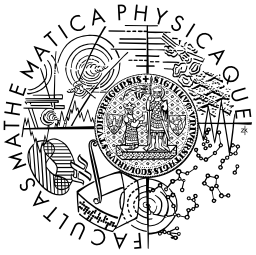


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

ROČNÍKOVÁ PRÁCE

Jméno Příjmení

Závěrečná zpráva z ročníkového projektu

Název katedry nebo ústavu

Vedoucí bakalářské práce: Vedoucí práce
Studijní program: studijní program
Studijní obor: studijní obor

Praha 2024

Obsah

Úvod

V rámci GAČR projektu Unified Management of Multi-Model Data (č. 20-22276S) byla navržena sada nástrojů pro modelování a správu multi-modelových dat, např. MM-cat a MM-evocat. Nástroje vznikaly postupně a za podpory různých lidí. Díky tomu nemají jednotné rozhraní pro práci. Navíc jsou současná rozhraní často nepřehledná a uživatelsky nepřívětivá. Proto se v rámci ročníkového projektu zabýváme návrhem sjednoceného rozhraní. Důraz je kladen na návrh vhodného uživatelského rozhraní a jeho udržovatelnost.

Práce je soustředěna hlavně na nástroj MM-evocat. A to tak, aby bylo možné rozšířit aplikaci o další nástroje. Výsledkem práce je návrh low-fidelity prototypu MM-evocat a jeho otestování na vybraném vzorku reálných uživatelů.

TODO: nahradit mezery za no-break space všude v dokumentu... pomocí ~

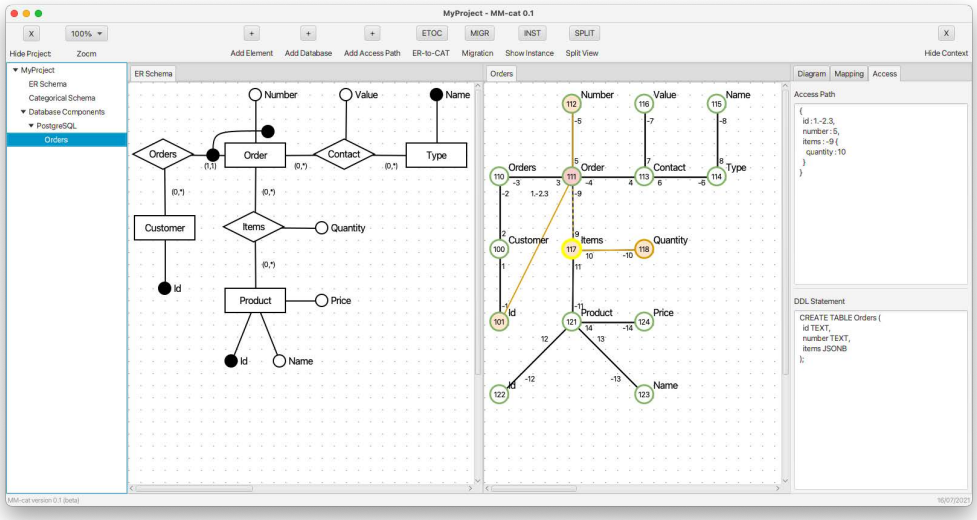
1. Sada nástrojů

1.1 MM-cat

MM-cat framework je navržený tak, aby řešil složitosti spojené s návrhem a správou multi-modelových databází. Jeho hlavním úkolem je modelování multi-modelových schémat a jejich mapování na příslušné DBMS (Database Management System). Slouží i jako základ pro rozšíření o složitější úkoly.

Typickým scénářem použití je vytvoření ER (entity-relationship) schématu uživatelem. Takové schéma je pak automaticky převedeno do jednotné kategorické reprezentace, která umožňuje namapování na kombinaci DBMS. Díky specifikaci schématu je vytvořen skript s příkazy CREATE, které se aplikují na přiřazené DBMS. Uživatel může následně dál upravovat ER diagram, nebo provádět SELECT dotazy.

Na ukázce uživatelského rozhraní (Obrázek 1.1) je výsledek typického scénáře rozebraného v předchozím odstavci. Na levé straně je uživatelem vytvořený ER diagram, na straně pravé pak jeho kategorická reprezentace. Na pravé straně ještě stojí za povšimnutí panel s přístupovými cestami (značené oranžově) a CREATE příkaz.



Obrázek 1.1: Ukázka uživatelského rozhraní nástroje MM-cat.

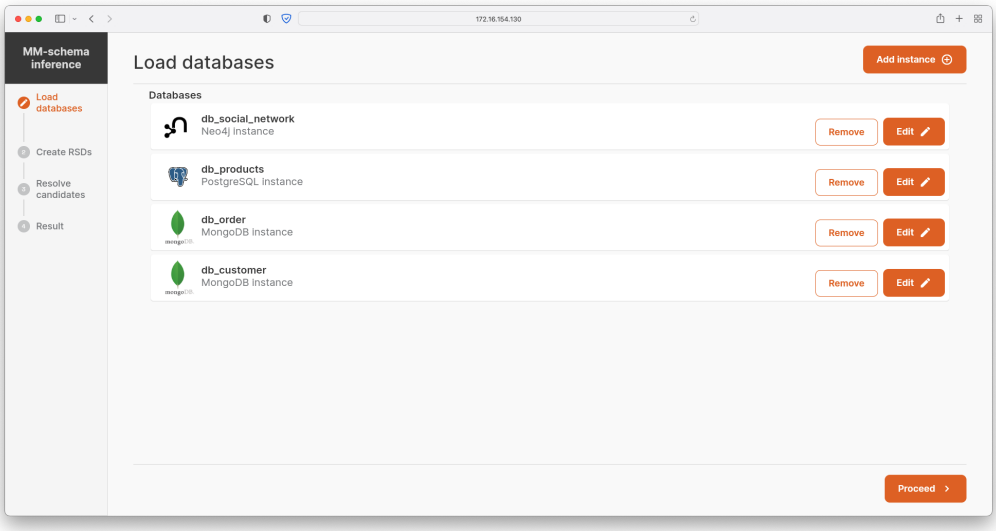
Další možnosti použití, i podrobnější popis nástroje, lze nalézt v článku o MM-cat nástroji [1].

1.2 MM-infer

Ne všechna data mají předem definované schéma. Nástroj MM-infer se snaží schéma zpětně zrekonstruovat z již uložených multi-modelových dat. Je schopen odhalit intra- a inter-modelové reference a překrývání modelů. Zároveň nástroj umí efektivně zpracovávat velké množství dat.

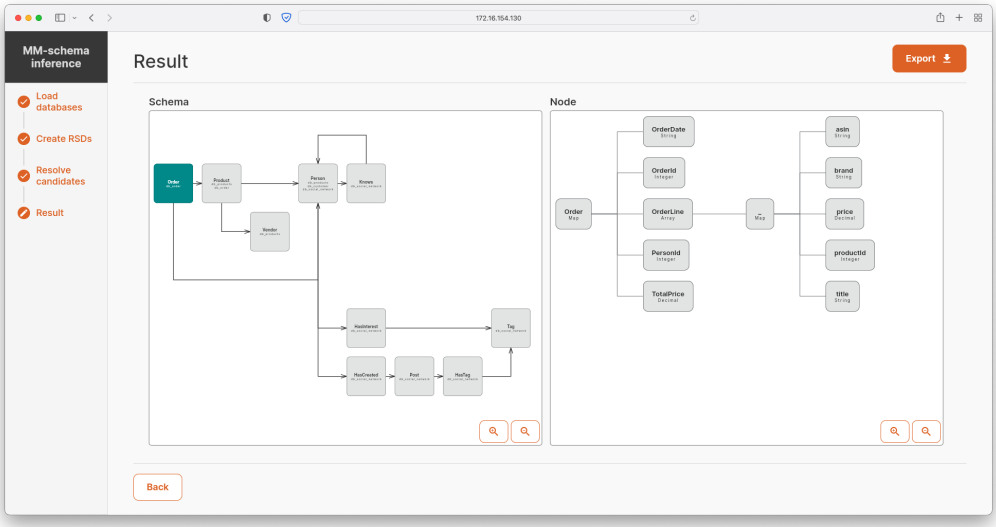
MM-infer podporuje tři druhy DBMS, vybrané tak, aby bylo pokryto co nejvíc funkcí takových systémů. PostreSQL byl vybraný jako zástupce schema-full, relační DBMS. Neo4j reprezentuje schema-less a MongoDB obojí.

Rozhraní aplikace nás provede hned několika kroky, jejichž výsledkem je globální schéma pro vybranou množinu DBMS. Uživatel musí nejdřív takové DBMS vybrat (Obrázek 1.2). V levém panelu je osa ukazující již splněné a následující kroky, pro lepší orientaci v procesu.



Obrázek 1.2: Načtení databází v nástroji MM-infer.

Podle vybraných databází a dalších parametrů je vytvořen RSD (Record Schema Description), sjednocující reprezentace. Po automatickém načtení je pak předána ruka uživateli, který kontroluje navržené kandidáty. Výsledkem je vizualizace globálního schématu (Obrázek 1.3).



Obrázek 1.3: Vizualizace globálního schématu v nástroji MM-infer.

Aplikace příjemně provádí uživatele všemi potřebnými kroky. Nemá zbytečně

příliš různých grafických prvků, které by uživatele zahlcovaly. Stejně jako mm-cat využívá dvou oken pro diagramy vedle sebe.

Další informace o návrhu a možnostech použití aplikace lze nalézt v článku o MM-infer [2].

1.3 MM-evocat

MM-evocat je nástroj pro modelování a správu evoluce v multi-modelových datech. Řeší případy, kdy se struktura dat v čase mění tak, aby například vyhovovala novým uživatelským požadavkům. Takové změny jsou pak propagovány napříč definovanými modely a datovými instancemi.

Umožňuje vytvářet kategorický model, umí ho dekomponovat a umožňuje uživateli, aby si zvolil mapování na podporované DBMS. MM-cat podporuje stejné druhy systémů jako framework MM-infer (PostreSQL, Neo4j, MongoDB).

Kategorický model, mapování a základy konceptuálního modelování popisuje článek MM-evocat [3].

MM-evocat má grafické webové rozhraní.

TODO: rozepsat rozhraní.

1.4 MM-quecat

Více v článku o MM-quecat [4].

TODO: dopsat vše ke quecat.

1.5 Nápady k této sekci, dočasná kapitola

The Importance of the User Interface (v knize Galitz, W. O. The Essential Guide to User Interface Design. Wiley, 2002., tahle kniha je uvedená jako zdroj předmětu https://dcgi.fel.cvut.cz/courses/nur) -> proč vlastně děláme lepší návrh, je tam i user memory a obecně jak pracuje uživatel.

Dost podobné Designing Interfaces (co doporučil Jáchym)

Napsat motivaci pro sjednocení všech nástrojů.

Postupně vznikající nástroje, každý v jiné fázi návrhu, potřeba sjednotit funkce.

2. Cílové skupiny uživatelů

Pro dobré nastavení projektu si definujeme cílové skupiny uživatelů, kteří budou naši aplikaci používat. Pokud už k předchozím projektům, které se snažíme sjednotit, cílové skupiny existují, je dobré se s nimi alespoň seznámit.

Jakmile nastavíme potenciální cílové skupiny uživatelů, zvolíme si konkrétní persony. Pod personou si můžeme představit detailní popis fiktivní osoby, která reprezentuje cílovou skupinu. Persony nám pomohou lépe rozebrat ekonomický status a vlastnosti lidí ve skupinách. Nezapomínáme ani na jejich motivace a zájmy. Nesnažíme se vystihnout všechny ze skupiny, ale vybíráme si hlavně stereotypní vlastnosti a chování. Stejně jako kladné rysy a zájmy chceme vystihnout i co dotyčný nemá rád, případně vůči čemu je úplně odmítavý. O tom co jsou persony a proč se nám vyplatí je definovat, se krátce zmiňuje Jenifer Tidwell v knize Designing Interfaces [5].

Pro práci si definujeme jenom persony, které reprezentují uživatele na které cílíme. Lze ale i definovat vylučující persony. Tedy takové, na které cílit nechceme.

Vytváření cílových uživatelů je důležitým nástrojem, který nás bude provádět v dalších krocích projektu. Díky konkrétním představám nás budou lépe napadat konkrétní řešení a realizace. Snažíme se uzpůsobit návrh personám a vyvolat v nich kladné emoce. Díky tomu získáme konkrétně zaměřený projekt, který nebude tolik odtržený od reálných uživatelů.

2.1 Definice cílových skupin

Protože k předchozím projektům, které se snažíme sjednotit, persony a skupiny neexistují, definovali jsme skupiny nové.

Nejdřív jsme potenciální uživatele rozdělili na začínající a expertní. Jedni očekávají jednoduché koncepty, protože neznají ty složitější. Seznamují se s doménou poprvé a proto bývají mnohem rychleji frustrovaní, či odrazení. Nemají zkušenost ani nadhled, kterým by se učili aplikaci používat rychleji. Tak by se dala v krátkosti vystihnout skupina začínajících. Druzí už mají nějakou zkušenost v oboru, jsou zvyklí na dotazovací jazyky. Nerozumí přímo teorii co stojí za aplikací, ale spíše se opírají o již zmíněnou zkušenost v oboru.

Bylo těžké najít jednotné zástupce skupiny začínajících. I proto jsme se rozhodli skupinu rozdělit na podskupinu mladých lidí, kteří mají více sklony k hravosti a experimentování při objevování nových konceptů a na podskupinu lidí ve věku 40 až 50 let, kteří se chtějí rekvalifikovat pro práci s daty, ale zatím nemají vhodné zkušenosti.

Nakonec jsme si ještě pohrávali s myšlenkou, že by aplikaci využíval stroj. Jednotlivé úkony by byly prováděny pomocí skriptů. Ačkoli se taková skupina nedá zařadit do skupin potenciálních uživatelů, je potřeba i s takovým využitím počítat. Nicméně ji dál nebudeme rozebírat.

2.2 Popis cílových skupin

S charakterizací skupin nám pomůže představa konkrétních person. Do skupiny mladých začínajících můžeme zařadit bakalářského studenta informatického oboru. Pod starším začínajícím uživatelem si představíme padesátiletou pracovnici České pošty, která se chce rekvalifikovat pro práci s daty. Člověk co vystudoval MFF UK a dále pracuje v oboru s daty, třeba učitel Datového inženýrství, bude reprezentovat zástupce expertní skupiny uživatelů.

Za pomocí person rozebereme jednotlivé skupiny potenciálních uživatelů. Každou skupinu krátce charakterizujeme, následně se snažíme vystihnout její chování. Nakonec se zamyslíme nad úkoly, které budou uživatelé v aplikaci dělat.

2.2.1 Skupina mladých lidí

Je skupinou začínajících laiků, kteří se s aplikací seznamují úplně poprvé. Nemají příliš zkušeností v oboru, neznají dotazovací jazyk. Budou pracovat hlavně s grafickým prostředím, které by jim mělo práci usnadnit a motivovat je.

Chovají se spontánně, rozhodují se impulzivně a experimentují. Z toho co dělají vyzařuje hravost. Tímto přístupem velmi rychle prozkoumají různá zákoutí aplikace a dobře otestují její funkčnost. Učí se velmi rychle, ale stejně tak je dokáže rychle odradit i maličkost. Lépe si představují věci vizuálně a ocení odezvu nebo odměnu za úkol, který provedou. Odměna pro ně může být rychlá zpětná vazba, třeba odezva na úkony v grafickém prostředí. Pro pohyb v prostředí využívají hlavně počítačovou myš. Ocení barevnost, různorodost a zajímavé podání.

V aplikaci budou využívat možnosti dotazování nad daty, jednoduché modelování schematických kategorií, bez přidávání mapování a dalších složitějších operací. Nebude se jich týkat převod konceptuálního znázornění dat do multimodelové databáze a funkce nástroje MM-infer.

2.2.2 Skupina začínajících ve věku 40-50 let

Je druhá skupina začínajících uživatelů. Je to skupina, co se typicky chce rekvalifikovat do práce s daty, ale mají zatím jen nevhodné zkušenosti. Jsou seznámeni se základní prací na počítači, umí používat webový prohlížeč, tabulkový a textový procesor. Aplikace by pro ně měla být ideální na přeučení. Nemusí se učit komplexní technologie a dál budou dělat úkony přes grafické rozhraní.

Narozdíl od mladých začínajících mají metodický přístup chování na webu, přistupují k věci konzervativně. Upřednostňují umírněné podání grafiky. Jsou pro ně důležité popsané konkrétní kroky a argumenty. Nepracují spontánně, dělají zadanou práci. Je potřeba servírovat jim množství práce postupně v dávkách, aby nepřišlo informační zahlcení a odrazení od aplikace. Stejně jako mladí začínající jsou klikací typ, k práci využívají počítačovou myš.

Úkony prováděné v aplikaci jsou podobné skupině mladých začínajících lidí.

2.2.3 Skupina expertních uživatelů

Poslední skupina, která už zná koncepty ve světě databází. Zvyklá na textové dotazovací jazyky, databázové modely. Jsou schopní psát si vlastní skripty a jinak automatizovat a zefektivnit práci.

Nechovají se spontánně a k aplikaci přistupují hlavně metodicky. Dokážou se lépe orientovat v aplikaci, díky zkušenosti z oboru. Jsou spíše tolerantní, nemají přehnané nároky na aplikaci. Vědí, že je složité přecházet na jiný systém, narozdíl od začínajících uživatelů, které odradí i maličkost. Více si potrpí na funkce, které jim urychlí práci. Myšleno možnost používat skripty a klávesové zkratky. Ocení střízlivé podání a strohá data. Pracuje jak s grafickým rozhraním, tak i textovým.

Expertní uživatel využije více funkcí aplikace. Není omezen jen na tvorbu schematické kategorie. Může přidávat mapování a joby, převádět konceptuální znázornění dat do multimodelové databáze. Stejně tak bude využívat funkce nástroje MM-infer.

3. HTA (Hierarchical Task Analysis)

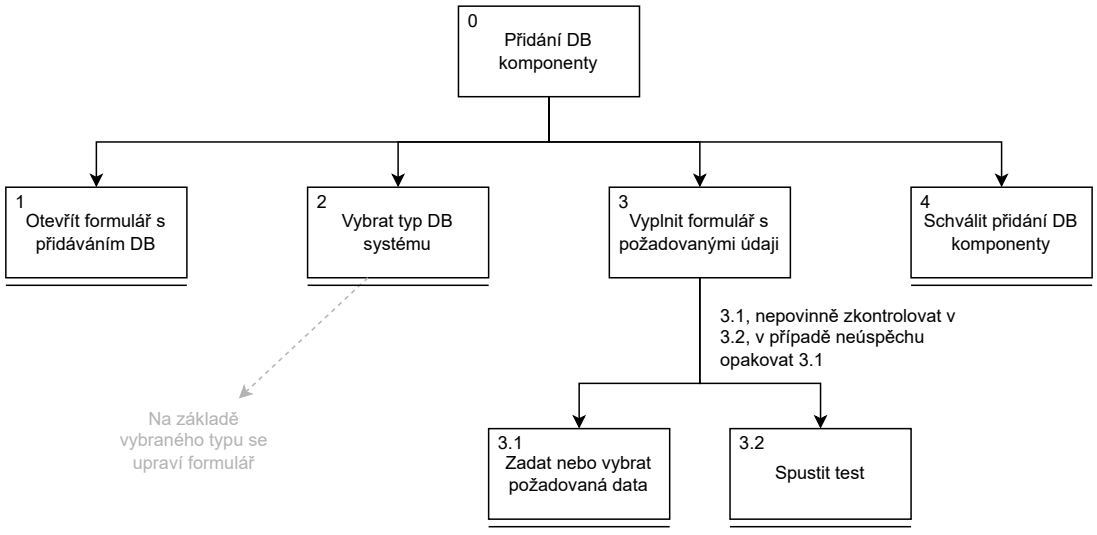
Na definování skupin uživatelů navážeme analýzou úkolů (Task Analysis) prováděných uživateli v aplikaci. Hlavním cílem analýzy je dokumentovat a strukturovat kroky nezbytné k dokončení úkolů. Díky definování různých úkolů, rozdělíme komplexní úlohy na menší, lépe představitelné části. Vše zaznamenáme do hierarchické struktury, odtud se vzalo slovo Hierarchical. Soustředíme se jen na pozitivní průchod.

Hierarchickou analýzu úkolů můžeme rozdělit na několik částí. Jako první určíme úkoly, které chtějí uživatelé v aplikaci dokončit. Každý hlavní úkol rozdělíme na podúkoly a akce, které definují vztah mezi podúkoly. Navíc každému hlavnímu úkolu přiřadíme výchozí situaci. Následně z jednotlivých komponent vytvoříme hierarchický diagram, v našem případě se stromovou strukturou. Jako poslední analyzujeme posloupnost akcí a snažíme se strukturu pochopit a vylepšit.

3.1 Vytváření diagramů

V charakterizaci cílových skupin (2.2) už jsme narazili na úkoly, které by uživatelé mohli dělat. Vybereme si z nich tři, které budeme analyzovat. Začneme úkolem, kdy uživatel přidá databázovou komponentu. Další hlavní úkol je vytvoření jednoduchého schématu a poslední jeho dekompozice. Vždy začínáme v situaci systému otevřeného na hlavní stránce.

Podívejme se na první jednoduchý diagram (Obrázek 3.1). Uživatel přidá nový databázový systém k již existujícím. Začíná ve spuštěném systému.



Obrázek 3.1: Diagram pro přidání databázové komponenty.

Kořen oznamuje jaký úkol bude uživatel provádět. V první hladině máme kroky 1 až 4, které nám v základních rysech ukazují, jak bude uživatel postupovat při plnění úkolu. Krok 3 dál rozdělíme na 3.1 Zadání dat a 3.2 Otestování. Na

hraně pak definujeme postup uživatele při plnění kroků 3.1 a 3.2. Když není hrana popsaná, předpokládáme průběh kroků podle čísel v hladině vzestupně. Konkrétní pořadí ale není podmíněné. Číslování 3 a 3.x vyjadřuje vztah mezi rodičem a podúkoly. Podtržené stavy v HTA se dál nedělí.

Takto vytvořený diagram nám neříká nic o interakci uživatele s konkrétním systémem. Díky tomu ale můžeme kroky rychle měnit a upravovat. Lépe zhodnotíme, jestli uživatel nedělá příliš mnoho kroků pro dokončení úkolu, nebo naopak. Často se může stát, že jeden krok je ještě potřeba rozdělit.

Všechny tři analýzy úkolů, včetně přidání DB komponenty jsou k nahlédnutí v příloze A.1.

4. Storyboardy

Na vytvoření uživatelských skupin a HTA diagramů navážeme tvořením Storyboardů. Storyboard je náčrt sekvence kroků, které uživatel v aplikaci provádí. Každý důležitý bod je zaznamenán do buňky. Ty pak skládáme lineárně za sebe, až nám vznikne příběh připomínající komiks. Takové rozdělení umožňuje soustředit se na každý krok zvlášť. Zároveň se jedná o další techniku, kde můžeme udělat rychlý náčrt a podle potřeby upravovat. Jeden storyboard nezabere víc než jednu stránku.

Cílem storyboardu je zjednodušeně zobrazit problém co se uživatel snaží v aplikaci řešit a zamyslet se nad jeho přístupem k danému problému. Jde nám hlavně o vizuální zobrazení konkrétních kroků, které jsme popisovali v HTA diagramech. Zároveň nám může být návrh nápomocný, abychom si vizualizovali, jak budou uživatelé používat navrhovanou aplikaci a jak bude aplikace v náznacích vypadat. Získané představy a vizualizace použijeme v detailnějším návrhu rozhraní a budeme se k nim snažit přiblížit.

V našem případě vezmeme reprezentanta každé skupiny uživatelů a představíme si jeho chování pro jednu vybranou situaci z námi vytvořeného HTA diagramu.

Všechny storyboardy jsou v příloze A.2.

4.1 Ukázka vytváření storyboardu

Jako první začneme se zkušeným uživatelem. V aplikaci bude přidávat dvě různé databázové komponenty (PostgreSQL, MongoDB). Strukturu kroků zachováváme podobnou prvnímu hta (3.1). Chceme zachytit uživatele, který se umí pohybovat v doméně, jenom nepoužíval náš nástroj.

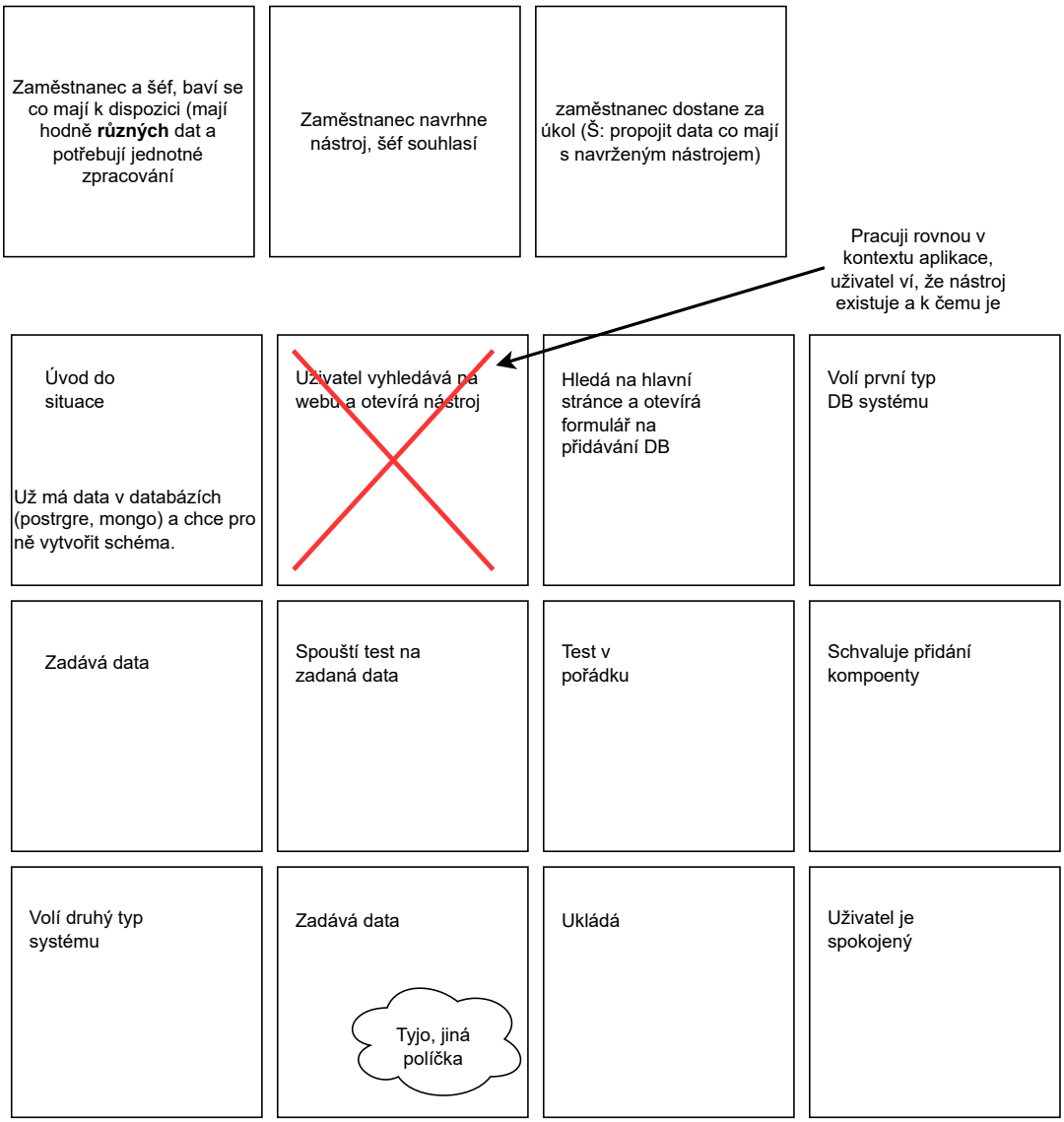
Na Obrázku 4.1 můžeme vidět rychlý návrh obsahu storyboardu. Kroky v buňkách nejdřív místo kreslení popisujeme. Získáme tím přehled všech hlavních kroků, které bude uživatel dělat a ujasníme si, kolik asi bude potřeba buněk.

V rámci návrhu jsme vytvořili několik iterací. Některé snímky jsme z návrhu vyřadili, protože nebyli potřebné (např. červeně škrtlý snímek). První tři kroky jsme k původnímu návrhu přidávali jako širší úvod do situace.

Dáváme si pozor, abychom se v této fázi návrhu nenechali strhnout detaily, které zatím nejsou potřeba. O prvotních fázích návrhu se zmiňuje i kniha Refactoring UI [6]. Navrhuje jednoduchý způsob, jak se nenechat pohltit detaily jako jsou typy písma, stíny, ikony atd. Podle autorů stačí vzít papír, tlustou fixu a tím si v podstatě znemožnit kreslení malých detailů. My jsme navíc omezení velikostí buněk, do kterých kreslíme.

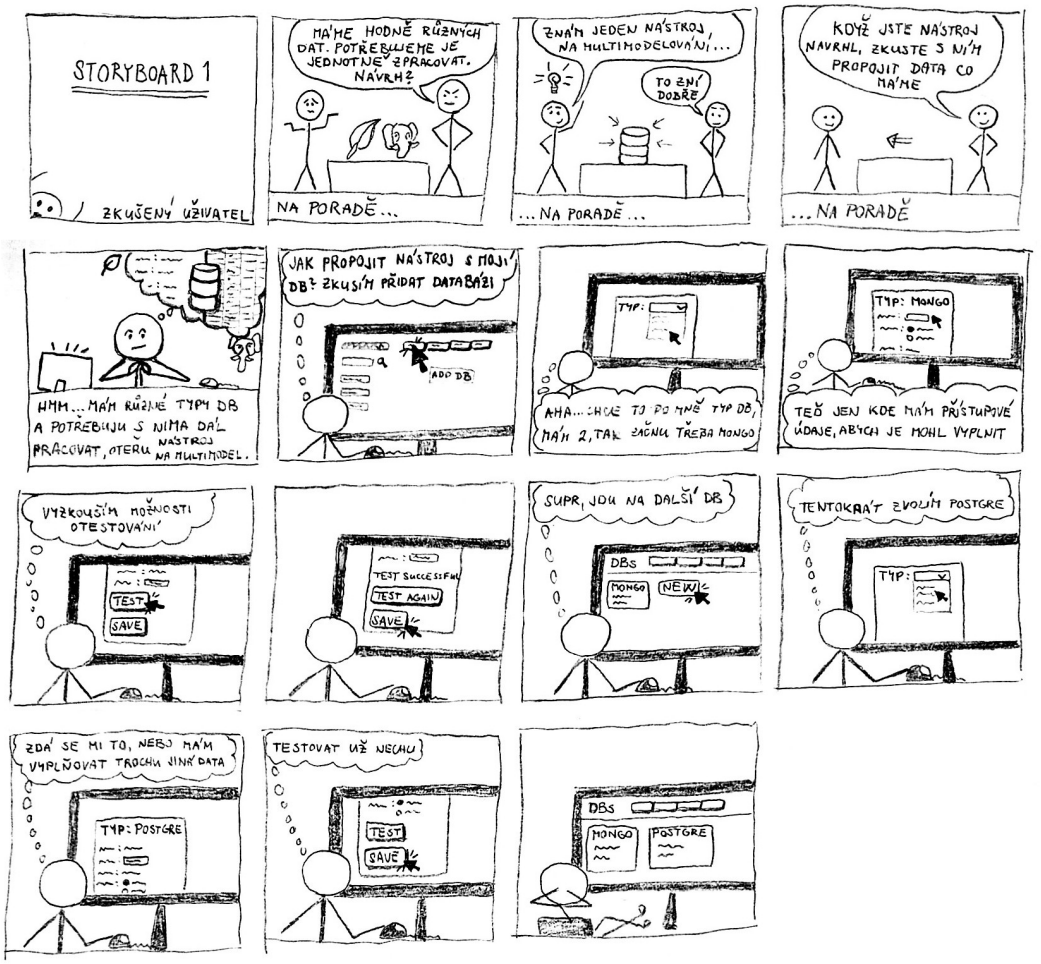
Při vyjadřování situace pamatujeme na to, že aplikaci momentálně ovládá zkušený uživatel. V aplikaci se zorientuje rychle a s úkolem nemá problém. Obrázek 4.2 je kompletní storyboard znázorněný pomocí kreseb místo textu. V návrhu je vidět, že uživatel nemá s používáním aplikace problém. Sám navrhl použití nástroje. Na několika snímcích je vidět i prvotní rozhraní aplikace, zatím opravdu jen v náznacích.

Oproti Obrázku 4.1 si díky vizualizaci v Obrázku 4.2 lépe představíme co bude uživatel dělat a jak bude v náznacích vypadat aplikace, ve které pracuje. Stejně



Obrázek 4.1: Rychlý návrh storyboardu pomocí textu v buňkách.

tak se potom konkrétní představy lépe komunikují s lidmi se kterými storyboard konzultujeme.



Obrázek 4.2: Výsledný storyboard, který znázorňuje uživatele přidávajícího databázové komponenty

5. Low-fidelity prototyp

Po storyboardech, které nám ukázali možné sekvence akcí, vytvoříme prototyp aplikace, se kterým budou interagovat uživatelé. V rámci ročníkového projektu vytvoříme jen low-fidelity prototyp. Tedy se budeme zabývat vytvořením rané verze aplikace, která nebude věrnou kopií aplikace výsledné. V této fázi, chceme hlavně rychle vytvořit prototyp, na kterém můžeme vyhodnocovat nápady a zároveň ho můžeme otestovat na uživatelích spadajících do jednotlivých uživatelských skupin, které jsme si definovali.

5.1 Paper prototyping (mockups)

Pro navrhování a testování uživatelského rozhraní ve verzi low-fidelity jsme si vybrali jednu z využívaných metod, prototypování na papír (paper prototyping). Někdy také nazývaná jako tvorba paper mockupů. Celý proces od vytvoření prototypu až po uživatelské testování popisuje detailně kniha Paper prototyping od Carolyn Snyder [7].

Začneme vytvořením papírových komponent aplikace (oken, menu, stránek, dialogových boxů, dat, pop-up zpráv). Po vytvoření prototypu provedeme usability otestování. V takovém testování provádí uživatel, v jednom sezení, realistické úkoly na papírovém prototypu. Nejedná se o studii, kdy jsou prováděny série testů použitelnosti během několika dní. Uživatelé jsou vybraní podle uživatelských skupin, které jsme si definovali na začátku.

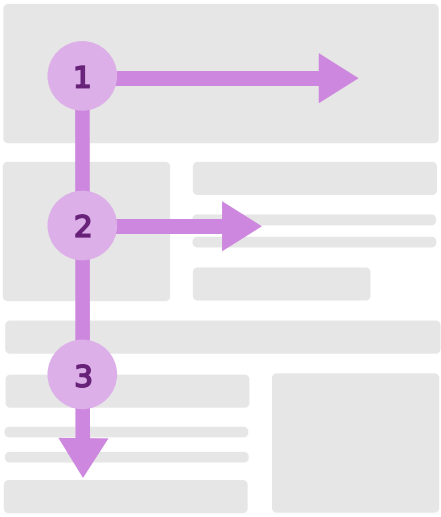
Prototyp bude ovládaný osobou, která reprezentuje konání počítače, ale nevysvětluje, jak má rozhraní fungovat. Další zkušený člověk funguje jako zapisující pozorovatel. Pozoruje chování uživatele, zapisuje co uživatel dělá a co říká. Tímto způsobem provedeme velmi rychle iterativní testování na několika uživatelích. V průběhu můžeme aplikaci vylepšovat a všímat si opakujících se vzorů. Každé sezení s uživatelem končí vyplněním dotazníku spokojenosti s aplikací.

5.1.1 Vytváření prototypu

Papírový prototyp vytvoříme v nezávislosti na existujících aplikacích popsaných v 1. kapitole. V tomto momentě je dobré si ujasnit, na jakém zařízení budeme aplikaci používat. Implicitně jsme od začátku předpokládali, že půjde o počítačové rozhraní. Mobilní aplikace se k němu nepřidá, alespoň ne ve verzi s možností úprav. Stejně tak aplikace pro tablety. Tahle zařízení jsou užší a menší. Mají tak méně prostoru pro obsah, navigaci a interakci. Použití modelovacích a složitějších funkčností by bylo často složité až nemožné.

Pro obecné rozložení komponent jsme vzali v úvahu F-Pattern. Je pojmenovaný podle tvaru, který tvoří pohyby očí při skenování a čtení aplikace, připomínající písmeno F (Obr. 5.1). Nehodí se pro telefony, to nás ale neomezuje. Dnes spíš návrh aplikací tíhne k rozmanitosti, proto na něm aplikaci nestavíme, jenom ho bereme v potaz. Hlavním jazykem aplikace bude angličtina. Uživatelé jsou zvyklí číst zleva doprava, proto je pattern orientovaný stejným směrem.

Při návrhu se zaměřujeme na dostupnost různých částí aplikace. U těch, které jsou často používané se snažíme minimalizovat počet kliků, díky nimž se k hledané



Obrázek 5.1: F-Pattern layout.

informaci dostaneme. Vezměme si jako příklad otevření formuláře pro přidání databázové instance. Na jeho otevření nám poslouží přesun jedním kliknutím do okna se správou databázových instancí a na něm tlačítko pro přidání, které je na stránce nejvýraznější.

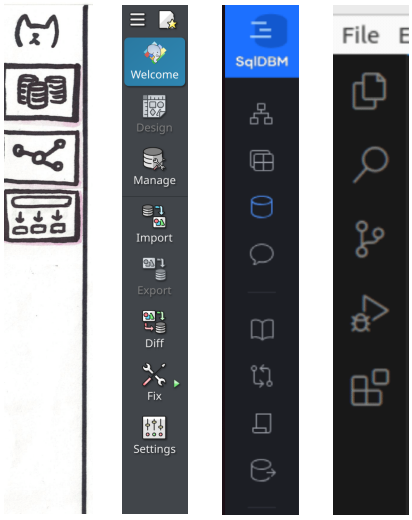
Neoficiální pravidlo tří kliků (3-click rule) tvrdí, že přístup k jakékoli informaci v aplikaci by neměl být delší než tři kliknutí. Vzniklo z přesvědčení, že budou uživatelé frustrovaní a opustí aplikaci, když nenaleznou hledanou informaci do tří kliků. Pravidlo je ale příliš zjednodušené a nezohledňuje složité interakce. Dosažení nízkého počtu kliků s sebou nese riziko, že budeme ignorovat vše ostatní. Může vést i k frustraci uživatelů způsobené obšírnými navigacemi. Proto se pravidlem přímo neřídíme, ale snažíme se minimalizovat počet kliků, když to jde.

Chceme, aby se uživatel ocitl v pro něj známém (intuitivním) prostředí. Jako inspirace nám posloužily existující webové nástroje a desktopové aplikace. Úvodní obrazovka je tvořená z levé lišty (Obr. 5.2), která obsahuje tlačítka pro otevření hlavních částí aplikace (spojení s databázovými instancemi, modelování, vytváření mapování), a volného prostoru. Podobnou lištu lze nalézt například v nástrojích pgModeler[1], sqlDMB[2] a VS Code[3] (Obr. 5.2). Počítáme s tím, že až budou přibývat funkčnosti dalších nástrojů, budou přibývat tlačítka v levé liště.

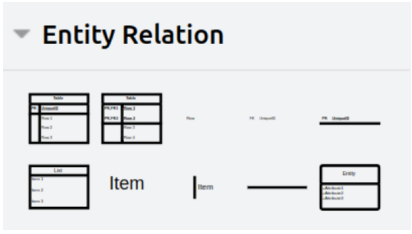
V části aplikace pro správu databázových instancí vytvoříme lištu s již existujícími spojeními a jednoduchý formulář, který bude umožňovat přidání nové instance.

U vytváření části aplikace s modelováním schématu nám jako inspirace posloužil nástroj Draw.io[4]. Konkrétně hlavně u ER modelování lišta vlevo (Obr. 5.3) a jednotlivé komponenty, které se přidávají na plochu. Do našeho papírového návrhu jsme totiž přidávali, místo vytváření cathegory schema, vytváření ER modelu. Proto, abychom uživateli představili něco jednoduššího s čím se už pravděpodobně setkal.

[1] `https://pgmodeler.io/`
[2] `https://sqldbm.com/Home/`
[3] `https://code.visualstudio.com/`
[4] `https://app.diagrams.net/`



Obrázek 5.2: Porovnání papírové lišty (vlevo) s lištami existujících aplikací.



Obrázek 5.3: ER lišta z nástroje Draw.io, sloužící jako inspirace.

Snažili jsme se ponechat co nejvíc volné bílé plochy pro pohodlné modelování a dost místa na umisťování komponent. Dodatečné lišty, sloužící k úpravě vlastnostní jednotlivých komponent, jsme vytvořili plovoucí, tak aby je uživatel mohl přetáhnout, pokud by někde překážely a aby zabíraly co nejméně místa.

Pro přidávání mapování do schématu jsme se drželi podobné struktury jako u vytváření schématu. Všechny papírové komponenty prototypu jsou naskenované v příloze A.3.

5.2 Uživatelské testování

Proč vůbec chceme low-fidelity návrh testovat na uživatelích? Kromě toho, že papírový návrh se dá rychle měnit, se můžeme vyhnout hnidopišské zpětné vazbě. O ní se zmiňují i v knize Paper Prototyping na straně 58 [7]. Komponenty a barvy nejsou jasně dané, uživatel se spíš zaměří na koncepty a funkčnost. Je totiž očividné, že jsme ještě nespecifikovali vzhled. Schumann a další se v jednom ze svých článků [8] zabývají tím, že nedokončený návrh povzbuzuje ke kreativitě a k tomu, aby uživatel nebyl pasivní a sám přemýšlel nad koncepty.

Připravíme si úkoly, které budou uživatelé dělat. Budou to ty nejběžnější, které jsme rozebírali v předchozích kapitolách. Jmenovitě první úkol, kdy uživatel přidá databázovou komponentu. Další hlavní úkol je vytvoření jednoduchého schématu a poslední jeho dekompozice. Po připravení úkolů najdeme uživatele spadající do vytvořených uživatelských skupin. Dáváme si pozor, aby neznali aplikaci nebo

naše názory na ni.

V kapitole Some Techniques for Observing Users knihy The Art of Human-computer Interface Design [9] autorka popisuje techniky pro pozorování uživatelů, které se nám při testování budou hodit. Chceme zjistit, kde mají uživatelé problém aplikaci používat a co jim naopak vyhovuje.

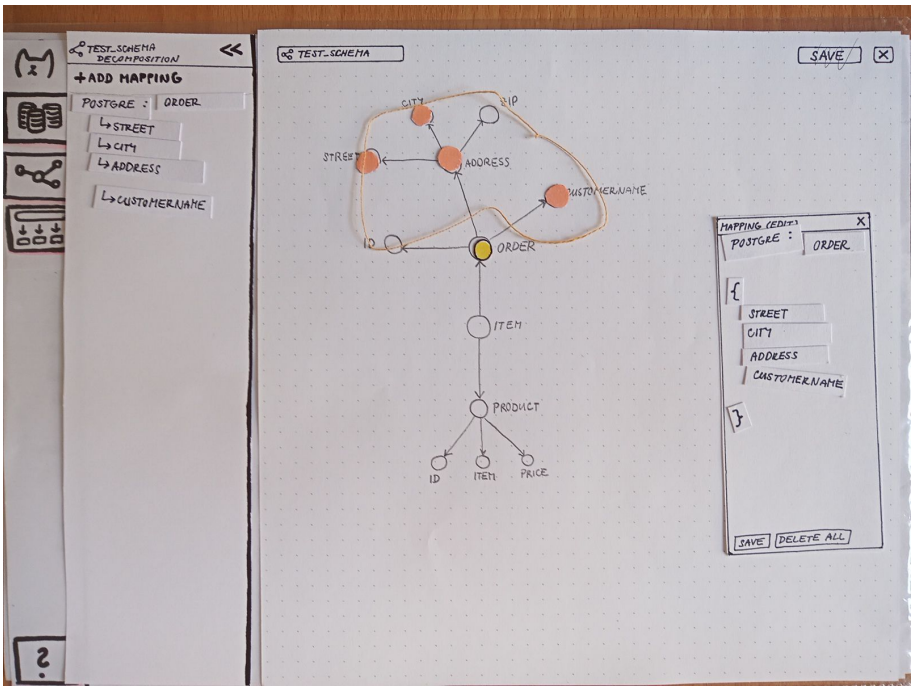
Pro samotné testování vybereme místo, které je tiché a bez zbytečných vnějších vyrušování. Popíšeme uživateli o co se jedná a vyvětlíme, že jsou zapojeni do raných fází návrhu. Zdůrazníme, že testujeme aplikaci, ne uživatele. Poprosíme uživatele, aby přemýšleli nahlas a říkali to co jim přijde na mysl v průběhu plnění úkolů. Díky tomu prozkoumáme jejich očekávání od produktu, taky jejich úmysly a jejich strategie řešení problémů. Nakonec ještě upozorníme, že nebudeme uživateli pomáhat při plnění úkolů. Je to nejlepší způsob jak zjistit jak uživatelé reálně interagují s aplikací.

Po otestování zodpovíme zbývající otázky od uživatele. Případně diskutujeme nějaké zajímavé chování, které uživatel při testování měl. Závěrem pak bude vyplnění dotazníku uživatelem. Je důležité, aby v dotazníku byly jenom otevřené otázky, které uživatele nijak nenavádí. Ptáme se hlavně na pocity z aplikace, jestli něco neodvádělo pozornost uživatele a jestli dává smysl navigace po aplikaci.

5.3 Závěr z uživatelského testování

Papírový prototyp otestovalo sedm uživatelů pokrývajících všechny uživatelské skupiny. Zpětnou vazbu jsme získali ze zápisků od pozorovatele a z dotazníku. Všechny zápisky, dotazník i odpovědi na dotazník lze nalézt v příloze A.4

Pro představu průběhu uživatelského testování je na Obrázku 5.4 zachycena situace přidávání mapování do existujícího schématu.



Obrázek 5.4: Ukázka rozložení aplikace při uživatelském testování.

V průběhu testování na uživatelích jsme do aplikace přidali vyskakující okna s potvrzením o úspěšně provedené akci (například uložení souboru).

V dotazníku uživatelům připadala aplikace konzistentní a nic neodvádělo jejich pozornost od komponent na které v dalších krocích klikali. Díky konzistenci rozložení v aplikaci se uživatelé po splnění prvního úkolu pohybovali rychleji.

Uživatelé používali i otazník s nápovědou v hlavní liště. Původním předpokladem bylo, že uživatelé budou skeptiční a raději budou zkoušet klikat na různé prvky. Několik z nich si taky všimlo, že na úvodní obrazovku, která je tvořená jen z levé lišty a jinak je prázdná, by bylo dobré přidat nějaký úvodní text s nabídkou úkonů, které může uživatel začít dělat, pro rychlejší orientaci, když aplikaci otevře. Takovou nabídku jsme do aplikace nepřidávali, ale nakonec nás to napadlo a uživatelé nám domněnku potvrdili.

S jedním uživatelem jsme diskutovali rozložení hlavních funkcí. Poukázal na to, že by nebylo špatné schovat vytváření databázových spojení místo první funkce někam, aby k tomu měl přístup hlavně administrátor a expertní uživatel a obecně, aby se méně zkušeným uživatelům tahle možnost nenabízela. Určitě plánujeme zapojit tento poznatek do dalšího návrhu a v hlavní liště schovat nebo alespoň prohodit tlačítka urená hlavně pro expertní uživatele.

Obecně měli uživatelé problém s posledním úkolem, kde vytvářeli mapování. Protože je to ale úkol hlavně pro expertní uživatele, byly takové problémy očekávané. Naopak formulář pro přidávání DB komponenty pochopili všichni velmi rychle.

Závěr

Seznam použité literatury

[1] Pavel Koupil, Martin Svoboda, and Irena Holubová. MM-cat: A Tool for Modeling and Transformation of Multi-Model Data using Category Theory. In *Proc. of MODELS '21*, pages 635–639. IEEE, 2021.

[2] Pavel Koupil, Sebastián Hricko, and Irena Holubová. MM-infer: A Tool for Inference of Multi-Model Schemas. In *Proc. of EDBT '22*, pages 2:566–2:569. OpenProceedings.org, 2022.

[3] Pavel Koupil, Jáchym Bártík, and Irena Holubová. MM-evocat: A Tool for Modelling and Evolution Management of Multi-Model Data. In *Proc. of CIKM '22*, pages 4892–4896. ACM, 2022.

[4] Pavel Koupil, Daniel Crha, and Irena Holubová. MM-quecat: A Tool for Unified Querying of Multi-Model Data. In *Proc. of EDBT '23*, pages 831–834. OpenProceedings.org, 2023.

[5] J. Tidwell and an O'Reilly Media Company Safari. *Designing Interfaces, 2nd Edition*. O'Reilly Media, Incorporated, 2010.

[6] A. Wathan and S. Schoger. *Refactoring UI*. Adam Wathan & Steve Schoger, 2019.

[7] Carolyn Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. The Morgan Kaufmann series in interactive technologies. Elsevier Science, 2003.

[8] Jutta Schumann, Thomas Strothotte, Andreas Raab, and Stefan Laser. Assessing the Effect of Non-photorealistic Rendered Images in CAD. In *Proceedings of CHI'96 Conference on Human Factors in Computing Systems (Vancouver, Canada, apr 1996)*, pages 35–42. ACM SIGCHI, 1996.

[9] Brenda Laurel and S.J. Mountford. *The Art of Human-computer Interface Design*. Computer systems. Addison-Wesley, 1990.

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

A. Přílohy

A.1 Hierarchical Task Analysis

V souboru `hta.pdf` jsou všechny provedené analýzy úkolů.

A.2 Storyboardy

Storyboardy k vybraným úkolům jsou k nahlédnutí v `storyboardy.pdf`

A.3 Paper Mockupy

Rozložené komponenty papírového mockupu jsou naskenované v souboru `paper-mockup-laid-out.pdf`.

A.4 Uživatelské testování

Google Forms formulář je dostupný jako `formular-testovani-mockupu.pdf`. Odpovědi na formulář jsou zaznamenány v `formular-odpovedi.pdf` a výsledky od zapisovatele v `souhrn-od-zapisovatele.txt`.