlastní životní cyklus. Tento cyklus popisuje fáze, kterými komponenta prochází během své existence. Pro každou z těchto fází `Vue.js` obsahuje metodu, která umožňuje reagovat na přechod komponenty do určité fáze životního cyklu.

3.1.2 Výhody Vue.js oproti alternativním frameworkům

Jednoduchost na naučení

`Vue.js` je známý svou jednoduchostí pro začínající programátory. Všechen kód (HTML, CSS i JavaScript) týkající se jedné komponenty, je totiž zapsán v jediném souboru, a tak je pro programátora jednodušší se v kódu orientovat, jelikož vše má na jednom místě a nemusí hledat mezi soubory. Díky tomu je `Vue.js` ideální volba pro programátory, kteří nemají s vývojem webových aplikací dlouholeté zkušenosti.

Lehčí a modulární architektura

`Vue.js` je založeno na jednoduché a modulární architektuře, která vývojářům umožňuje začít vyvíjet rychle a bez zbytečných komplikací. `Vue.js` v základu, na rozdíl od některých ostatních frameworků, jako je Angular, neobsahuje funkce pro tvorbu pokročilejších funkcionalit, jako například směrování, správa komplexnějšího stavového systému aplikace, nebo funkce pro podporu práce s formuláři. Vývojář si tyto funkce přidává, až pokud opravdu potřebuje, a má možnost vybrat si z více nabízených možností formou instalovatelných balíčků.

Výkon a optimalizace

`Vue.js` nabízí výborný výkon díky efektivnímu sledování změn v aplikaci pomocí reaktivity založené na JavaScriptovém Proxy [17] objektu. JavaScript Proxy je objekt, který umožňuje provázání více objektů za účelem sledování změn a úpravy dat mezi objekty. `Vue.js` díky tomu automaticky aktualizuje pouze ty části rozhraní, které se skutečně změnily, což znatelně šetří výkon. `Vue.js` také využívá techniky jako postupné načítání komponent (tzv. lazy loading), renderování na straně serveru (tzv. server-side rendering) a dělení kódu (tzv. code-splitting), což zrychluje načítání a celkový běh aplikace.

3.2 Směrování ve Vue.js aplikaci: `Vue Router`

Vue Router [26] je oficiální knihovna pro správu směrování (routingu) ve `Vue.js` aplikacích. Umožňuje definovat různé URL adresy a přiřadit jim odpovídající komponenty, čímž umožňuje navigaci mezi jednotlivými částmi aplikace.

3.2.1 Rozhraní a schopnosti Vue routeru

Router nabízí mnoho funkcionalit, které umožňují jednoduchou a přehlednou správu navigace ve webové aplikaci. Mezi nejdůležitější patří:

- **Deklarativní definice tras**: Umožňuje přehledné mapování cest (URL adres) na komponenty (viz výpis 3.1)
- **Dynamické trasy s parametry**: Umožňuje používání proměnných v rámci cest, které lze na příslušné stránce ze směrovače získat a dále s nimi pracovat.
- **Autentizace navigace**: Router nabízí možnost vytvoření vlastní autentizace a kontroly přístupu k vybraným cestám tak, aby se mohl nepřihlášený uživatel dostat jen na stránky, které jsou k tomu určeny a nedostal se na stránky vyžadující autentizaci ani při zadání cesty přímo do okna vyhledávače.

Následná navigace na jednotlivé URL adresy aplikace poté probíhá pomocí funkce `router.push( name: 'nazevCesty', params )`. Tato funkce provede přesměrování na příslušnou adresu aplikace podle jejího `name` atributu a přiřadí této adrese parametry poskytnuté v proměnné `params`, v případě, že se jedná o parametrizovanou adresu (viz adresa `/Sorting/:datasetId/:uid` ve výpise 3.1). Kromě navigace také přidá tuto adresu do historie směrovače, takže je uživateli umožněno pomocí routeru vracet se v historii otevřených stránek.

```
const routes = [
    {
        path: "/",
        name: "Home",
        component: Home
    },
    {
        path: "/Sorting/:datasetId/:uid",
        name: "Sorting",
        component: Sorting
    }
]

const router = createRouter({
    history: createWebHistory('/'),
    routes
})

export default router;
```

Výpis 3.1: Ukázka mapování URL cest na komponenty aplikace pomocí knihovny `Vue router`. V poli `routes` jsou definovány všechny dostupné cesty aplikace. Atribut `path` slouží k definici cesty, atribut `name` obsahuje název cesty, pomocí kterého se lze na příslušnou cestu odkazovat a atribut `component` obsahuje komponentu, která se uživateli zobrazí při navigaci na příslušnou cestu. Pole `routes` se poté použije jako parametr k volání funkce `createRouter`, která vytvoří instanci routeru a ten je ze souboru exportován a použit v dalších komponentách.

3.3 Nástroj pro správu globálního stavového úložiště: `Pinia`

Pinia [4] je nástroj pro Vue.js určený pro správu globálních stavů aplikace. Tento nástroj umožňuje vytvoření globálního úložiště a jeho metod pracujících se stavy. Toto úložiště poté propaguje do celé hierarchické struktury webové aplikace, a tím umožňuje přístup k těmto datům ze všech částí aplikace. Při změně dat v některém ze stavových úložišť se automaticky změna projeví všude, kde se data používají, a dojde k překreslení komponent, které tato data používají. Tímto mechanismem je zajištěno, že pokud jedna komponenta data modifikuje a druhá komponenta s těmito daty pracuje, vždy budou použita pouze aktuální data a nebude docházet k nekonzistenci dat v aplikaci. viz výpis 3.2.

3.4 Nástroj pro stylování aplikace: `Tailwind CSS`

Tailwind CSS [25] je moderní CSS framework, který umožňuje rychlé a efektivní stylování webových aplikací. Na rozdíl od klasického CSS, kde si programátor musí sám nadefinovat třídy s příslušnými styly, Tailwind CSS přichází s úplně jiným přístupem ke stylování zvaným *utility-first*. Přístup je založen na tom, že každý CSS styl v knihovně má vlastní třídu, která ho reprezentuje. Programátor tyto třídy zapisuje přímo do HTML kódu aplikace bez nutnosti předchozího definování (viz výpis 3.3).

I přes to, že tato knihovna obsahuje styly, které vystačí na velkou většinu požadavků, nabízí také možnost konfigurace tříd pomocí konfiguračního souboru `tailwind.config.js`.

```
import { defineStore } from 'pinia'

export const useGlobalStore = defineStore('globalStore', () => {
  const count = ref(0)

  function increment() {
    count.value++
  }

  function decrement() {
    count.value--
  }

  return { count, increment, decrement }
})
```

Výpis 3.2: Vytvoření globálního úložiště pomocí nástroje Pinia. Úložiště nazvané globalStore obsahuje reaktivní proměnnou count a metody increment a decrement určené pro zvyšování a snižování hodnoty proměnné. Celé toto uložiště je exportováno jako funkce useGlobalStore, která po zavolání vrací referenci na úložiště globalStore včetně všech reaktivních proměnných a metod.

V tomto souboru lze definovat vlastní barevné šablony, velikosti fontů, velikosti obrazovek pro nastavování responzivity a mnoho dalšího[1].

Knihovna navíc obsahuje zabudované PurgeCSS [7], což je nástroj, který automaticky při sestavování aplikace pro produkční prostředí odstraní všechny nepoužité styly. Díky tomu je výsledný CSS soubor velice malý a obsahuje pouze styly, které jsou reálně v aplikaci použity. To urychluje stahování CSS souboru ze serveru a tím i čas načítání aplikace.

```
<div class="bg-blue-500 text-white p-4">
</div>
```

Výpis 3.3: Ukázka použití předdefinovaných Tailwind CSS tříd v kódu. Třída `bg-blue-500` nastaví barvu pozadí elementu na modrou, `text-white` nastaví barvu textu na bílou a `p-4` nastaví vnitřní odsazení elementu na `1rem`.

3.5 Knihovna pro implementaci drag and drop funkcionality: `vue-draggable-plus`

`Vue-draggable-plus` [22] je knihovna, která nabízí možnost implementace drag and drop funkcionality do Vue.js aplikací. Tato knihovna vznikla jako nadstavba nad knihovnou `SortableJS` [21] a umožňuje tvorbu interaktivních seznamů prvků s možností přeuspořádání pomocí myši nebo pomocí dotykového ovládání.

Díky reaktivnímu přístupu Vue.js se všechny změny v pořadí položek automaticky synchronizují se stavem celé aplikace. Hlavní výhodou je snadné použití a integrace do aplikace, a to bez nutnosti složité konfigurace a nastavování. Knihovna podporuje pokročilejší funkcionality jako drag and drop mezi seznamy, změnu pořadí položek seznamu, nebo možnost

[1] Popis možností konfigurace knihovny Tailwind CSS zde: https://v1.tailwindcss.com/docs/configuration

```
import { createApp } from 'vue'
import App from './App.vue'
import VueDragscroll from "vue-dragscroll";

const app = createApp(App);

app.use(VueDragscroll);
app.mount('#app')
```

Výpis 3.4: Ukázka přidání funkcionality posunu obsahu stránky pomocí tažení myši pomocí knihovny `VueDragscroll`. Pro správné fungování je potřeba následující část kódu zakomponovat do hlavního souboru Vue.js aplikace `main.js`

tvorby obslužných metod pro události vyvolávané knihovnou. Knihovna je dobře optimalizovaná pro mobilní zařízení a není potřeba speciálních úprav. Umožňuje například ale nastavit čas zpoždění, po který musí uživatel držet element, než ho bude moci přesunout. To je kvůli možné přítomnosti funkcionality drag to scroll v aplikaci.

3.5.1 Komponenta `VueDraggable`

Hlavní komponentou knihovny je komponenta s `VueDraggable`. Touto komponentou se obalí seznam položek, na které je potřeba aplikovat drag and drop funkcionalitu. Komponenta automaticky propojuje DOM s reaktivním datovým modelem Vue.js aplikace (viz kapitolu 3.1.1), takže při změně pořadí položek se změna okamžitě projeví jak v uživatelském rozhraní, tak i v datech.

3.6 Knihovna pro drag to scroll: `vue-dragscroll`

Vue-dragscroll [3] je knihovna pro Vue.js, umožňující snadné přidání funkcionality `drag to scroll` na libovolný prvek aplikace. Tato funkcionalita umožňuje uživateli posouvat obsah stránky tažením myši, čímž zjednodušuje navigaci na stránce. Použití této funkcionality je obzvlášť důležité u aplikací, které mají obsah posouvatelný v obou směrech (horizontálním i vertikálním), a klasické posuvníky na krajích okna by nebyly vhodné z hlediska přívětivosti uživatelského rozhraní.

Po vložení výše uvedeného kódu do hlavní komponenty aplikace stačí přidat atribut `v-dragscroll` k prvku, který má funkcionalitu `drag to scroll` podporovat. Pokud je potřeba vyloučit nějakou z dceřiných komponent z této funkcionality, stačí na ní přidat atribut `data-no-dragscroll`.

3.7 Knihovna pro odesílání HTTP požadavků: `Axios`

Axios [1] je knihovna určená k odesílání HTTP požadavků, která usnadňuje komunikaci mezi klientskou aplikací a serverem. Často se využívá v moderních webových aplikacích vytvořených pomocí JavaScriptových frameworků. Mezi hlavní výhody `Axiosu` patří podpora asynchronního zpracování požadavků, automatická konverze odpovědí na `JSON` [2] formát a možnost snadného nastavení globálních konfigurací, jako například hlavičky požadavku nebo autentizaci.

Knihovna podporuje všechny klasické metody HTTP požadavku jako **GET**, **POST**, **PUT** a **DELETE**.

3.8 Komunikace přes websockets: knihovna `socket.io-client`

Socket.io-client [23] je JavaScriptová knihovna umožňující navázání spojení mezi klientskou aplikací a serverovou aplikací pomocí technologie Socket.IO [16]. Ke komunikaci knihovna využívá síťový protokol `WebSocket` [15], který na rozdíl od klasického HTTP protokolu nabízí možnost trvalého spojení a následné oboustranné komunikace.

Knihovna funguje na principu událostí. To znamená, že komunikace mezi klientem a serverem neprobíhá klasickým způsobem požadavek–odpověď, ale klient se serverem si posílají pojmenované události, na které může příjemce reagovat jinou pojmenovanou událostí.

3.8.1 Metoda pro naslouchání událostem: `socket.on`

Metoda `socket.on(nazev_udalosti, funkce)` se používá k naslouchání konkrétní události. Když je daná událost, jejíž název se metodě poskytne v parametru `nazev_udalosti` přijata ze serveru, provede se připojená funkce poskytnutá v parametru `funkce` a této funkci se předají i případná další data, která byla poslaná s událostí.

3.8.2 Metoda pro odesílání událostí: `socket.emit`

Metoda `socket.emit(nazev_udalosti, data)` se používá k odeslání události ostatním zařízením. V parametru `nazev_udalosti` se poskytne metodě název odesílané události a v parametru `data` se metodě předají data, která jsou připojena k odesílané události.

3.9 Nástroj pro sestavování a nasazování aplikací: `Docker`

Docker [8] je nástroj určený k vytváření, nasazování a správě aplikací v takzvaných kontejnerech. Kontejnery jsou lehká, izolovaná prostředí, která obsahují vše potřebné pro běh aplikace, jako jsou zdrojové kódy, instalované knihovny, konfigurace a jiné závislosti.

Kontejnery se v Dockeru vytvářejí pomocí dockerového obrazu. Dockerový obraz je šablona, která obsahuje všechny informace potřebné k vytvoření dockerového kontejneru. Obraz je neměnný a slouží jako základ pro vytvoření jednoho nebo více běžících kontejnerů. Uživatel má dvě možnosti, jak může pracovat s dockerovými obrazy. Buďto si může vytvořit vlastní obraz pomocí `Dockerfile` souboru, který obsahuje instrukce pro sestavení obrazu ze zdrojového kódu, nebo může použít některý z veřejně dostupných obrazů z repozitářů, jako je například Docker Hub[2]. Vlastní obrazy se vytvářejí pomocí Dockerfile, což je soubor obsahující sérii instrukcí, které Docker interpretuje a vykonává s cílem vytvoření obrazu.

Hlavními výhodami dockeru, oproti ostatním možnostem nasazení, jsou:

- **Izolace aplikace** – Každá aplikace běží ve vlastním kontejneru, jehož obsah je izolován od ostatních kontejnerů. Předchází se tak případným konfliktům mezi závislostmi.
- **Znovupoužitelnost** – Jeden dockerový obraz může být použit k vytvoření více kontejnerů, na různých prostředích.
- **Přenositelnost** – Aplikaci lze díky izolovanému prostředí uvnitř kontejneru spouštět na různých zařízeních bez ohledu na operační systém nebo platformu zařízení. Jedinou podmínkou pro spuštění aplikace zabalené v dockerovém kontejneru je nainstalovaný a funkční nástroj Docker.

[2] Adresa úložiště Docker Hub: `https://hub.docker.com/`

Docker nabízí součástí instalačního balíčku také nástroj `docker-compose`. Ten umožňuje definovat a spouštět skupinu více kontejnerů pomocí jednoduchého konfiguračního souboru `docker-compose.yml`. Ten obsahuje informace o tom, jaké kontejnery se spustí, jak jsou tyto kontejnery nakonfigurovány a jak budou mezi sebou komunikovat.

Kapitola 4

Iterativní testování uživatelského rozhraní aplikace

Iterativní testování uživatelského rozhraní aplikace je proces, při kterém se opakovaně testuje s uživateli použitelnost a uživatelská přívětivost uživatelského rozhraní. Jednotlivé iterace testování se provádí opakovaně, většinou po přidání funkcionality, u které je potřeba zvýšení uživatelské přívětivosti. Výsledky z testování se použijí pro návrh lepší verze funkcionality a provede se další iterace, aby se zjistilo, jestli vylepšení vedlo ke zvýšení použitelnosti.

4.1 Iterativní testování uživatelského rozhraní podle Steva Kruga

Steve Krug je uznávaný expert v oblasti testování použitelnosti webových aplikací a celkově softwaru. Je autorem knihy *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems* [13], ve které popisuje proces iterativního testování tak, aby byl jednoduchý, praktický a efektivní. Vzhledem k tomu, že má sám zkušenosti s uživatelským testováním, jsou jeho poznatky založené na reálných zkušenostech.

Základem jeho filozofie je testovat co nejdříve a co nejčastěji. Není potřeba čekat na finální verzi produktu, kdy bude produkt odpovídat návrhu, ale ideálně testovat aplikaci už ve fázi prototypu, aby se odhalily případné nedostatky návrhu už v rané fázi projektu. Čím dříve se totiž nedostatek odhalí, tím je levnější a jednodušší ho opravit.

Pro testování není potřeba velké množství uživatelů. Při velkém množství uživatelů se snižuje efektivita a zvyšují náklady. Nejlepší je tedy pro testování sehnat tři až pět lidí, kteří budou mít možnost aplikaci testovat opakovaně. Většina chyb v uživatelském rozhraní se často opakuje, takže více osob by pouze odhalovalo stejné chyby dokola.

Samotné testování probíhá tak, že uživatel dostane jednoduchý úkol, který musí v aplikaci splnit bez pomoci okolí. Úkol by se měl skládat z klasických scénářů, které jsou očekávané při používání koncovým uživatelem. Pozorovatel během provádění úkolu uživatele sleduje a snaží se odhalit nedostatky aplikace, které mu provádění úkolu ztěžují, a zapisuje si je. Pozorovatel, ani nikdo jiný, nesmí do procesu testování zasahovat, radit, nebo navádět uživatele k cíli. To, že si uživatel při provádění úkolu neví rady, znamená, že návrh aplikace je špatný a měl by být vylepšen.

Po skončení testu se pozorovatel zaměří na identifikaci největších problémů při testování. Nejlepší je zaměřit se nejprve na vyřešení problémů, které uživateli nejvíce ztěžovaly

provedení úkolu. Tyto nedostatky se opraví a poté je na řadě další iterace testování. Ta by v případě správného návrhu neměla odhalovat stejné problémy, jako předchozí iterace, ale případné další problémy.

4.2 Provádění iterativního testování uživatelského rozhraní aplikace

Při testování jsem se inspiroval radami Steva Kruga (viz v kapitole 4). Nejprve jsem si musel sehnat uživatele, kteří mi budou aplikaci pomáhat testovat. Snažil jsem se je vybrat tak, abych měl ideálně v této skupině jak lidi, kteří umí pracovat s počítači, tak i lidi, kteří nejsou v práci s počítačem nějak obzvlášť znalí.

První iterace testování byla provedena nad původní verzí aplikace od Michala Janů (viz v kapitole 2.3), abych zjistil hlavní nedostatky aktuálního řešení aplikace. Následné iterace testování byly prováděny již s úpravami v této aplikaci. V každé iteraci jsem se snažil klást důraz na to, aby uživatel byl schopen co nejjednodušeji seřadit všechna tvrzení bez nějakých větších problémů. Kvůli tomu jsem vytvořil dvě datové sady, které jsem při testování používal. První sada se zaměřovala na seřazení slovně psaných číslovek od nejmenší po největší. Druhá datová sada obsahovala obrázky usmívajícího se obličeje, které měly různé velikosti. Uživatel měl za úkol seřadit obrázky podle velikosti od nejmenšího po největší. Tyto datové sady byly vytvořeny za účelem, aby uživatelé při testování nemuseli úplně přemýšlet nad svými osobními postoji u datových sad, které se zaměřují více na zjišťování lidské subjektivity, a mohli se tak co nejvíce zamyslet nad samotným procesem řazení.

Iterativní testování nebylo prováděno nějak pravidelně a strukturovaně. Prováděl jsem iterace testování vždy až potom, co jsem implementoval zlepšení, nebo novou funkcionalitu, abych neztrácel čas testováním stejných nedostatků.

Podrobný popis průběhu iterativního testování uživatelského rozhraní aplikace a tvorby návrhu na základě výsledků iterativního testování je popsán v kapitole 5.

Kapitola 5

Návrhy a modely aplikace pro Q-metodologii

Po dokončení každé iterace testování (viz sekci 4.2) jsem navrhl změny na základě nedostatků, které vyšly najevo z výsledků testování. Při návrhu funkcionalit a komponent jsem se inspiroval knihou *Don't Make Me Think!* [14] od Steva Kruga, ve které popisuje, jak tvořit aplikace a systémy tak, aby byly pro uživatele snadno použitelné a nemusel přemýšlet nad tím, jak s nimi pracovat.

5.1 Průzkum uživatelského rozhraní aplikace `Trello`

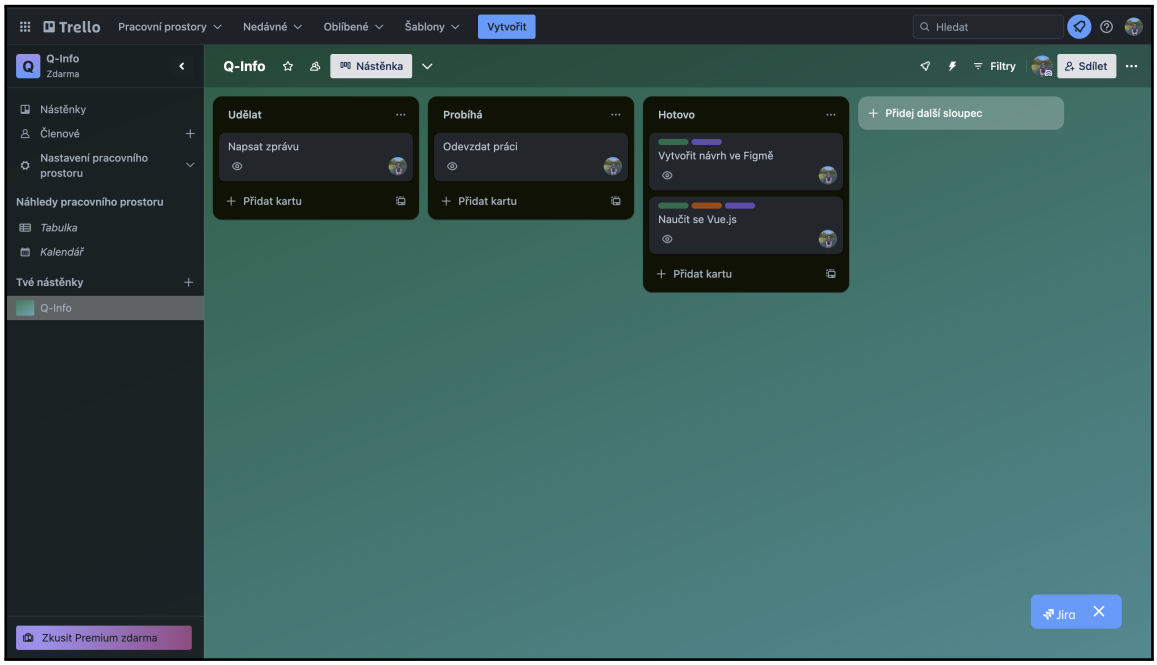
Trello je kolaborativní nástroj používaný pro správu projektů v rámci organizací. Je vyvíjeno velkou softwarovou společností Atlassian, která se zaměřuje na vývoj produktů pro řízení projektů, spolupráci, výměnu informací atd. Trello je určeno primárně pro správu úkolů v rámci projektu. Tyto úkoly lze přesouvat mezi jednotlivými stavy, jako je například analýza, vývoj, testování a další. Hlavní okno nástroje (viz obrázek 5.1) je rozděleno do jednotlivých sloupců, přičemž každý sloupec reprezentuje stav vývoje, ve kterém se daný úkol může nacházet. Kliknutím na prvek ve tvaru obdélníku obsahující název úkolu se otevře okno obsahující detailní informace o úkolu, jako například detailní popis úkolu, uživatele, kterému byl úkol přidělen, nebo požadovaný termín splnění úkolu.

Přesouvání úkolů v aplikaci Trello

Pro přesouvání úkolů mezi jednotlivými stavy vývoje je použita drag and drop funkcionalita (viz obrázek 5.2). Na zařízeních používajících myš je drag and drop funkcionalita velice intuitivní. Hned po zmáčknutí tlačítka myši na prvku úkolu se prvek s úkolem lehce nakloní, aby uživatel věděl, že s prvkem manipuluje a může ho přesouvat po ploše.

Na mobilních zařízeních je po přiložení prstu na element úkolu lehká prodleva, než se prvek nakloní. Je to z toho důvodu, že aplikace zároveň používá gesto tažení prstem po obrazovce pro přesun po ploše obsahující úkoly. Proto je potřeba touto malou časovou prodlevou odlišit, zda-li chce uživatel přesouvat kartu s úkolem, nebo chce provést posun obrazovky s úkoly.

Karta s úkolem je nakloněna po celou dobu, co uživatel drží levé tlačítko myši, nebo na mobilních zařízeních drží prst na obrazovce. Po přesunu nad některý ze sloupců se ve sloupci zobrazí lehce zatmavená oblast velikosti přetahované karty (viz obrázek 5.2), která reprezentuje místo, kam bude karta v případě puštění vložena. Pokud uživatel kartu upustí



Obrázek 5.1: Aplikace Trello používaná pro správu projektů. Sloupce v hlavní části aplikace reprezentují jednotlivé fáze vývoje projektu, mezi kterými se můžou jednotlivé úkoly přesouvat. V levé části aplikace se nachází nabídka s navigací na další stránky aplikace. V horní části aplikace se nachází panel nabídek aplikace.

mimo oblast, kam může být karta vložena, je karta vrácena na původní pozici, odkud uživatel přesun zahájil.

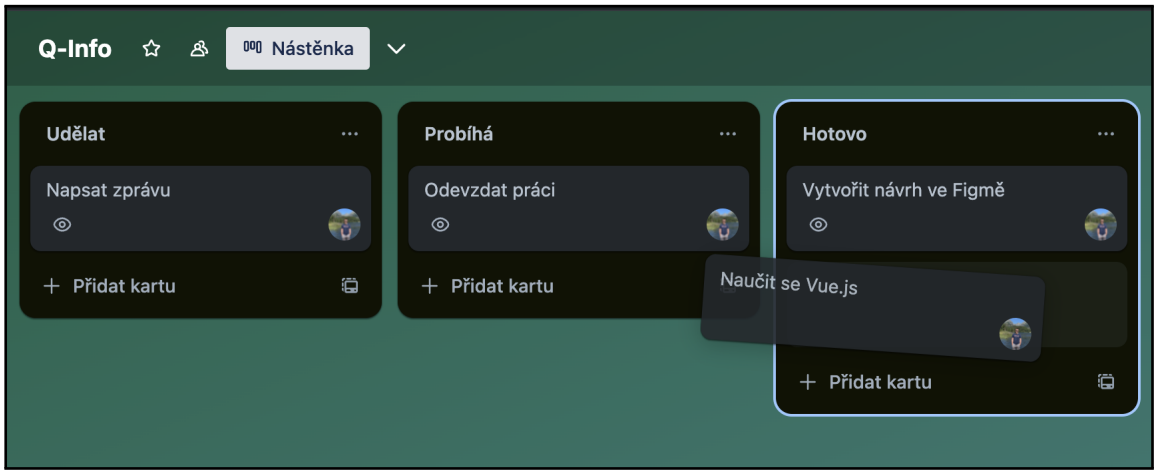
Funkcionalita přiřazení značek k úkolu v aplikaci Trello

Pro lepší organizaci úkolů aplikace Trello používá funkcionalitu přiřazení štítků k jednotlivým úkolům na ploše. Tato funkcionalita je dostupná na obrazovce detailu úkolu (obrázek 5.3). V pravé části okna detailu se nachází sekce „Přidat na kartu“ a v této sekci se nachází tlačítko „Štítky“. Po kliknutí na toto tlačítko se zobrazí nabídka všech dostupných štítků, které jsou odlišeny barvou. Ve spodní části komponenty pro výběr štítku se navíc nachází možnost tvorby vlastního štítku s možností výběru barvy.

Po výběru štítku u úkolu a zavření detailního pohledu úkolu se štítky zobrazí i na samotné ploše v levém horním rohu u úkolu jako malý proužek s příslušnou barvou (viz obrázek 5.1). Použití štítků v aplikaci pomáhá k lepší přehlednosti plochy díky možnosti rozřazení úkolů do pomyslných kategorií. Dobrým příkladem použití štítků je nastavení priority jednotlivých úkolů, kde červený štítek u úkolu může značit nejvyšší prioritu a například zelený štítek značit úkoly s nízkou prioritou.

5.2 Návrh komponent pro drag and drop funkcionalitu

Při návrhu komponent pro drag and drop funkcionalitu použitou při přesouvání tvrzení jsem se inspiroval aplikací Trello (popsáno v kapitole 5.1), ve které je tato funkcionalita použita a je implementována přehledně a intuitivně.



Obrázek 5.2: Ukázka použití funkcionality drag and drop pro přesun úkolů v aplikaci Trello. Při přetahování prvku se přesouvaný prvek nakloní, aby uživatel věděl, že s prvkem manipuluje. Při přesunu prvku nad oblast, kde může být prvek umístěn je zobrazena lehce zatmavená zóna značící místo vložení při puštění prvku.

Při návrhu této komponenty jsem se zaměřil na to, aby byla přehledná a intuitivní a uživatel měl přehled o tom, co se s přesouvaným prvkem děje a co se stane, pokud kartu na určitém místě pustí.

Když uživatel začne provádět přesun, musí mu být jasné, se kterým prvkem zrovna manipuluje. Při chycení karty se tak průhlednost této karty zvýší a také se jí nastaví přerušovaný, tenký šedý okraj na rozdíl od souvislého, tlustého černého kraje u ostatních karet. Díky tomu uživatel ihned pozná, se kterou kartou zrovna pracuje.

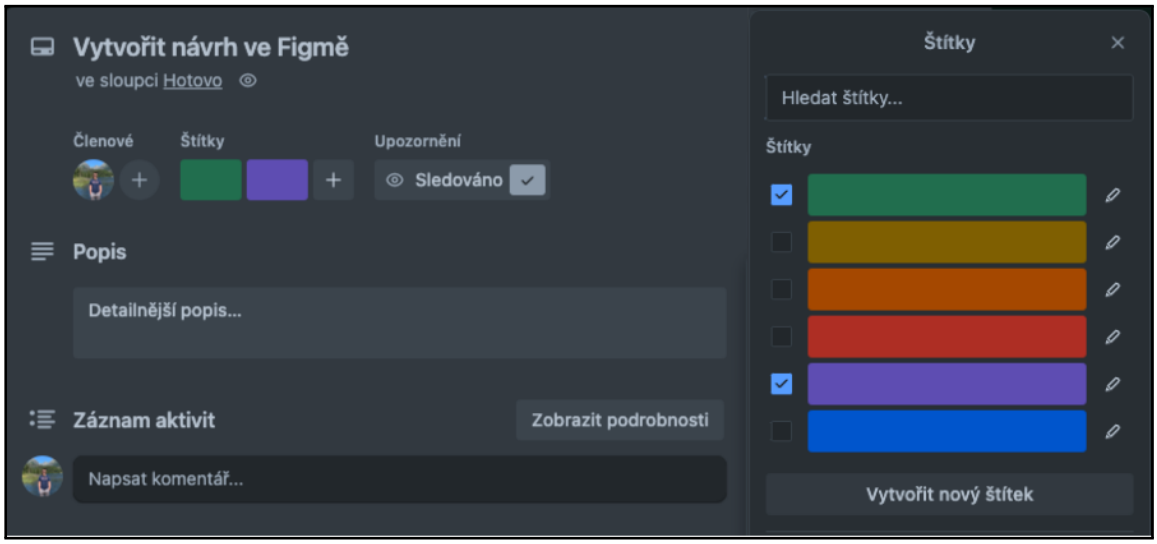
Poté co uživatel přesune prvek do oblasti, kde je možno tvrzení vložit, je potřeba uživateli zobrazit, kam bude v případě upuštění prvek vložen. K tomu slouží tzv. ghost komponenta, která se zobrazí v místě vložení prvku. V tomto případě vypadá ghost komponenta přesouvané karty jako přesouvaná karta, akorát má zvýšenou průhlednost a přerušovaný okraj. Je to z toho důvodu, aby si uživatel nemyslel, že karta byla do kategorie vložena ještě před jejím upuštěním, ale pouze mu aplikace radí, kam bude vložena po tom, co kartu pustí.

5.3 Návrh komponent pro funkcionalitu předřazení

5.3.1 Prvotní návrh komponenty pro předřazení tvrzení

V první verzi návrhu komponenty pro funkcionalitu předřazení (viz obrázek 5.5) jsem navrhl komponentu, která umožňuje předřazení tvrzení tak, jak mají tuto funkcionalitu implementovány většina již existujících aplikací pro Q-metodologii (popsáno v kapitole 2.2). Tento princip spočívá v předřazení karet do tří hlavních kategorií, které reprezentují krajní, extrémní kategorie a prostřední, neutrální kategorii.

K procesu předřazení není uživatel nucen jako ve většině existujících řešení. Ostatní řešení nutí uživatele předřadit všechna tvrzení, než mu je umožněno přesunout se k samotnému řazení karet do všech dostupných kategorií. Tato komponenta byla uživateli dostupná přímo v kroku hlavního řazení, kdy si karty pouze mohl roztřídit a poté tyto předřazené karty viděl ve třech kategoriích ve vrchním panelu (viz obrázek 5.6).



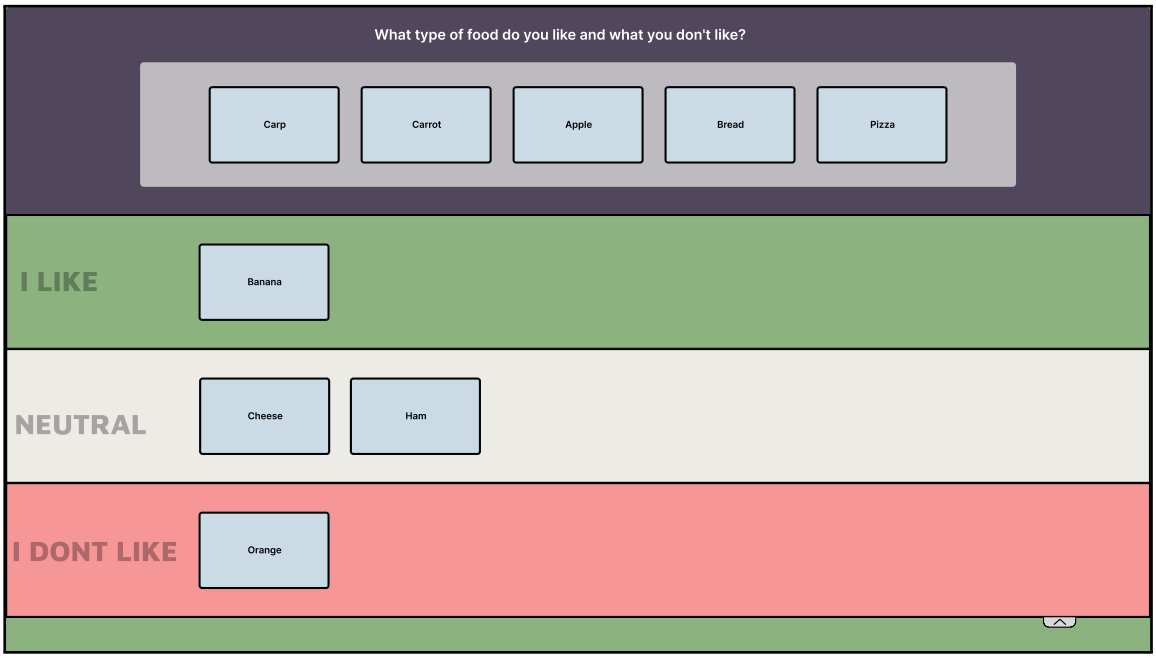
Obrázek 5.3: Ukázka komponenty pro zobrazení detailu úkolu a s ní spojené komponenty pro přidávání štítků k úkolu v aplikaci Trello. Komponenta v levé části zobrazuje detailní popis vybraného úkolu a obsahuje možnosti úpravy informací o úkolu jako popis úkolu, osoba přiřazená k řešení úkolu a další. V pravé části se nachází komponenta umožňující přiřazení štítku k úkolu, které slouží ke kategorizaci úkolů podle vlastních kritérií.

Obrázek 5.4: Návrh komponenty pro funkcionalitu drag and drop. Při chycení karty se pozadí karty lehce zprůhlední, aby uživatel věděl, že s kartou manipuluje. Po přesunu tvrzení nad oblast, kde může být vložena se v místě pro vložení zobrazí průhledná kopie přetahované karty. To uživateli ukazuje, kam bude tvrzení vloženo v případě puštění.

I přes to, že se toto řešení používá ve většině existujících nástrojů pro řazení pomocí Q-metodologie, rozhodl jsem se po iteraci testování s tímto vylepšením, že není vyhovující z důvodu malého prostoru pro zobrazování na mobilních zařízeních. Při zobrazení tří kategorií s předřazenými tvrzeními se totiž značně snižuje přehlednost aplikace a snižuje se také čitelnost jednotlivých tvrzení. Proto jsem se rozhodl, že bude lepší vymyslet jiné řešení, které bude přehlednější a bude vypadat lépe i na mobilních zařízeních.

5.3.2 Předřazení tvrzení pomocí štítků reprezentujících kategorie

Po neúspěchu předchozího návrhu jsem se rozhodl navrhnout nový způsob funkcionality předřazení a to pomocí štítků (obrázek 5.7), které může respondent přiřadit ke každému tvrzení po přepnutí vrchního panelu do režimu předřazení. K implementaci této funkcionality jsem se rozhodl po prozkoumání aplikace Trello, ve které je tato funkcionalita také používána právě tímto způsobem (popsáno v kapitole 5.1).



Obrázek 5.5: Návrh komponenty pro předřazení tvrzení pomocí tří hlavních kategorií. Po rozbalení komponenty pro předřazení jsou uživateli nabídnuty pozitivní, neutrální a negativní kategorie, do kterých si může tvrzení předtřídit.

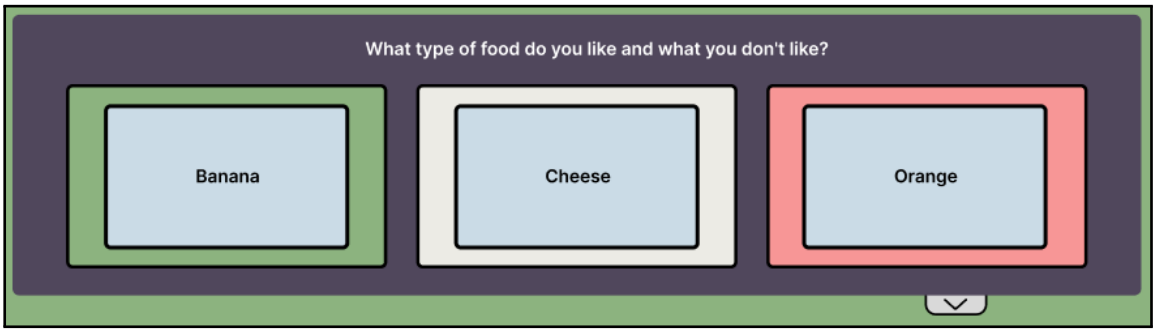
Uživateli se v rámci procesu předřazení zobrazí všechna tvrzení, které mají v pravém horním rohu barevné značky s čísly a barvami reprezentujícími jednotlivé kategorie z pole pro řazení. Tyto značky mají ve výchozím stavu nastavenou zvýšenou průhlednost a mají přerušovaný okraj, což značí, že štítek je pro dané tvrzení neaktivní. Pokud respondent na některý ze štítků klikne, zruší se jeho průhlednost a okraj se nastaví na plný a tím se štítek stane aktivní pro danou kartu.

Uživatel má možnost vybrat u karty libovolný počet štítků. Po zavření komponenty pro předřazení se aktivní značky zobrazují u tvrzení ve frontě a respondent má tak možnost karty řadit podle toho, které značky u jednotlivých karet vybral. Značky se zobrazují i u tvrzení, která již byla přesunuta do plochy pro řazení, aby uživatel mohl v průběhu řazení přehodnotit své rozhodnutí a kartu s tvrzením přesunout do jiné kategorie.

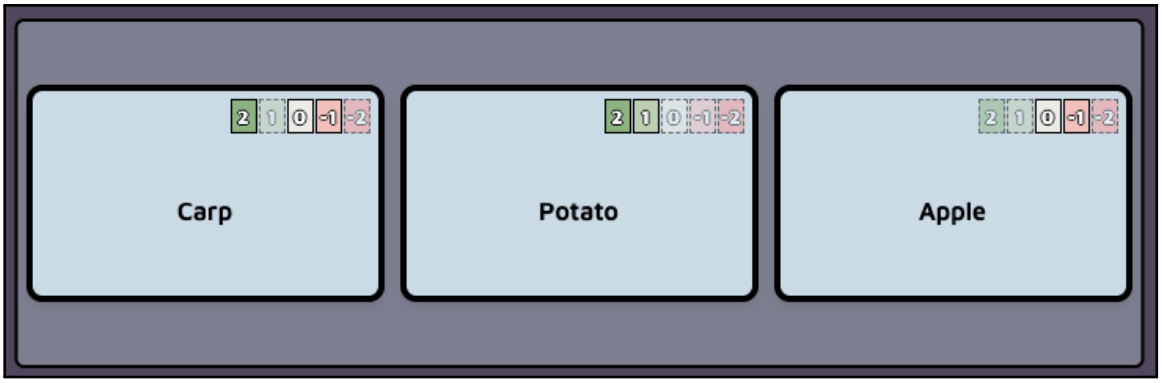
5.3.3 Možnost použití vlastních barevných značek

Po konzultaci s vedoucím práce a se studentkou Karolínou Pirohovou, která v rámci své bakalářské práce [19] navrhovala optimální vzhled plochy pro řazení, jsem se ale dozvěděl, že ne všechny studie musí mít nutně barevně odlišené jednotlivé kategorie. Kromě toho u jednotlivých kategorií nemusí být přiřazená číselná hodnota, která reprezentuje míru pozitivity či negativity. V některých případech může totiž například červená a zelená barva uživateli vzbuzovat pocit něčeho negativního a pozitivního, i když to tak není vůbec myšleno. Proto jsem se po konzultaci s Karolínou Pirohovou rozhodl jednotlivé kategorie barevně nerozlišovat a nechat všem nastavené stejné, světle modré pozadí (viz obrázek 6.3).

Z tohoto rozhodnutí ale plynula také potřeba nového návrhu značek u karet. V předchozí verzi (viz sekci 5.3.2) se totiž používaly barvy kategorií také u dostupných značek, kterými si uživatel mohl v rámci procesu předřazení jednotlivé karty označit. Vzhledem k tomu, že



Obrázek 5.6: Návrh vrchního panelu s komponentou pro zobrazení předřazených tvrzení do tří hlavních kategorií. Ve třech hlavních kategoriích se zobrazují tvrzení, která uživatel předřadil v komponentě pro předřazení (viz obrázek 5.5).

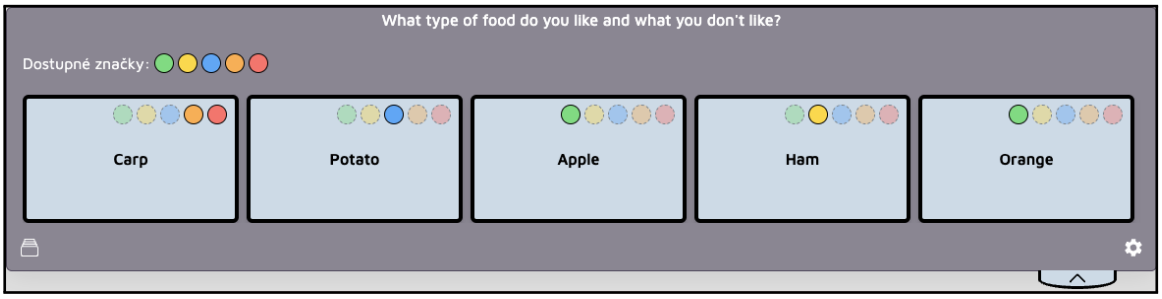


Obrázek 5.7: Návrh komponenty pro předřazení pomocí štítků přiřazených k jednotlivým tvrzením. Štítky se zobrazují v pravém horním rohu karty s tvrzením. Pro každou kategorii z tabulky pro řazení se vykreslí u tvrzení štítek s barvou kategorie, ke které štítek patří. Všechny štítky jsou ve výchozím stavu neaktivní a kliknutím na příslušný štítek na kartě ho může uživatel aktivovat.

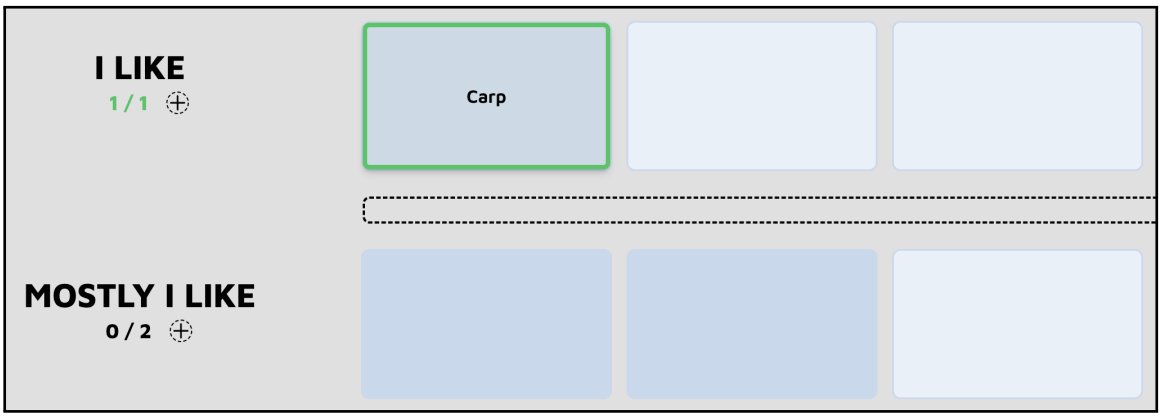
ale nyní kategorie přímo svou barvu nemají, bylo potřeba vymyslet jiný způsob, jak značky pro předřazení používat.

Barvy u řádků zůstaly pouze formou malých ukazatelů u popisu řádku v levé části obrazovky. Při začátku řazení jsou u všech řádků na začátku kolečka se symbolem +, který znázorňuje, že k řádku lze přiřadit nějakou z nabízených barev. Nabídku barev v paletě barev lze definovat v příslušné datové sadě studie. Pokud by to studie vyžadovala, lze v datové sadě i přiřadit výchozí barvy řádků. Po kliknutí na značku se symbolem + se otevře paleta dostupných barev pro značky.

Ve vrchním panelu pro předřazení se poté uživateli zobrazí všechny barevné značky přiřazené k řádkům v ploše. Tyto značky se také zobrazí u jednotlivých karet v panelu pro předřazení, pouze budou neaktivní. Uživatel má možnost kliknutím na značku aktivovat, nebo deaktivovat. Všechny značky, které uživatel aktivoval, se poté zobrazují i v samotné frontě pro řazení.



Obrázek 5.8: Finální návrh komponenty vrchního panelu pro předřazení pomocí barevných značek reprezentujících jednotlivé kategorie. Nad frontou karet se nachází seznam dostupných barevných značek. Pod ním je fronta karet, u kterých lze aktivovat či deaktivovat jednotlivé barevné značky.

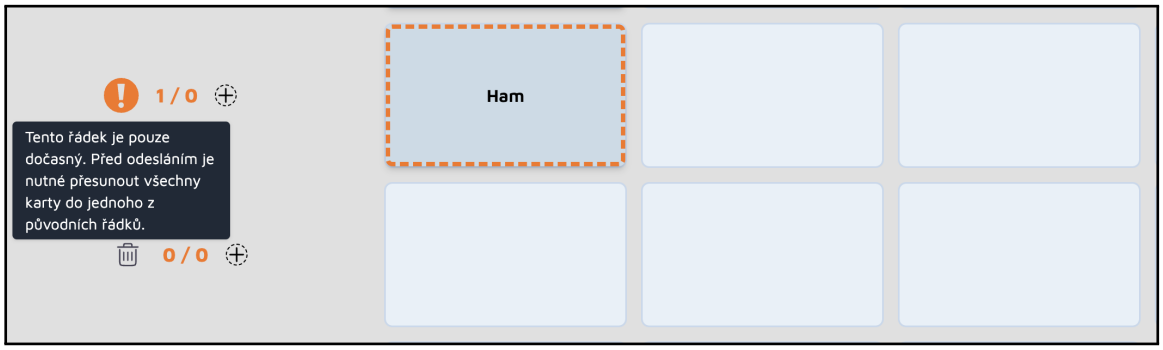


Obrázek 5.9: Komponenta pro přidání nového řádku mezi kategorie. Po vložení karty do plochy s přerušovaným okrajem mezi kategoriemi se přidá nová, dočasná kategorie do plochy pro řazení. Tato kategorie slouží jako místo pro dočasné odložení karet, pokud se uživatel rozhoduje mezi dvěma sousedícími kategoriemi.

5.4 Možnost přidání dočasných řádků do plochy

Při sledování uživatelů při řazení fyzických kartiček jsem často narazil na situace, kdy si uživatel v danou chvíli některá tvrzení nevkládal do předurčených políček, ale mezi jednotlivé kategorie. To dělali, protože se v daný moment nedokázali rozhodnout, do které ze dvou kategorií chtějí kartu přesunout. Dali si tak přesouvanou kartičku mezi kategorie, aby se mohli rozhodnout později, až do kategorie přesunou i další karty.

Na základě tohoto chování jsem se rozhodl tuto funkcionalitu navrhnout a implementovat. Mezi jednotlivými kategoriemi, které již obsahují alespoň jednu kartu, se zobrazí plocha ohraničená přerušovaným, černým okrajem (viz obrázek 5.9). Tato plocha slouží pro vložení karty. Vložení karty způsobí přidání nového dočasného řádku, který slouží jako místo, kam si může uživatel dočasně odkládat karty. Karty vložené do dočasného řádku jsou označeny přerušovaným oranžovým okrajem (viz obrázek 5.10), aby uživatel věděl, že karty mohou být zde vloženy pouze dočasně a před dokončením řazení je musí přesunout do některého z původních řádků plochy. V případě, že uživatel nechá některé karty v dočasné kategorii, nezobrazí se mu vůbec tlačítko pro dokončení řazení a musí je přesunout do některé z původních kategorií.



Obrázek 5.10: Komponenty s řádky pro dočasné vložení karet. Vrchní řádek obsahuje kartu, která je označena oranžovým přerušovaným okrajem. V levé části řádku je navíc bílý vykřičník, který po najetí myši zobrazí text s upozorněním, že uživatel musí před dokončením řazení přesunout tvrzení z tohoto dočasného řádku do některého z původních řádků plochy. Spodní řádek neobsahuje žádné karty, a tak je u něj místo vykřičníku tlačítko s ikonou koše. Kliknutí na toto tlačítko způsobí odstranění dočasného řádku.

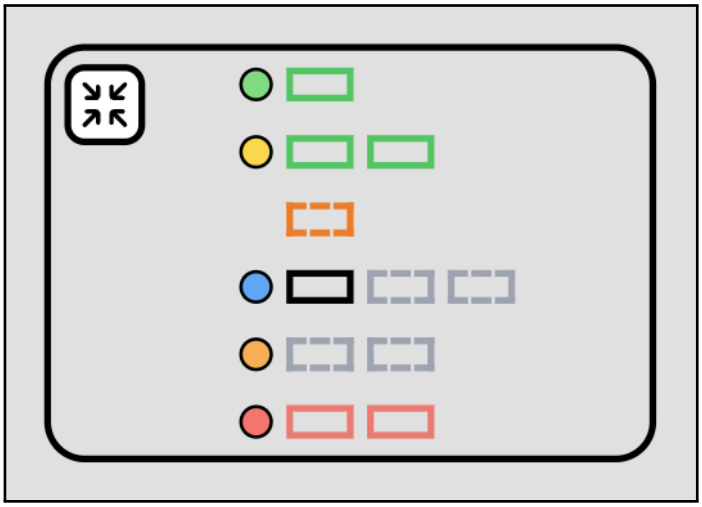
5.5 Nový vzhled minimapy

V původní verzi aplikace, kterou vytvořil Michal Janů (viz sekci 2.3), se minimapa nacházela podél celého pravého okraje aplikace. Tato minimapa zabírala zbytečně moc místa a navíc byla nepřehledná. Kvůli změně vzhledu plochy pro řazení bylo navíc vhodné, aby se i vzhled minimapy aktualizoval a odpovídal tak ploše pro řazení. Proto jsem se rozhodl navrhnout novou komponentu pro minimapu (viz obrázek 5.11), která bude odpovídat vzhledu plochy pro řazení, řádky budou zobrazeny stejně jako v ploše a nebude zabírat tolik místa, což je obzvlášť důležité u mobilních zařízení.

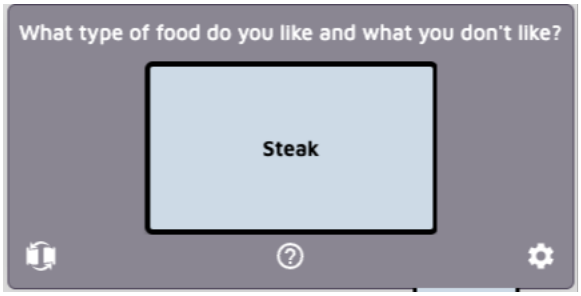
5.6 Návrh komponenty vrchního panelu

Vrchní panel v původní verzi aplikace sloužil pouze pro zobrazení fronty karet určených pro řazení. Vzhledem k tomu, že z iterativního testování vyšla potřeba přidat do aplikace funkcionalitu předřazení, rozhodl jsem se, že bude nejlepší komponentu vrchního panelu upravit a přidat do ní možnost zobrazení fronty pro předřazení. Kromě toho jsem se rozhodl, že i nastavení aplikace by mohlo být zobrazováno ve vrchním panelu místo modálního okénka implementovaného v původní verzi aplikace.

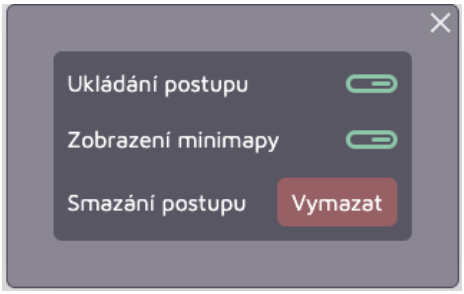
Nyní je tedy vrchní panel navržen tak, aby zobrazoval klasickou frontu karet a po kliknutí na tlačítko v levém spodním rohu (viz obrázek 5.12a) se otevřel panel pro předřazení karet (viz obrázek 5.8). Po kliknutí na ikonu ozubeného kolečka se otevře vrchní panel pro nastavení aplikace (viz obrázek 5.12b).



Obrázek 5.11: Komponenta nové minimapy pro lepší přehled v ploše. Řádek v minimapě reprezentující karegorii z plochy má na levé straně zobrazenou barevnou, kulatou značku. Tato značka znázorňuje barevné označení kategorie v ploše pro řazení. Elementy v řádku s přerušovaným šedým okrajem znázorňují místa, kam může být v ploše vložena karta. Elementy se souvislým okrajem znázorňují karty vložené do kategorie. Komponentu minimapy lze schovat pomocí tlačítka v levém horním rohu.



(a) Komponenta vrchního panelu s frontou karet pro řazení. Vždy se zobrazuje pouze jedna karta. Následující karty jsou skryté, dokud uživatel kartu, která je na řadě nepřesune do některé z kategorií. Ve vrchní části panelu se nachází otázka pro řazení. Ve spodní části jsou tlačítka pro otevření vrchního panelu pro předřazení, skrytí nebo zobrazení otázky a otevření panelu pro nastavení.



(b) Komponenta vrchního panelu pro nastavení aplikace. Umožňuje povolení, či zakázání ukládání postupu, zobrazení nebo schování minimapy a vymazání postupu řazení. Kliknutím na tlačítko s křížkem v pravém horním rohu se panel přepne zpět do klasického zobrazení.

Kapitola 6

Implementace aplikace pro Q-metodologii

6.1 Datová úložiště aplikace

V aplikaci jsou vytvořena tři reaktivní datová úložiště pomocí nástroje Pinia (viz v kapitole 3.3), které slouží k ukládání a správě dat v aplikaci. Každé z datových úložišť je použito pro správu jiné skupiny dat a jsou logicky rozděleny tak, aby se jejich množiny dat nepřekrývaly.

Datová úložiště se nazývají `Global`, `Q-Sort` a `Settings`.

Jsou umístěny v adresáři `src/stores`, každý v samostatném souboru. K vytvoření reaktivního datového úložiště je použita funkce `defineStore` nástroje Pinia, které je jako první parametr posílán název úložiště a jako druhý parametr posílána funkce obsahující implementaci datového úložiště. Funkce `defineStore` po volání vrací referenci na objekt datového úložiště. Pomocí této reference je možné přistupovat k jednotlivým atributům a metodám úložiště.

6.1.1 Datové úložiště `Global`

Datové úložiště `Global` slouží k uchovávání datových sad a stará se o správu animovaných přechodů v aplikaci. Kromě toho slouží také k ukládání speciálních stavů, které jsou používány k úpravě chování knihovny `vue-draggable-plus` 3.5. Nachází se v souboru `src/stores/global.js`.

6.1.2 Datové úložiště `Settings`

Toto datové úložiště ukládá informace o celkovém nastavení aplikace v rámci jednoho sezení. Obsahuje informace o viditelnosti minimapy, o zapnutí či vypnutí automatického ukládání průběhu řazení a informaci o zobrazení otázky ve vrchním panelu aplikace. Kromě těchto informací datové úložiště obsahuje metody pro uložení nastavení aplikace do lokálního úložiště (local storage), aby uživatel nemusel s otevřením nového okna nastavovat aplikaci znovu.

6.1.3 Datové úložiště `Q-Sort`

Toto datové úložiště je ze všech zmíněných datových úložišť nejkomplexnější, jelikož obsahuje všechna data a metody týkající se samotného řazení. V tomto datovém úložišti je

```
<VueDraggable
    class="absolute left-0 top-0 z-[1] flex h-full w-full items-center"
    v-model="row.cards"
    group="sorting"
    @change="onChangeHandler"
    @add="addHandler"
    @start="onStartHandler"
>
<!-- Seznam přesouvatelných elementů -->
</VueDraggable>
```

Výpis 6.1: V této ukázce je zobrazeno použití komponenty `VueDraggable` z knihovny `vue-draggable-plus` k vytvoření přesouvatelných prvků v aplikaci. Atributem `v-model` se komponentě předává seznam prvků, u kterých je potřeba umožnit přetahování a pomocí `group` atributu lze implementovat přetahování prvků mezi jednotlivými seznamy pomocí stejného názvu. Metody `@change`, `@add` a `@start` umožňují předat funkce, které budou volány v reakci na událost vyvolanou uživatelem.

uchováván objekt, který obsahuje všechna data potřebná pro řazení. Obsahuje také všechny metody, kterými je možno získávat nebo upravovat obsah plochy pro řazení. Kód reprezentující toto úložiště se nachází v souboru `src/stores/q-sort.js`. Při velkém implementování změn v aplikaci bylo potřeba odstranit některé funkce tohoto datového úložiště a nahradit je novými. Bylo tak učiněno primárně z toho důvodu, že původní řešení přesunu tvrzení vyžadovalo více režie, zatímco momentální řešení využívá pro většinu případů použití přímo funkcionality knihovny `vue-draggable-plus` 3.5.

6.2 Implementace drag and drop funkcionality v aplikaci pro Q-metodologii

Komponenta VueDraggable je použita v komponentách `NewRowElement.vue`, `Row.vue`, `PresortCardQueue.vue` a `CardQueue.vue`. Všechny tyto komponenty totiž vyžadují použití drag and drop funkcionality. Pro zajištění správného fungování drag and drop přesouvání prvků mezi těmito komponentami je klíčové propojení pomocí atributu group. Tento atribut umožňuje definovat, které seznamy nebo kolekce jsou navzájem propojeny, což umožňuje přetahování položek z jednoho seznamu do druhého. Bez tohoto propojení by uživatelé nemohli efektivně přesouvat položky mezi seznamy.

Při přesunu tvrzení z fronty do pole se přiřazení prvku do kategorie projeví také v komponentě minimapy, kterou lze vidět v pravé části aplikace (obrázek 6.3). Implementaci komponenty pro minimapu bylo potřeba lehce upravit z důvodu změny struktury datového objektu pro ukládání prvků v poli oproti původní implementaci aplikace. drag and drop funkcionalita na rozdíl od původní implementace aplikace umožňuje dočasně vložit do jedné kategorie více tvrzení, než má povoleno. V tomto případě ale řazení nepůjde odeslat a respondent musí tvrzení přeřadit tak, aby splňovala kritéria pro řazení popsaná v sekci 2.1.

6.3 Implementace funkcionality předřazení pomocí barevných značek

```
<template>
    <div class="absolute right-2 flex gap-1 hover:cursor-pointer">
    <div
      v-for="row in qStore.table.filter(
        row =>
          (row.color !== undefined && tagsEditable) ||
          (row.color && qStore.cardContainsTag(cardId,row.color))
      )"
      :key="row.color"
      @click="onClickHandler(row)"
      :style="{ backgroundColor: row.color }"
      :class="classObject(row)"
    ></div>
  </div>
</template>

...
```

Výpis 6.2: Ukázka komponenty `CardTags.vue`, která slouží k zobrazení značek u jednotlivých karet. Značky, které se mají zobrazit se získávají z barev přiřazených k jednotlivým řádkům v ploše pro řazení. Pomocí vlastnosti `tagsEditable`, která se získává z rodičovské komponenty se určuje, zda-li může uživatel aktivovat či deaktivovat značky, či nikoliv a také se pomocí této vlastnosti nastavujou styly jednotlivých značek. Pokud je značka neaktivní, má přerušovaný okraj a má zvýšenou průhlednost. Aktivní značka má plný okraj a není vůbec průhledná. Funkce `classObject` vrací objektovou reprezentaci tříd pro daný řádek.

Pro zobrazení barevných značek u kartičky je použita komponenta `CardTags.vue` (viz výpis 6.2). Tato komponenta přijímá ze své rodičovské komponenty `Card.vue` vlastnost `tagsEditable`, která určuje, jestli budou značky u karty aktivovatelné a deaktivovatelné. Dostupné značky v této komponentě se berou přímo z barev jednotlivých řádků z proměnné `table` v uložišti `q-sort`. Pomocí metody `cardContainsTag` z uložiště `q-sort` komponenta zjistí, jestli je příslušná značka aktivní u dané karty, či nikoliv, a podle toho nastaví styly elementu značky. Aktivní značky mají plný okraj a nulovou průhlednost, neaktivní značky mají nastavený přerušovaný okraj a 40% neprůhlednost.

6.4 Implementace sdílení průběhu řazení pomocí websockets

Po konzultaci s vedoucím práce bylo zjištěno, že by bylo vhodné přidat do aplikace sdílení průběhu řazení mezi více zařízeními pomocí websocketové komunikace. Tato funkcionalita je žádoucí hlavně z důvodu, aby vedoucí výzkumu mohl živě sledovat ze svého zařízení, jak respondent provádí výzkum na jiném zařízení.

Popis relace v aplikaci

Relací se v této aplikaci myslí koncept sdílení průběhu řazení v aplikaci mezi zařízeními, které se připojí ke stejné relaci pomocí unikátního identifikátoru v odkazu.

Pro jednoznačnou identifikaci relace aplikace používá v případě statických datových sad dvojici parametrů `datasetId`, `uid`, které se nachází v cestě URL adresy obrazovky pro řazení, která má následující schéma: `/Sorting/:datasetId/:uid`. Parametr `datasetId` je číselný identifikátor datové sady aplikace a `uid` parametr je 36 znaků dlouhý identifikátor, který je generovaný při příchodu na stránku řazení a slouží jako identifikátor relace.

Varianta aplikace, která získává data z Q-panel rozhraní (viz sekci 6.6) používá k identifikaci relace parametr `sortingGuid`, který komponenta získává stejně jako v předchozím případě z cesty URL adresy `/SortingApi/:sortingGuid/`. Tyto parametry se při připojení k websocketovému serveru pošlou v parametrech dotazu a server si je uloží pro pozdější identifikaci relace.

6.4.1 Tvorba a použití websocketového klienta pro komunikaci

Pro implementaci websocketové komunikace na klientské straně je potřeba nejprve vytvořit klienta, který bude zprostředkovávat rozhraní pro komunikaci s websocketovým serverem. Klient pro websocketovou komunikaci, který je znázorněný ve výpise 6.3 byl implementován studentem Sebastiánem Krajňákem v rámci jeho diplomové práce a nachází se v souboru `socket.js` v `src` adresáři aplikace.

Tento soubor je implementován jako javascriptový modul, jehož výstupem je instance websocketového klienta, který se dále používá na všech místech v aplikaci, kde je potřeba využití websocketové komunikace.

Na začátku souboru je importována funkce `reactive` z knihovny `vue` a knihovna `io` z knihovny `socket.io-client`.

Funkce `reactive` umožňuje vytvářet reaktivní objekty, které jsou určeny pro sledování a reagování na změny hodnot v aplikaci. V tomto případě je reaktivní objekt použit pro správu informace o stavu připojení websocketového klienta k serveru pomocí hodnoty `connected`. V případě úspěšného připojení k websocketovému serveru se tato proměnná nastaví na hodnotu `true` a při jakémkoliv dotazu na websocketový server je nejprve provedena kontrola, jestli je připojení aktivní. Po odpojení klienta od websocketového serveru je hodnota této proměnné nastavena zpět na hodnotu `false`.

Funkce `io` umožňuje již výše zmiňovanou tvorbu instance websocketového klienta. Parametrem této funkce je adresa URL websocketového serveru, která se nastavuje na základě proměnné prostředí `NODE_ENV`. Hodnota této proměnné udává, jestli je aplikace spuštěna v dockerovém prostředí, či nikoliv. V případě dockeru se totiž pro komunikaci mezi dvěma kontejnery používá URL adresa obsahující název dockerového kontejneru, ve kterém se služba nachází.

Výsledek této funkce je přiřazen do proměnné `socket`, která obsahuje výslednou instanci websocketového klienta. Po vytvoření klienta se na server odešle zpráva „hello“ pomocí metody `emit`. Tato zpráva je následně zpracována na straně serveru a server pomocí této zprávy pozná, že s ním klient navázal spojení.

Poté už jsou v klientovi pomocí metody `on` definovány akce, které se mají stát v případě příchozí zprávy od serveru. Při příchodu zprávy „connect“ se nastaví hodnota výše zmíněné proměnné `connected` na hodnotu `true` a při příchodu zprávy „disconnect“ od serveru se nastaví hodnota této proměnné na hodnotu `false`.

6.5 Umožnění přesažení limitu karet v kategorii

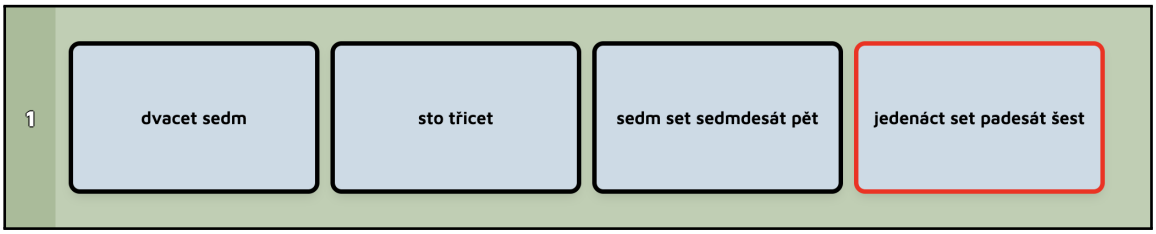
Po implementaci drag and drop funkcionality (popsáno v kapitole 6.2) do aplikace je uživateli umožněno dočasně vložit do kategorie více tvrzení, než je její maximální povolený počet. Tuto funkcionalitu jsem se rozhodl implementovat, protože uživatelé bez implementování této funkcionality museli tvrzení z plné kategorie nejdříve přesunout zpět do fronty, aby mohli do kategorie vložit tvrzení nové.

```
import { io } from "socket.io-client";

...

const server_URL =
  process.env.NODE_ENV === "docker" ? "q-info.fit.vutbr.cz:3000" : "localhost:3000";

const socket = io(server_URL, { transports: ["websocket"] });

socket.on("connect", () => {
  state.connected = true;

  eventQueue.forEach(({ event, data }) => {
    safeEmit(event, data);
  });
});

socket.on("disconnect", () => {
  state.connected = false;
});

export const safeEmit = (event, data) => {
  if (state.connected) {
    socket.emit(event, data);
  } else {
    eventQueue.push({ event, data });
  }
};

export default socket;
```

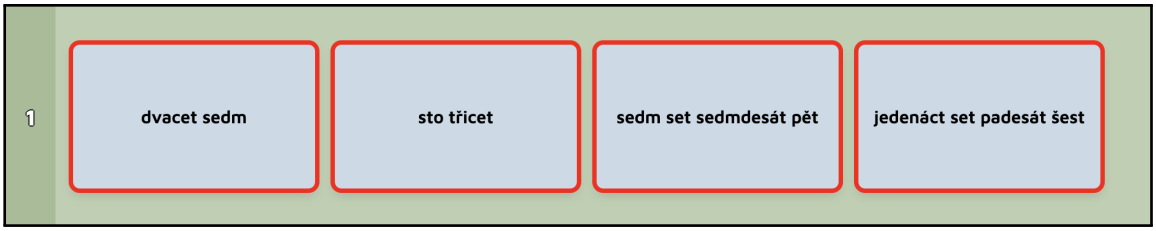
Výpis 6.3: Ukázka souboru `socket.js` obsahující implementaci klienta pro připojení k websocket serveru implementovaného studentem Sebastiánem Krajňákem. K vytvoření instance websocketového klienta slouží funkce `io` z knihovny `socket.io-client`. Na instanci klienta jsou hned po vytvoření připojeny obslužné funkce pro akce `connect` a `disconnect`. Funkce safeEmit slouží k bezpečnému odeslání akce na websocketový server. V případě, že by došlo k výpadku serveru, akce se ukládají do fronty a po opětovném připojení se odešlou.

Problém ale nastával v tom, že uživatelé při řazení nebyli upozorněni na přesáhnutí maximálního počtu tvrzení v kategorii. Kvůli tomu docházelo k situacím, kdy aplikace nedovolila uživateli dokončit řazení z důvodu přesažení maximálního počtu v některé z kategorií. Proto bylo potřeba vymyslet, jak uživatele upozornit na to, že přesáhl maximální počet tvrzení v kategorii a musí před odevzdáním obsah plochy upravit.

Tato funkcionalita byla nakonec implementována způsobem zvýraznění všech karet v kategorii, ve které byl limit přesažen (obrázek 6.2). Při používání první verze implementace této funkcionality (obrázek 6.1) si totiž uživatelé mysleli, že červeně zvýrazněná karta nad limitem kategorie nemůže být vložena do kategorie z důvodu nevhodnosti karty, nikoliv kvůli přesažení limitu karet kategorie. To uživateli přivádělo pocit, že je chyba v jedné jediné kartě, i když chyba byla vlastně v celé kategorii. Proto bylo nakonec zvoleno řešení pomocí zvýraznění všech karet v kategorii, u které byl přesažen limit tvrzení.



Obrázek 6.1: První verze funkcionality pro zobrazení maximálního počtu tvrzení v kategorii. Při přesažení maximálního počtu tvrzení v kategorii dojde pouze ke zvýraznění těch tvrzení, která byla vložena přes limit kategorie. V tomto případě má kategorie můžou být do kategorie vloženy maximálně tři tvrzení, ale byla vložena čtyři tvrzení.



Obrázek 6.2: Druhá verze zobrazení přesažení maximálního počtu tvrzení v kategorii. Při přesažení maximálního počtu tvrzení v kategorii dojde k zvýraznění všech tvrzení v kategorii pomocí červeného okraje.

6.6 Napojení aplikace pro řazení na rozhraní aplikace Q-panel

Q-panel je aplikace vyvíjená studentem Nikitou Pasynkovem [18] v rámci jeho bakalářské práce, která slouží ke správě studií, datových sad, uživatelů, kteří mají provádět řazení atd. Aplikace nabízí rozhraní postavené na architektuře REST [20], které umožňuje přístup k backendovým službám aplikace[1].

V aplikaci je vytvořená nová cesta, která vede na komponentu řazení se stahováním dat z rozhraní aplikace Q-panel. Cesta k této komponentě vypadá `/SortingApi/sortingGuid`. Na rozdíl od cesty, kterou používá varianta aplikace se statickými datovými sadami, používá tato cesta segment `SortingApi` místo segmentu `Sorting` a také obsahuje pouze jeden parametr `sortingGuid` což je jednoznačný identifikátor řazení, který je vygenerován aplikací Q-panel a používá se pro veškerou komunikaci s rozhraním.

Rozhraní je používáno k získávání datových sad pro řazení. V hlavní nabídce, která je dostupná na adrese `http://q-info.fit.vutbr.cz:8080/xpavel41/` se kromě seznamu statických datových sad, které jsou uloženy v aplikaci, nachází také políčko určené pro vložení unikátního identifikátoru řazení, který lze vygenerovat v grafickém rozhraní aplikace Q-panel. Tento unikátní identifikátor (v aplikaci nazváno `sortingGuid`) slouží k jednoznačné identifikaci řazení.

Kromě získávání datových sad se toto rozhraní používá i pro ukládání a získávání pokroku při řazení. Konkrétně vždy po provedení pěti přesunů karet v aplikaci dojde k uložení postupu na server. Rozpracovaný postup v řazení je poté možné získat opětovným připoje-

[1] Popis rozhraní aplikace Q-panel dostupný zde `https://backend.qpanel.settler.tech/docs#/`

```
const { get, data, error } = useApi();

await get(`/round-links/${sortingGuid}/progress`);

if(error) handleError(error);
if(data) handleDataFetched(data);
...
```

Výpis 6.4: Ukázka získávání dat pomocí hooku `useApi`, který po zavolání vrací metodu `get`, určenou pro odeslání GET HTTP požadavku na parametrizovanou adresu `/round-links/sortingGuid/progress`. Proměnná `data` obsahuje odpověď serveru a proměnná `error` obsahuje případné chybové hlášky, které server vrátí.

ním k řazení zadáním identifikátoru v hlavní nabídce aplikace, nebo přímým přístupem na URL adresu s identifikátorem.

V aplikaci se při komunikaci s rozhraním volají tři metody Q-panel rozhraní:

- `GET /round-links/link/cardset` – Slouží pro získání datové sady pro řazení. Obsahuje všechny kartičky, strukturu plochy, barevné značky atd.
- `GET /round-links/link/progress` – Slouží pro získání postupu v rámci řazení. Obsahuje struktury karet a plochy ve stejném formátu, jako jsou uloženy v samotné aplikaci přo řazení.
- `POST /round-links/link/progress` – Slouží pro uložení postupu na server. Všechny data se rozhraní předávají v atributu `data`, který může obsahovat libovolnou strukturu dat.

Pro odesílání HTTP dotazů na rozhraní aplikace Q-panel jsem si v aplikaci vytvořil vlastní hook `useApi`. Hook je funkce, která obsahuje reaktivní hodnoty a logiku, která je v aplikaci používána opakovaně na více místech. Hook `useApi` po zavolání vrací objekt, který obsahuje obslužné funkce pro všechny potřebné HTTP metody `get()`, `post()`, `put()`, `delete()` a stavové hodnoty `data`, `error`, `loading`. Ke komunikaci s rozhraním je použita knihovna Axios (viz v kapitole 3.7).

Pro ukázku získávání dat z Q-panel rozhraní jsem v grafickém rozhraní aplikace Q-panel vytvořil dvě studie pro řazení. První se zaměřuje na seřazení povinných předmětů bakalářského studia na VUT FIT. Druhá se zabývá seřazením jídel podle oblíbenosti. Datové sady byly použity i při iterativním testování. Odkazy na obě studie lze najít ve spodní části hlavní nabídky na úvodní stránce aplikace.

6.7 Načítání a ukládání dat pro řazení

V aplikaci se nachází dvě varianty plochy pro řazení. První varianta používá pro řazení statické datové sady uložené přímo v adresáři projektu. Druhá varianta používá pro řazení datové sady získané z Q-panel rozhraní (viz sekci 6.6) a toto rozhraní také používá pro průběžné ukládání postupu při řazení. Celkově je ale proces získávání dat v obou variantách lehce komplikovanější, jelikož je potřeba synchronizovat data také s websocketovým serverem (viz sekci 6.4 a lokálním úložištěm prohlížeče, do něhož se data ukládají pro případ výpadku websocketového serveru.

Aplikace má všechna data používaná k řazení uložena v reaktivním úložišti `q-sort` (viz sekci 6.1.3) v následujících atributech.

- `table` – Struktura celé plochy pro řazení. Jedná se o pole řádků, kde každý řádek obsahuje informace o vložených kartách, maximálním možném počtu karet, barvě řádku, o tom jestli jde o původní řádek plochy, nebo přidaný dočasný řádek (viz sekci 5.4) atd.
- `queue` – Pole kartiček, které se zobrazují ve frontě pro řazení.
- `cardDeck` – Pole kartiček stejné jako v atributu `queue`. Z atributu `cardDeck` se ale kartičky při přesunu z fronty neodstraňují a slouží tak jako informace o všech dostupných kartách při řazení.
- `addableRows` – Pole identifikátorů řádků, nad kterými se má zobrazit element pro přidání dočasné kategorie (viz sekci 5.4). Pokud se má zobrazit element i za posledním řádkem tabulky, nachází se zde řetězec „end“.
- `availableTags` – Seznam všech dostupných značek v aplikaci (viz sekci 5.3.3).
- `question` – Otázka zobrazená ve vrchním panelu při řazení.
- `name` – Název datové sady

Ukládání dat aplikace

Ukládání dat aplikace se provádí pouze, pokud má uživatel v nastavení povolené ukládání postupu (viz obrázek 5.12b). Pokud uživatel ukládání deaktivuje, nebude se mu postup ukládat ani na jedno z výše zmíněných úložišť. Pokud poté obnoví okno prohlížeče, bude jeho postup řazení nenávratně ztracen a musí řazení provést znovu.

Ve variantě aplikace, která používá statické datové sady, se data ukládají do *localStorage* a posílají se pomocí websocketové komunikace (viz sekci 6.4) k uložení do *MongoDB* databáze.

Ve variantě aplikace, která získává datové sady z Q-panel rozhraní, se data ukládají také do *localStorage* a *MongoDB* databáze a navíc se také posílají pomocí Q-panel rozhraní k uložení do databáze systému Q-panel.

Při ukládání dat do *localStorage* a do *MongoDB* databáze pomocí websocketového serveru jsou data nejprve serializována do formátu JSON.

Načítání dat aplikace

Načítání dat aplikace je lehce komplikovanější, než ukládání dat do jednotlivých úložišť. Je totiž potřeba synchronizovat data z jednotlivých úložišť a do aplikace načíst ta data, která jsou nejaktuálnější. K tomu slouží atribut v ukládané datové struktuře nazvaný `updated`, který je datového typu `Date` a udává, kdy byla data uložena do úložiště. Při načítání dat do aplikace se porovnávají tyto atributy mezi datovými objekty z jednotlivých úložišť. Data s nejnovějším datem a časem úpravy se vyberou a jsou načtena do aplikace.

V obou variantách aplikace se postup načítá pouze v případě, že má uživatel povolené ukládání postupu. V opačném případě se načítají výchozí data pro řazení bez postupu.

Pro variantu aplikace s napojením na Q-panel rozhraní se data načítají pomocí metody `loadDataset` z uložiště `q-sort`, a pro variantu se statickými datasety pomocí metody

```
FROM node:20-slim
WORKDIR /frontend-q-sort
EXPOSE 8080
RUN npm install -g npm
RUN npm audit fix
```

Výpis 6.5: Ukázka souboru `Dockerfile-frontend`, který popisuje jednotlivé kroky procesu tvorby dockerového obrazu aplikace. Na začátku je pomocí příkazu `FROM` definován základní obraz node:20-slim, který obsahuje všechny potřebné závislosti pro běh aplikace. Přípona „slim“ znamená, že obraz je optimalizován pro menší velikost výsledného obrazu. Příkaz `WORKDIR` nastavuje pracovní adresář uvnitř kontejneru na adresář `/frontend-q-sort`, ve kterém se budou spouštět všechny následující příkazy. `EXPOSE` příkaz označuje, že aplikace uvnitř kontejneru bude naslouchat na portu 8080. Příkaz `RUN` spouští příkaz ve fázi sestavování dockerového obrazu. Pomocí tohoto příkazu se nainstaluje nejnovější verze nástroje `npm` [29], který slouží k instalaci závislostí Node.js [28] aplikací.

`loadDatasetFromApi`. Tyto metody jsou volány před připojením obrazovky pro řazení do DOM.

Obrazovky pro řazení jsou obaleny komponentou `LoadingContainer.vue`, která se stará o to, aby se obrazovka zobrazila až potom, co se načtou data. Do té doby zobrazuje komponentu točícího se kolečka, upozorňujícího uživatele na načítání dat.

Pro načítání dat z jednotlivých externích datových úložišť jsou v datovém úložišti `q-sort` (viz sekci 6.1.3) vytvořeny čtyři metody `loadLocalStorageData`, `loadJsonDataset`, `loadWebsocketData`, `loadApiData`. Těmto metodám se předávají datové struktury získané z příslušného datového úložiště a metody se postarají o správné uložení dat ve správném formátu do datového úložiště aplikace nebo externího úložiště.

6.8 Nasazení aplikace pomocí nástroje Docker

Pro sestavení a nasazení aplikace na server jsem zvolil použití nástrojů `Docker` a `Docker Compose`, které jsou popsány v kapitole 3.9. K tomuto kroku jsem se rozhodl primárně z důvodu jednoduchosti, jelikož tato aplikace nevyžaduje žádné komplikované postupy při sestavování nebo nasazování, a tak je `Docker` a `Docker Compose` ideální volbou. Všechny soubory obsahující popisy potřebné pro nasazení aplikace pomocí nástroje `Docker` se nacházejí v projektu v adresáři `docker`.

V rámci sestavování projektu pomocí nástroje `Docker` jsem řešil i sestavení a nasazení websocketového serveru (popsáno v kapitole 6.4 a MongoDB databázového serveru, který websocketový server vyžaduje. Oba tyto servery byly navrženy a vyvinuty studentem Sebastiánem Krajňákem [12]. Já jsem řešil pouze lehké úpravy websocketového serveru a následné sestavení a nasazení. Celý proces nasazení aplikace na server `q-info` se skládá z následujících kroků:

1. **Vytvoření souboru Dockerfile-frontend** – Prvním krokem při sestavování aplikace bylo vytvoření souboru `Dockerfile`, který slouží jako předpis pro vytvoření dockerového obrazu aplikace. Ukázka souboru `Dockerfile-frontend` včetně jeho popisu je ve výpise 6.5. `Dockerfile` ostatních spouštěných aplikací v rámci celého projektu lze najít v adresáři `docker`.

```yaml
services:
  frontend:
    build:
      context: ../
      dockerfile: docker/Dockerfile-frontend
    container_name: q-info-frontend-container
    environment:
      NODE_ENV: docker
    ports:
      - '8080:8080'
    command: sh -c "npm install && npm run dev"
    networks:
      q-info-network:
        ipv4_address: 172.20.0.4
networks:
  q-info-network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/16
```

Výpis 6.6: Ukázka souboru `docker-compose.yaml` definujícího spuštění kontejneru pro službu s názvem `frontend`. V sekci `build` je předepsáno, aby kontejner při vytvoření používal obraz sestavený podle definice v souboru `./Dockerfile-frontend` (viz výpis 6.5). Parametrem `container_name` je vytvořený kontejner pojmenován `q-info-frontend-container`. Pomocí `environment` lze nastavit systémové proměnné prostředí uvnitř kontejnteru. Sekce `ports` popisuje přesměrování vnitřního portu kontejneru `8080` na port `3000` v prostředí, ve kterém kontejner běží. Parametr `command` definuje příkaz, který se zavolá po spuštění kontejneru a sekce `network` připojí kontejner po vytvoření do sítě `q-info-network` a přidělí kontejneru IP adresu `172.20.0.4`. Vytvoření této sítě je popsáno v sekci `networks` na stejné úrovni, jako sekce `services`. V této ukázce je zobrazeno pouze vytváření kontejneru `q-info-frontend`, v souboru `docker\docker-compose.yaml` jsou však definovány i další aplikace spouštěné v rámci projektu.

2. **Vytvoření souboru docker-compose.yaml** – Poté, co se mi podařilo vytvořit soubor `Dockerfile`, bylo potřeba definovat soubor `docker-compose.yaml`, který slouží ke spuštění všech kontejnerů v rámci projektu. Ukázka souboru `docker-compose.yaml` včetně jeho popisu je ve výpise 6.6.

3. **Nahrání projektu na server** – Po vytvoření všech souborů potřebných pro vytvoření obrazu i kontejneru bylo potřeba tyto soubory včetně zdrojových kódů projektu nahrát na server, na kterém má být aplikace spuštěna. V tomto případě se jedná o server `q-info.fit.vutbr.cz`, na kterém již byly nainstalovány nástroje `Docker` a `Docker Compose`. Adresáře `docker` a `q-sort-app` jsem nahrál na server pomocí protokolu SFTP [9] (SSH File Transfer Protocol).

4. **Spuštění příkazu docker compose** – Poté, co se soubory nahrály na server už zbýval pouze jeden krok a to samotné spuštění příkazu `docker compose up -d` v adresáři `docker`. Spuštěním tohoto příkazu se sestaví dockerové obrazy všech aplikací definovaných v souboru `docker-compose.yaml` a z toho se následně vytvoří kontejnery s aplikacemi a spustí se. Aplikace běží na adrese `http://q-info.fit.vutbr.cz:8080/`.

V kořenovém adresáři projektu se nachází skripty `setup_docker.bat` (pro Windows) a `setup_docker.sh` (pro Unix). Tyto skripty slouží k zjednodušení procesu sestavení a spuštění aplikace. Před sestavením a spuštěním aplikace tyto skripty ještě vymažou všechny obrazy, kontejnery a sítě, které by mohly způsobovat kolize v názvech.

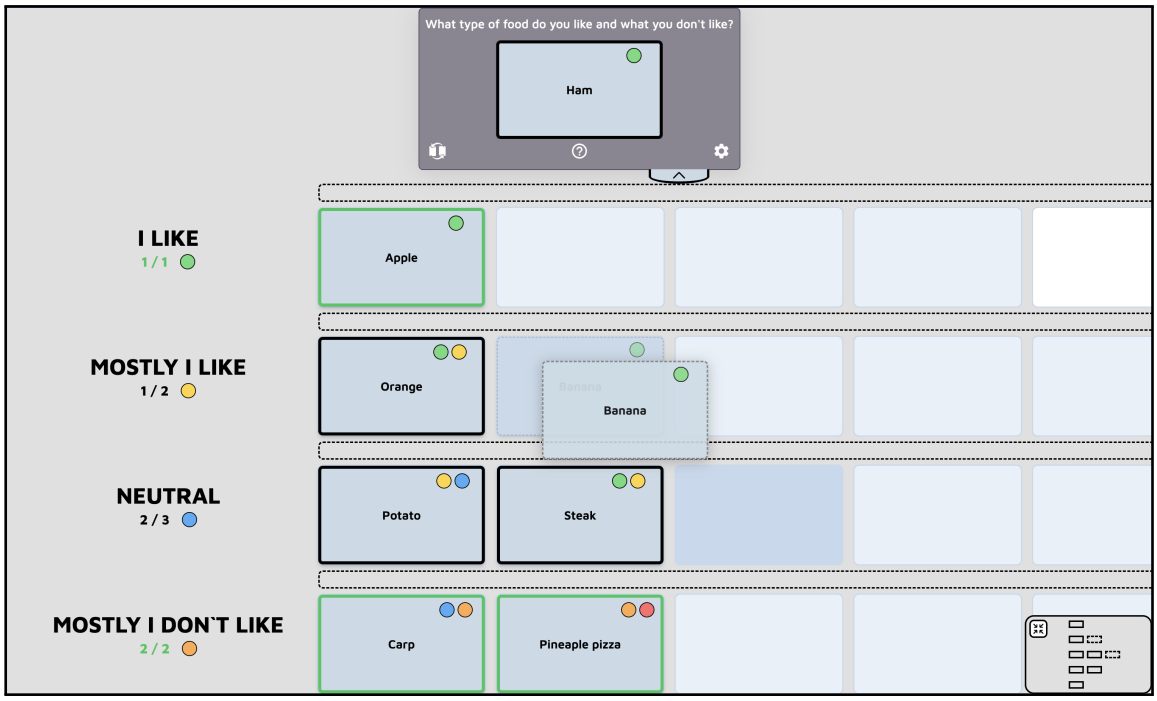
6.8.1 Lokální vytvoření obrazu

V některých případech, kdy je například na cílovém serveru málo úložného prostoru, může nastat situace, kdy je potřeba sestavovat dockerové obrazy aplikací lokálně a ty poté nahrát na server a následně z nich spustit kontejnery. Díky tomuto řešení není potřeba na server kopírovat zdrojové kódy a stačí pouze nakopírovat na server archiv ve formátu `tar`, který obsahuje vyexportované dockerové obrazy, načíst je a použít je ke spuštění kontejnerů.

Pro tento případ jsem vytvořil skripty, které při spuštění zavolají příslušné příkazy nástroje `Docker`. V kořenovém adresáři projektu se nachází celkem šest skriptů (tři ve formátu `sh` pro UNIXové systémy a tři soubory ve formátu `bat` pro systém Windows).

Proces spuštění skriptů pro lokální sestavení a následné spuštění kontejnerů na serveru se skládá z následujících kroků:

- **Sestavení a uložení obrazů** – Pro sestavení a uložení obrazů všech aplikací projektu je potřeba spustit skript `build_save.sh` (pro Unix), nebo `build_save.bat` (pro Windows). Tento skript sestaví obrazy pomocí dockerového příkazu `docker build`. Poté spustí příkaz `docker save`, který všechny tyto sestavené obrazy uloží do archivu s názvem `q-info.tar`.
- **Nahrání souborů na server** – Po sestavení všech obrazů je potřeba nahrát soubor `q-info.tar` a skripty `load_run.sh` a `clear.sh` (pro Unix) nebo `load_run.bat` a `clear.bat` (pro Windows) na příslušný server, na kterém je potřeba aplikaci spustit. K tomu může být opět použit nástroj SFTP (SSH File Transfer Protocol).
- **Načtení obrazu a spuštění kontejneru** – Po nahrání archivu na server je potřeba na server ještě nahrát do stejného adresáře také skript `load_run.sh`, pokud jde o UNIXový server, nebo skript `load_run.bat` v případě, že jde o Windows server. Tento skript po spuštění zavolá příkaz `docker load`, který načte obrazy z archivu `q-info.tar` do nástroje `Docker`. Následně se spustí příkazy `docker run` se všemi obrazy získanými z archivu a spustí pomocí nich kontejnery.



Obrázek 6.3: Ukázka výsledné verze aplikace po implementaci všech změn a vylepšení, které vyšly z iterativního testování popsaného v kapitole 4. Ve vrchní části se nachází panel s frontou karet určených pro řazení. Pomocí tlačítek na spodní straně panelu lze přepínat typ panelu (klasický panel, nebo panel pro předřazení karet), zobrazit, nebo schovat otázku a nebo otevřít panel s nastavením. Panel lze také zavřít pomocí tlačítka s šipkou nahoru na spodní hraně panelu. Pod panelem se nachází plocha pro řazení s jednotlivými kategoriemi, které již obsahují některá tvrzení. Tyto karty jsou navíc označeny barevnými značkami znázorňujícími barvy jednotlivých kategorií. V pravé spodní části obrazovky se nachází minimapa pro lepší přehled v řazení.

Kapitola 7

Závěr

V této práci jsem úspěšně realizoval všechny kroky ze zadání a výsledkem je intuitivní a přehledná aplikace. pro provádění řazení pomocí Q-metodologie. Iterativní testování s uživateli přineslo mnoho užitečných poznatků, které vedly k zlepšení použitelnosti a uživatelské přívětivosti aplikace. Při jednotlivých iteracích testování jsem vytvořil různé verze vylepšení, které jsem průběžně testoval a snažil se vylepšovat k dokonalosti. Hlavními vylepšeními, které přispěly ke zvýšení uživatelské přívětivosti a použitelnosti, jsou drag and drop funkcionalita pro přesouvání prvků (popsáno v kapitole 5.2), možnost předřazení tvrzení pomocí barevných značek (popsáno v kapitole 5.3), možnost dočasného přidání více karet do kategorie, než je její limit (popsáno v kapitole 6.5) a možnost vytváření dočasných kategorií do plochy pro odkládání karet (popsáno v kapitole 5.4).

Kromě těchto funkcionalit se mi povedlo úspěšně nad rámec zadání implementovat také napojení aplikace na websocketový server (popsáno v kapitole 6.4), který slouží ke komunikaci v reálném čase mezi instancemi aplikace otevřenými na různých zařízeních, a také napojení na rozhraní aplikace Q-panel, sloužící ke získávání datových sad a získávání a ukládání postupu při ražení (popsáno v kapitole 6.6). Podařilo se mi i mimo jiné zprovoznit nasazování aplikace na server q-info pomocí nástroje `docker` a vytvořit skripty, které zjednodušují proces nasazení (popsáno v kapitole 6.8).

Aplikace byla vyvíjena tak, aby byla ideálně použitelná i na malých mobilních zařízeních. To se povedlo do takové míry, že všechny funkcionality aplikace jsou funkční i na mobilních zařízeních využívajících dotykovou obrazovku. Doporučuji ale aplikaci používat spíše na tabletu nebo laptopu. Aplikace má totiž na malých mobilních zařízeních málo zobrazovacího prostoru a snižuje se tím její přehlednost a použitelnost.

Aplikace byla testována v prohlížečích Google Chrome, Mozilla Firefox, Safari a Opera.

Celkově jsou ale na aplikaci velice pozitivní ohlasy od uživatelů, se kterými jsem prováděl testování. Jde vidět, že je pro ně proces řazení jednodušší než při používání původní aplikace. Nejvíce uživatelé ocenili drag and drop funkcionalitu a možnost předřazení karet pomocí barevných značek. Vzhledem k tomu, že tohle byla moje první zkušenost s iterativním testováním aplikace, myslím si, že jsem odvedl dobrou práci a vytvořil jsem kvalitní aplikaci, která uživatelům nabízí jednoduché a intuitivní používání.

Při implementaci websocketové komunikace byla v původním návrhu také funkcionalita, kde by uživatel na různých připojených zařízeních kromě přesunutých karet v cílových políčkách viděl také samotné přesouvání karet mezi políčky v ploše. Z tohoto nápadu ale nakonec sešlo z důvodu nedostatku času a také kvůli vysokým požadavkům na výkon serveru. Nicméně tato funkcionalita by mohla zpříjemnit používání aplikace mezi více zařízeními, a tak je to jedna z možností, jak ve vývoji aplikace pokračovat.

Při finálním testování aplikace bylo navíc zjištěno, že funkcionalita automatického posunu plochy pro řazení při drag and drop přesunu nefunguje napříč všemi prohlížeči stejně. V prohlížečích Chrome a Safari tato funkcionalita funguje docela spolehlivě, ale prohlížeče Opera a Firefox tuto funkcionalitu nepodporují. Proto by možným pokračováním mohl být také vývoj vlastního mechanismu pro posouvání plochy při přesouvání prvků pomocí drag and dropu.

Dalším možným pokračováním v aplikaci by mohlo být zapracování algoritmů pro určování optimálního rozložení karet v ploše při přidávání dalších kategorií, které navrhl vedoucí práce prof. Ing. Adam Herout, Ph.D.

Výsledná aplikace vytvořená v rámci této práce je dostupná na URL adrese `https://q-info.fit.vutbr.cz:8080/xpavel41/`

V rámci této práce jsem vytvořil také ukázkový plakát a video z používání aplikace, které jsou součástí odevzdaných souborů aplikace.

Literatura

[1] Axios Project. *Axios Documentation – Introduction* [online]. [cit. 2025-04-07]. Dostupné z: https://axios-http.com/docs/intro.

[2] Crockford, D. *The JSON Data Interchange Format* [online]. [cit. 2025-04-07]. Dostupné z: https://www.json.org/json-en.html.

[3] donmbelembe. *Vue-dragscroll* [online]. [cit. 2025-04-02]. Dostupné z: https://vue-dragscroll.clebinfosys.com/.

[4] Eduardo San Martin Morote et al.. *Pinia – The Intuitive Store for Vue.js* [online]. [cit. 2025-04-02]. Dostupné z: https://pinia.vuejs.org/.

[5] Eseme, S. *Architecting Vue.js 3 Enterprise-Ready Web Applications: Build and deliver scalable and high-performance, enterprise-ready applications with Vue and JavaScript*. 1st. Packt Publishing, 2023. ISBN 9781801071734.

[6] Foundation, T. A. S. *Apache HTTP Server 2.2 Official Documentation – Volume I. Server Administration*. 1st. Fultus Corporation, 2010. ISBN 9781596821910.

[7] FullHuman. *PurgeCSS: Remove unused CSS* [online]. [cit. 2025-04-02]. Dostupné z: https://purgecss.com.

[8] Ghosh, S. *Docker Demystified: Learn How to Develop and Deploy Applications Using Docker*. 1st. Bpb Publications, 2020. Demystified Series. ISBN 9789389845877.

[9] Gillis, A. S. *Secure File Transfer Protocol (SSH File Transfer Protocol)* [online]. [cit. 2025-04-07]. Dostupné z: https://www.techtarget.com/searchcontentmanagement/definition/Secure-File-Transfer-Protocol-SSH-File-Transfer-Protocol.

[10] Goldberg, J. *Learning TypeScript*. 1st. O'Reilly Media, 2022. ISBN 9781098110284.

[11] Janů, M. *Q-sort-App* [online]. 2023 [cit. 2025-04-02]. Dostupné z: https://github.com/janumichal/Q-sort-App.

[12] Krajňák, S. *Výzkum a vývoj komunikační infrastruktury pro Q řazení* [online]. 2024. [cit. 2025-05-05]. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Adam Herout, P. prof. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=267544.

[13] Krug, S. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. 1st. Pearson Education, 2009. Voices That Matter. ISBN 9780321702845.

[14] Krug, S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3rd. Pearson Education, 2014. Voices That Matter. ISBN 9780321965516.

[15] Lombardi, A. *WebSocket: Lightweight Client-Server Communications*. 1st. O'Reilly Media, 2015. ISBN 9781449369255.

[16] Ltd., A. R. *Socket.IO: How it works, when to use it, and how to get started* [online]. [cit. 2025-04-12]. Dostupné z: https://ably.com/topic/socketio.

[17] MDN Web Docs. *Proxy – JavaScript | MDN* [online]. [cit. 2025-04-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy.

[18] Pasynkov, N. *Administrativní rozhraní pro Q řazení*. [online]. 2025. [cit. 2025-05-05]. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D. Práce bude teprve odevzdána. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/162958.

[19] Pirohová, K. *Výzkum uživatelských preferencí pro Q-řazení a návrh uživatelsky přívětivého interakčního modelu* [online]. 2025. [cit. 2025-05-05]. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D. Práce bude teprve odevzdána. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/163014.

[20] Red Hat. *What is a REST API?* [online]. [cit. 2025-04-02]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[21] RubaXa. *SortableJS — JavaScript library for reorderable drag-and-drop lists* [online]. [cit. 2025-04-02]. Dostupné z: https://sortablejs.github.io/Sortable/.

[22] Skyblue, A. *Vue-draggable-plus* [online]. [cit. 2025-04-02]. Dostupné z: https://github.com/Alfred-Skyblue/vue-draggable-plus.

[23] Socket.IO. *Client API | Socket.IO v3* [online]. [cit. 2025-04-12]. Dostupné z: https://socket.io/docs/v3/client-api/.

[24] Stephenson, W. *The Study of Behavior: Q-technique and Its Methodology*. University of Chicago Press, 1953. Midway reprint series. ISBN 9780226772783. Dostupné z: https://books.google.cz/books?id=TIKIxAEACAAJ.

[25] Tailwind Labs. *Tailwind CSS – Rapidly build modern websites without ever leaving your HTML* [online]. [cit. 2025-04-07]. Dostupné z: https://tailwindcss.com/.

[26] Vue.js Team. *Vue Router – The official router for Vue.js* [online]. [cit. 2025-04-07]. Dostupné z: https://router.vuejs.org/.

[27] Vue.js Team. *Vue.js – The Progressive JavaScript Framework* [online]. [cit. 2025-04-07]. Dostupné z: https://vuejs.org/.

[28] W3Schools. *What is Node.js?* [online]. [cit. 2025-04-07]. Dostupné z: https://www.w3schools.com/nodejs/nodejs_intro.asp.

[29] W3Schools. *What is npm?* [online]. [cit. 2025-04-07]. Dostupné z: https://www.w3schools.com/whatis/whatis_npm.asp.

[30] Watts, S. a Stenner, P. *Doing Q Methodological Research: Theory, Method & Interpretation*. Illustrated edition. SAGE Publications, 2012. Doing Q Methodological Research: Theory, Method and Interpretation. ISBN 9781849204156. Dostupné z: https://books.google.cz/books?id=PwfTZu-JEmoC.

[31] WikiSkripta. *Normální rozdělení*. No date [cit. 2025-04-02]. Dostupné z: https://www.wikiskripta.eu/w/Norm%C3%A1ln%C3%AD_rozd%C4%9Blen%C3%AD.

Příloha A

Adresářová struktura aplikace

- `docker` – Adresář se soubory pro spuštění aplikace v docker kontejneru
- `poster` – Adresář s plakátem o aplikaci
- `latex` – Adresář se zdrojovými soubory technické zprávy
- `video` – Adresář s videem z používání aplikace
- `q-sort-app` – Adresář se soubory aplikace
  - `src` – Adresář se zdrojovými soubory aplikace
    - `assets` – Adresář se statickými soubory aplikace
    - `components` – Adresář se všemi komponentami použitými v aplikaci
    - `enums` – Adresář s výčtovými typy aplikace
    - `fonts` – Adresář s písmy použitými v aplikaci
    - `router` – Adresář se směrovačem aplikace
    - `scss` – Adresář s SCSS styly aplikace
    - `stores` – Adresář s reaktivními úložišti aplikace
    - `utils` – Adresář s pomocnými funkcemi pro řazení
    - `views` – Adresář se všemi obrazovkami aplikace
    - `App.vue` – Hlavní komponenta aplikace vyvíjené ve Vue.js
    - `main.js` – Hlavní soubor Vue.js aplikace
    - `socket.js` – Soubor s implementací websocket klienta (Sebastián Krajňák)
  - `index.html` – Soubor, do kterého je vložena Vue.js aplikace
  - `package.json` – Soubor obsahující informace o závislostech aplikace
  - `package-lock.json` – Soubor obsahující informace o závislostech aplikace
  - `vite.config.js` – Konfigurační soubor nástroje Vite
- `server` – Adresář s implementací websocket serveru (Sebastián Krajňák)
- `README.md` – Soubor s návodem pro lokální spuštění aplikace
- `run_all.sh` – Skript pro spuštění aplikace v Linuxu
- `run_all.bat` – Skript pro spuštění aplikace ve Windows
- `clear.sh` – Skript pro vyčištění všech dockerových prvků aplikace v Linuxu
- `clear.bat` – Skript pro vyčištění všech dockerových prvků aplikace ve Windows
- `build_save.sh` – Skript pro sestavení a uložení dockerových obrazů v Linuxu
- `build_save.bat` – Skript pro sestavení a uložení dockerových obrazů ve Windows
- `load_run.sh` – Skript pro načtení obrazů ze souboru a spuštění kontejnerů v Linuxu
- `build_save.bat` – Skript pro načtení obrazů a spuštění kontejnerů ve Windows

Příloha B

Plakát