Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Možnosti zvýšení uživatelského komfortu adaptací uživatelského rozhraní aplikací

Diplomová práce

Autor: Karel Kobliha
Studijní obor: Aplikovaná informatika

Vedoucí práce: prof. Ing. Ondřej Krejcar, Ph.D.

Hradec Králové listopad 2020

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 14.11.2020 Karel Kobliha

Poděkování:

Rád bych poděkoval vedoucímu mé diplomové práce, panu prof. Ing. Ondřeji Krejcarovi, Ph.D. za metodické vedení práce a trpělivost.

Anotace

Tato diplomová práce se zabývá problematikou uživatelského rozhraní. Zkoumá různé druhy a popisuje omezení, která se vyskytují u konkrétních typů. Dále se zabývá metodikami návrhu uživatelského rozhraní, jejich zaměřením a cíli při zvolení jejich postupů. Současně sleduje samotnou tvorbu uživatelského rozhraní, výhody a nevýhody daných typů.

Práce se také zaměřuje na adaptaci uživatelského rozhraní. Zkoumá návrhy adaptivních systémů a frameworků, které se používají pro potřeby uživatelů. Následně obsahuje návrh na změnu uživatelského rozhraní existující firemní aplikace, které se lépe přizpůsobí potřebám uživatelů. Popisuje implementaci návrhu a zpětnou vazbu v průběhu vývoje. Dále popisuje výsledky z implementace a navrhuje doporučení pro další úpravy systému.

Annotation

Title: The possibilities of increasing user comfort by adaptation of applications UI

This thesis deals with the issue of the user interface. It examines differences of their types and describes the limitations of that concrete types. It also deals with user interface design methodology, describing the types, their focus and goals when choosing their procedures. Concurrently, the thesis deals with the user inteface creation, describing advantages and disadvantages of the given types.

This thesis also focuses on the adaptation of the user interface. It examines the design of adaptive systems and frameworks that are used for the needs of users. Subsequently, it describes a proposal to change the existing business application user interface for its better adaptation for needs of users. Thesis contains design and its following implementation and user feedback during development. It also describes the result of the implementation and provides recommendations for further modifications of the system.

Obsah

Seznam obrázků

Seznam tabulek

1 Teoretický úvod do problematiky

Při používání výpočetní techniky se každý člověk stává uživatelem. Aplikací, se kterými přijdeme do styku, je velké množství a svým provedením jsou rozmanité. Jako soukromé osoby máme možnost vybrat si takovou aplikaci, která nám nejlépe vyhovuje, popřípadě se kterou se nám nejlépe pracuje. Rozhodujeme se podle účelu, ke kterému má být aplikace použita, ale také podle uživatelského rozhraní, které aplikace poskytuje. Díky velkému množství aplikací zdarma máme k dispozici téměř neomezený výběr, zvláště v mobilním světě. Pokud nás rozhraní neosloví nebo nebude splňovat naše očekávání, raději se pokusíme najít nějakou jinou aplikaci, s níž budeme více spokojeni. Aplikace s přehlednou pracovní plochou, vhodně umístěnými ovládacími prvky a nabízející možnost přizpůsobit si uživatelské rozhraní podle konkrétních individuálních potřeb uživatele zpravidla vítězí.

Ve firemním prostředí není výběr aplikací velký a nový zaměstnanec se jako uživatel často musí přizpůsobit tomu, co se ve firmě již používá. Firemní aplikace nebývají jednoúčelové, ale mají za cíl poskytovat velké množství informací pro různé typy zaměstnanců. Aby poskytovaly co nejvíce informací pro práci svým uživatelům, mají často stejné uživatelské rozhraní pro všechny, rozdělené na seznamy údajů v tabulkách a na lišty s ovládacími tlačítky, které nabízí co nejvíce funkcí svým uživatelům. Nový uživatel se proto musí nějakou dobu přizpůsobovat uživatelskému rozhraní a naučit se, jaké části uživatelského rozhraní může používat a které nejsou určeny pro něj. Málokterá aplikace nabízí možnost přizpůsobení uživatelského rozhraní uživateli tak, aby se mu lépe pracovalo a měl k dispozici pouze ty ovládací prvky a údaje, které ke své práci skutečně potřebuje.

Diplomová práce se zabývá možnostmi adaptace firemní aplikace uživateli. Druhá kapitola práce popisuje situaci ve firemním prostředí a definuje cíle práce. Třetí kapitola se věnuje hledání odpovědi na otázku, co je to uživatelské rozhraní, jak může vypadat a jaké jsou způsoby jeho návrhu. Také hledá možnosti adaptace uživatelského rozhraní. Čtvrtá kapitola se zabývá návrhem změny firemní aplikace a pátá kapitola popisuje implementaci návrhu. V předposlední kapitole jsou zhodnoceny výsledky z testování aplikace a změny aplikace v průběhu testů. Závěrečná kapitola vyhodnocuje přínos adaptace uživatelského rozhraní.

2 Cíle práce

Cílem práce je prozkoumat existující řešení a dostupnou literaturu, věnující se adaptaci uživatelského rozhraní, na základě zjištěných informací navrhnout a implementovat vlastní řešení pro zvýšení uživatelského komfortu adaptací uživatelského rozhraní a zhodnotit přínosy tohoto řešení diskuzí zjištěné zpětné vazby od uživatelů aplikace.

Podrobněji lze cíle práce rozdělit na následující body:

- Seznámit se s druhy uživatelského rozhraní,
- seznámit se s metodami návrhu uživatelského rozhraní,
- prozkoumat druhy adaptací uživatelského rozhraní,
- navrhnout vlastní řešení a technologie pro jeho tvorbu,
- implementovat navržené řešení s využitím vybraných technologií,
- realizovat testovaní implementovaného řešení a diskutovat získané výsledky testování.

3 Rešerše existujících řešení a literatury

Počítačové programy jsou lidmi používány za pomocí uživatelských rozhraní (*angl. user interface, UI*). Ta umožňují uživatelům nejen komunikovat se zařízením, ale také získávat, měnit a ukládat jejich data. Rozhraní tak fungují jako prostředník a umožňují uživatelům využívat prostředky zařízení přívětivým způsobem, zatímco náročnost provádění jednotlivých instrukcí, potřebných ke splnění požadovaných úkonů, je uživateli skryta. S vývojem výpočetní techniky postupně vznikaly různé typy uživatelského rozhraní.

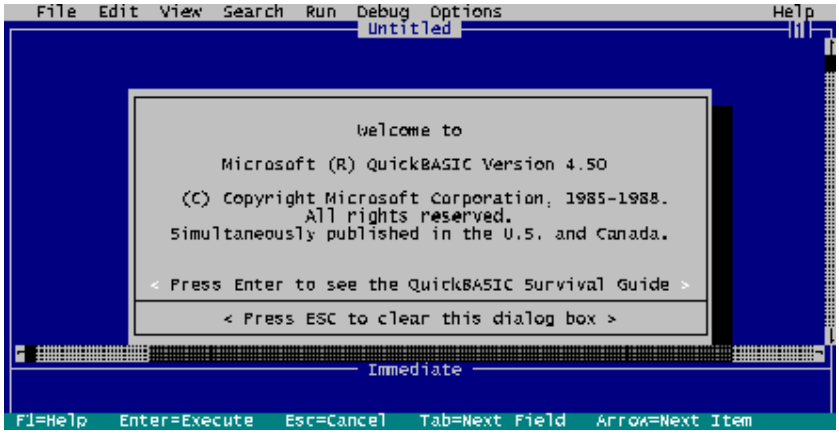
3.1 Druhy uživatelských rozhraní

Jedním z prvních typů uživatelského rozhraní, který umožnil interakci počítače a uživatele, je rozhraní příkazového řádku (*angl. command-line interface, CLI*). Jeho historie sahá až do šedesátých let minulého století, kdy se objevil operační systém Multics [1] a jím inspirovaný systém UNIX [2].

Prostředí se skládá z příkazového řádku, do kterého jsou vkládány příkazy uživatele, a výpisu řádků, kterými operační systém odpovídá. Řádek, který slouží uživateli pro komunikaci se zařízením, je zobrazen systémem jako výzva (*angl. prompt*) a blikající řádek, čímž systém dává najevo uživateli, že je připraven zpracovávat další příkazy. Uživatel při zadávání příkazů používá především klávesnici pro zadávání jednotlivých příkazů, případně i jejich parametrů. Za pomoci šipek se může pohybovat mezi jednotlivými znaky právě zapisovaných údajů nebo zobrazovat nedávno zapisované příkazy. Po stisknutí klávesy *Enter* jsou vložená data analyzována a následně je spuštěn odpovídající program, který je příkazem volán.

Přestože jsou v dnešní době k dispozici i jiná uživatelská rozhraní, příkazový řádek je stále využíván v operačních systémech, jako jsou UNIX, Linux nebo Windows. Výhodou je mimo jiné i vytváření skriptů, které mohou provádět několik příkazů po sobě, aniž by je musel jeden po druhém zadávat uživatel. Takový skript může potom jako naplánovaná úloha být spouštěn pravidelně, přestože uživatel není přítomen. Nevýhodou je nutná znalost příkazů a dostupných parametrů, stejně jako absence grafického prostředí, kde si uživatel lépe uvědomuje, co dělá.

Pro zvýšení uživatelského komfortu vznikaly programy, využívající textové uživatelské rozhraní (*angl. text user interface, TUI*). Tyto programy místo jednoho řádku pracují s celou obrazovkou monitoru a vytváří oblasti pro správu dat grafickou formou. K rozlišení částí programu využívají barevných pozadí a speciálních znaků ASCII tabulky, kterými se vytváří rámečky, ohraničující první typy dialogových oken a umožňující zadávání údajů do přesně vymezených oblastí. Tímto způsobem jsou vytvářeny i první vizuální souborové manažery jako Norton Commander [3]. Textová uživatelská rozhraní také umožňují oproti práci v rozhraní příkazového řádku přehlednější prostředí pro programování, zdrojový kód je zobrazován v několika řádcích pod sebou. Klávesové šipky pak umožňují pohybovat se nejen mezi znaky na řádku, ale přeskakovat na různé řádky obrazovky. Příkladem může být verze QuickBasic, určená pro operační systém MS-DOS.

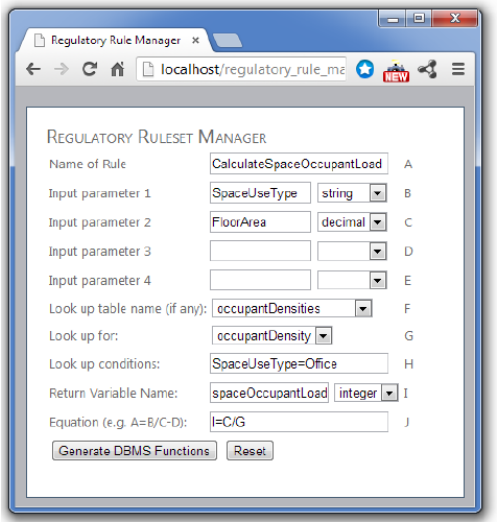


Obr. 1 Uvítací obrazovka programu QuickBasic [4].

Zařízení, používané pro jednoduchou obsluhu malého počtu přednastavených úloh, využívají řízené menu rozhraní (*angl. menu-driven interface*). Jejich uživatel má k dispozici omezený a předem nastavený počet položek menu, ze kterých si vybírá požadovanou činnost. K výběru používá zpravidla tlačítek nebo dotykové obrazovky. Volbou jedné položky může být zobrazena nová obrazovka s upřesňujícími volbami.

Řízené menu rozhraní lze nalézt na platebních terminálech, informačních tabulích nebo audio vizuálních zařízeních jako jsou kabelové televize a mp3 přehrávače. Toto rozhraní využívaly také starší typy mobilních telefonů.

Rozhraní, určené pro získávání většího množství informací od uživatele, je založené na formulářích (*angl. form-based interface*). Uživatel pomocí tohoto rozhraní může zadávat různé informace, jako je objednávka materiálu, rezervace letenky, stížnost na hotelové služby, a podobně.
Formuláře nabízí pro podrobný popis různé ovládací prvky, např. textová pole pro zadání údajů, seznamy pro výběr jedné nebo více možností ze zobrazeného listu, přepínače pro výběr jedné z nabízených možností, zaškrtávací políčka pro označení několika voleb nebo tlačítka, sloužící pro provedení potřebné akce. Uživatel pomocí prvků může zadávat, měnit nebo mazat údaje v systému.



Obr. 2 Formulářové rozhraní [5].

Grafické uživatelské rozhraní (*angl. graphical user interface, GUI*) patří mezi nejrozšířenější rozhraní. Vznik prvního grafického uživatelského rozhraní lze datovat do sedmdesátých let minulého století, kdy firma Xerox PARC vyvinula systém používající okna, ikony, menu a ukazatel (*angl. windows, icons, menus and pointing device, WIMP*) pro interakci uživatele a počítače[6].

Uživatel pro komunikaci se zařízením nepoužívá pouze klávesnici, ale i myš (později trackpoint, trackball, touchpad) pro ovládání ukazatele. Grafické uživatelské rozhraní zobrazuje uživatelská data jako seznamy ikon, zpravidla definujícími typ uživatelských dat. Soubory a adresáře jsou zobrazovány v oknech, která je možné přesouvat po obrazovce monitoru, schovávat nebo měnit jejich velikost. Ikony nepředstavují pouze soubory, ale i programy, které jsou spouštěny ve vlastních oknech.
Grafické rozhraní je založeno na intuitivnosti používání, aby i méně zkušený uživatel dokázal poměrně rychle pochopit používání objektů v rozhraní. Programy, určené pro editaci textu a obrázků, pracují zpravidla na bázi WYSIWYG (*angl. what you see is what you get*), umožňující uživateli pracovat s rozhraním tak, že vzhled jeho vytvořené práce je identický s výstupem, kterým může být například tisk na papír[7].

S vývojem mobilních zařízení (telefony, přenosné počítače, PDA) se rozvinulo dotykové uživatelské rozhraní (angl. touch user interface, TUI). Uživatel komunikuje se zařízením pomocí doteku obrazovky, zpočátku pomocí stylusu, později pomocí prstů. Doteky fungují podobně jako myš v grafickém uživatelském rozhraní, a je možné pomocí nich aktivovat jednotlivé prvky. Součástí rozhraní je i klávesnice, rovněž ovládaná pomocí dotyku. Systém reaguje na doteky zpravidla tzv. haptickou odezvou, potvrzením uživateli, že jeho dotek byl zaznamenán.
Dotykové uživatelské rozhraní podporuje také ovládání pomocí gest. Umí rozeznávat dotyk jedním prstem nebo více a podle toho reagovat. Například dotyk dvou prstů a jejich vzájemné oddalování po obrazovce lze vyhodnotit jako pokyn pro přiblížení, respektive zvětšení právě zobrazovaného obsahu. Podobným způsobem může systém reagovat i na sílu dotyku na obrazovce. Při kresbě čar ve vybrané aplikaci umožňuje systém zaznamenávat sílu dotyku a na základě síly tlaku uzpůsobit tloušťku vykreslované čáry[8].

Jedním z typů uživatelského rozhraní je rozhraní přirozeného jazyka uživatele (*angl. natural language user interface, NLUI*). Uživatel komunikuje se zařízením psaním slov dle specifikované gramatiky. Na podobném principu funguje i rozhraní

pro komunikaci se zařízením bez nutnosti používání klávesnice. Hlasové uživatelské rozhraní (*angl. voice user interface, VUI*) využívá technologii rozpoznání řeči, takže uživatel ovládá zařízení pomocí hlasových povelů. Díky tomu lze uživatelem zaznamenávat do zařízení zprávy, ovládat fungování zařízení, a podobně.

Hlasového uživatelského rozhraní využívají virtuální asistenti, kteří jsou součástí operačních systémů a ulehčují práci s nacházením a zobrazením vyhledávaných údajů v zařízení nebo na internetu, případně ovládáním přehrávačů[9].

Relativně mladé rozhraní pro komunikaci uživatele se zařízením je rozšířená realita (*angl. virtual reality, VR*). Zařízení vytváří simulované prostředí, promítané buď do reálného světa nebo mimo něj (zobrazuje se v zařízení). Uživatel komunikuje se zařízením ve virtuální realitě pomocí ovládacích prvků, které mohou simulovat například řízení dopravního prostředku pro přípravu uživatele před používáním reálného stroje, nebo pomocí ručních gest, která jsou snímána a rozpoznávána senzory zařízení, které pak podle typu gesta adekvátně reaguje změnou chování programu.

Obr. 3 Microsoft HoloLens [10].

3.2 Návrh uživatelského prostředí

S rostoucím počtem počítačů a jejich uživatelů rostl i zájem o studium problematiky interakce mezi uživatelem a zařízením, respektive člověkem a počítačem. V 80. letech minulého století, kdy se začaly do domácností dostávat počítače jako Apple Macintosh, Commodore 64 nebo IBM PC 5150, se začalo čím dál více diskutovat téma snadné a efektivní práce i méně zkušených uživatelů s počítači. Dané téma se brzy stalo i tématem akademického výzkumu, což vedlo ke vzniku

výzkumného oboru interakce člověk-počítač (*angl. human-comupter interaction, HCI*)[11].

Zmíněný výzkumný obor v sobě zahrnuje velké množství jiných vědních disciplín, například psychologii, filosofii, antropologii, ergonomii, fyziologii nebo sociologii. Zaměření je i na design a tvorbu uživatelských rozhraní tak, aby byly co nejjednodušší a nejlépe použitelné pro daného uživatele nebo skupinu uživatelů[12].

Design uživatelského rozhraní se zaměřuje na předvídání potřeb uživatele a zajištění, že rozhraní bude tvořeno prvky, kterým uživatel porozumí a bude je umět používat. Design tak může být tvořen vstupními prvky (textová pole, přepínače, tlačítka), navigačními prvky (posuvníky, záložky, stránkovače) a informačními prvky (nápovědy, ikony, indikátor průběhu - progress bar)[13].

Při tvorbě uživatelského rozhraní by se neměly opomíjet následující principy[13]:

- **Jednoduché prostředí** - rozhraní by nemělo obsahovat nedůležité prvky a mělo by být jednoduché, přehledné a popisky a zprávy by měly být v jazyce, kterému uživatel rozumí.
- **Konzistence a známé ovládací prvky** - používáním ovládacích prvků, které se vyskytují v jiných systémech s podobným chováním, je uživatel schopen snadněji porozumět uživatelskému rozhraní a pracovat s ním i v dalších částech aplikace.
- **Účelné rozvržení stránky** - správně zvolená struktura rozhraní může pomoci upozornit na podstatné údaje a lepší orientaci v něm.
- **Práce s barvami a texturami** - využití barev, textur, světla a kontrastu může pozornost nasměrovat na požadované místo nebo naopak odpoutat pozornost jinam.
- **Využití typografie pro přehlednost a hierarchii** - přehlednost a lepší orientaci v uživatelském rozhraní lze vhodným způsobem zvýšit na základě zvolené velikosti písma dle hierarchického uspořádání.
- **Zpětná vazba systému** - uživatel by měl být neustále informován o tom, co se v systému děje, ať už se jedná o změnu stavu, průběh prováděné akce nebo vzniklou chybu.

- **Původní smysl systému** - při tvorbě uživatelského rozhraní by měl stále brán zřetel na předvídání potřeb uživatele a snížení jeho pracovní zátěže. Je vhodné umožnit u některých prvků předvybrané nebo předvyplněné údaje.

Přístupů k návrhu uživatelského rozhraní existuje více v závislosti na úhlu pohledu, charakteru daného systému nebo potřeb uživatele či skupiny uživatelů. Mezi starší typy návrhů patřil například důraz na technologie a data (*angl. Data-Centered Design*), kde se uživatel musel spíše přizpůsobit výslednému rozhraní[12].
Opačným způsobem jde proces tvorby návrhu designu, zaměřeným na uživatele (*angl. User-Centered Design, UCD*). Proces je iterativní a zahrnuje pochopení uživatele a jeho způsob práce, specifikaci uživatelských požadavků, tvorbu návrhu designu, implementaci a testování, přičemž cílem je aktivně zapojit uživatele do všech fází vývoje, čímž lze docílit účinnějších, efektivnějších a bezpečnějších produktů[14].
Proces se nesnaží říct, jakou metodou postupovat v každé fázi. Naopak obsahuje množství metod, jak zapojit uživatele do tvorby. Cílem není pouze analýza a návrh produktu dle uživatelových specifikací, ale také ověření a potvrzení, že potřeby uživatele byly pochopeny správně.
První fází iteračního procesu designu zaměřeného na uživatele se zaměřuje na specifikace způsobu použití. Fáze se zaměřuje na poznání uživatele či skupiny uživatelů, kteří budou s novým produktem pracovat, k čemu ho budou používat a za jakých podmínek ho budou používat[15].
Ve druhé fázi se specifikují požadavky na nový systém a uživatelské cíle, pro které se nový produkt plánuje. Třetí fází je tvorba konkrétního designu produktu, což zahrnuje kompletní tvorbu od základních návrhů až po konkrétní řešení. Ve finální fázi pak dochází ke kontrole a zhodnocení výsledného řešení konkrétními uživateli.[15]

Zatímco design zaměřený na uživatele je zaměřený především na informační systémy, design zaměřený na člověka (*angl. Human-Centered Design, HCD*) zasahuje do různých rovin tvorby designu. Tento přístup je rozdělen do tří fází, které se také neustále opakují.

V počáteční fázi tvorby nového produktu se stává, že zadání a požadavky na produkt není řešen s budoucími uživateli, ale jejich zástupci, kteří jsou přesvědčeni o tom, jak má výsledky produkt vypadat a fungovat. Výsledný produkt pak ve finále skutečným uživatelům, kterým měl produkt ulehčit práci, přinejmenším nepomáhá. Proces designu, zaměřeného na uživatele, se snaží poznat právě potenciálního uživatele, pochopit jeho styl práce, potřeby a představy. Cílem je získat o uživateli co nejvíce informací, jak rozhovory, tak sledováním způsobu práce[16].

Další fáze tvorby přetváří zjištěné údaje a představy do návrhů řešení. Tato fáze může být nesprávně řešena jedním směrem, který je potenciálně nesprávným řešením, protože představy zadavatele mohou být pochopeny jinak, než byly myšleny. Proces designu, zaměřeného na uživatele pracuje s tím, že jedno řešení nemusí být správné, a proto se snaží přicházet s mnoha různými a rozmanitými návrhy, z nichž nejslibnější přetváří do funkčních prototypů[16].

Závěrečná fáze představuje funkční prototypy zákazníkům a dochází k testování reálnými uživateli. Při vytvoření jediného řešení může nastat situace, že uživatelé výsledné řešení odmítnou nebo ho minimálně nebudou používat, protože nebude ulehčovat práci. Proces designu, zaměřeného na uživatele proto překládá několik funkčních prototypů a sleduje reakce reálných uživatelů na používání. Následné podněty pomáhají finální řešení vylepšovat a upřesnit, který návrh je nejvhodnější pro uživatele[16].

Závěrečná fáze ukončuje jeden cyklus iterace, který vrací proces opět na začátek, kdy se sleduje chování zákazníka a jeho potřeby pro správnou funkci produktu. Po několika iteracích se vytváří finální představa produktu, který následně může být vyvíjen.

V roce 2004 definoval Peter Morville design, orientovaný na zkušenost nebo prožitek uživatele při používání produktu nebo služby (*angl. user experience, UX*). Tvorba není soustředěná na tvorbu uživatelského prostředí, ale na uživatele a jeho zkušenost s produktem či službou, zahrnující emoce, preference, reakce či chování, a to nejen během používání, ale před a po použití.

V rámci uživatelské zkušenosti, díky které například uživatel získá důvod, proč se vracet právě k danému produktu či službě, definoval Peter Morville sedm aspektů, které mají vliv na uživatelskou zkušenost[17]:

- **Dohledatelnost** - (*angl. findable*) vhodnou volbou a optimalizací způsobů dohledatelnosti lze dosáhnout toho, že uživatel při používání vyhledávacích systémů nalezne právě ten produkt či službu, který firma či společnost nabízí.
- **Užitečnost** - (*angl. usable*) je důležité vědět, zda uživatel považuje produkt za užitečný tím, že mu pomohl získat to, co potřeboval, když produkt použil.
- **Přístupnost** - (*angl. accessible*) pozitivní uživatelskou zkušenost by produkt měl přinášet i uživatelům s omezením. Součástí návrhu produktu by měla být i implementace pravidel pro bezbariérový přístup k používání.
- **Přitažlivost** - (*angl. desirable*) firemní značka, image společnosti a další emoční prvky mohou na uživateli zanechat takovou uživatelskou zkušenost nebo prožitek, že bude produkt nebo službu upřednostňovat před konkurencí.
- **Důvěryhodnost** - (*angl. credible*) důležitým aspektem uživatelské zkušenosti je i důvěra uživatele k informacím, které se dozvídá v produktu, který používá. Pokud uživatel nalezne nepravdivé informace, jeho důvěryhodnost a tím i uživatelská zkušenost s používáním produktu nebo služby klesá.
- **Užitečnost** - (*angl. useful*) na základě získávání informací od uživatelů se hledají stále nové způsoby a metody úprav produktů a služeb takovou formou, aby pro uživatele byly co nejužitečnější.
- **Hodnotnost** - (*angl. valuable*) produkt musí dodávat uživatelům hodnotu. Pokud není produkt zaměřen na tvorbu zisku, měla by uživatelská zkušenost zanechat dobrý dojem z použití. Pokud produkt na tvorbu zisku zaměřen je, měla by spokojenost uživatele s produktem dosáhnout jeho dalšímu používání.

Všechny aspekty sestavil Peter Morville do tzv. UX plástve (angl. *UX honeycomb*). Při jejím následném zkoumání byly jednotlivé aspekty rozděleny do třech různých oblastí, které se v některých aspektech překrývají[18]:

- **Mínění** - (*angl. think*) co si uživatel o daném produktu myslí? Považuje ho za užitečný? Je pro uživatele přínosný, hodnotný? Dá se považovat za důvěryhodný?
- **Pocity** - (*angl. feel*) cítí uživatel důvěru k danému produktu při jeho používání? Má tak pozitivní zkušenost s produktem, že ho chce nadále používat a upřednostňovat před konkurenčními produkty?
- **Použití** - (*angl. use*) když uživatel vyhledává řešení pro své potřeby, dokáže produkt jednoduše nalézt? Když produkt nalezne a rozhodne se ho vyzkoušet, může ho používat bez jakýchkoliv omezení? Když produkt bude používat, bude mít pozitivní uživatelskou zkušenost z práce s daným produktem?

Obr. 4 UX Honeycomb [18].

3.3 Tvorba uživatelského prostředí

Již při vytváření jednotlivých návrhů a prototypů uživatelských prostředí je možné se setkat s několika různými způsoby vzhledu. Většina těchto vzhledů by se dala rozdělit do čtyř různých typů, které se od sebe liší. Rozdíl mezi nimi lze nejlépe sledovat a definovat na webových aplikacích, které se zobrazí v internetovém prohlížeči a kde je velikost zobrazovacího okna velmi variabilní.

Statický, nebo-li fixní vzhled je založen na zjištění nejběžnějšího rozlišení obrazovky uživatele, a nastavení pevně stanovené velikosti obsahu, udané v pixelech. Struktura obsahu se nemění a velikost, především šířka, je nastavena na jednu hodnotu, a to bez ohledu na to, zda jsou dostupnější větší rozlišení zobrazovacího okna.
U větších rozlišení obsahuje fixní vzhled zpravidla prázdné plochy po obou stranách zobrazovacího okna. U menších rozlišení dochází zobrazení pouze části fixního vzhledu a uživatel musí zbytek obsahu zobrazovat pohybem posuvníku. V případě zobrazení fixního vzhledu u mobilních zařízení je obsah zobrazen celý, avšak často zmenšený tak, aby se přizpůsobil velikosti zobrazovacího okna, což mívá za následek až nečitelný obsah[19].

Plovoucí, nebo-li fluidní vzhled není založen na definici velikosti počtem pixelů, ale počtem procent. Struktura obsahu se nemění a velikost, především šířka, je nastavena procentuálně. To umožní roztáhnout obsah na celou obrazovku, čímž se dosáhne odstranění prázdných ploch po stranách obsahu. Při menším rozlišení je procentuálně velikost obsahu zmenšena a obsah je stále zobrazen kompletní[19].
Komplikace fluidního vzhledu se objevují při použití některých prvků jako obrázků nebo videa. V mobilních zařízeních navíc vzhled vypadá nevzhledně, v případě využití vícesloupcového rozložení pak mají sloupce tak malou šířku, že řádek obsahuje několik málo slov a obsah může být rozložen do několinásobně většího počtu řádků, než je tomu při zobrazení v oknech s větším rozlišením. Příliš velké zmenšení šířky může dokonce způsobit i přetečení obsahu jednoho sloupce do dalších.
Responsivní vzhled využívá výhod fluidního designu a posouvá ho na jinou úroveň. Velikost některých elementů upravuje také procentuálním definováním šířky a automatickým nastavením výšky (například u obrázků). Nevýhody fluidního vzhledu jsou řešeny pomocí definování různých stylů zobrazení, určených pro konkrétní rozsahy velikostí zobrazovacích oken.
Modul Media Queries, který je součástí kaskádových stylů (*angl. cascade style sheets, CSS*), umožňuje definovat blok stylů, které se použijí jen v případě, se zobrazovací okno svým rozlišením zasahuje do daného bloku. Dokud se zobrazovací okno při změnách své velikosti rozlišení stále vyskytuje v definovaném rozsahu, je použit

stále stejný blok stylů a velikost jednotlivým prvků struktury je procentuálně upravována. Jakmile je velikost rozlišení překročena, daný blok stylů se přestane používat a je vybrán jiný blok stylů.
Responsivní vzhled díky použití Media Queries reaguje na změnu velikosti rozlišení zobrazovacího okna změnou struktury obsahu. Vícesloupcový obsah je změněn zpravidla na jeden sloupec, velké obrázky jsou nahrazeny obrázky menšími (nedochází k degradaci zobrazení), a podobně.
Díky využití responsivního vzhledu je možné vystačit pouze s jednou šablonou, která se pomocí změny stylů překresluje. Nevýhodou je načtení kompletního obsahu včetně částí, které se nevyužívají, a větší úsilí a testování při navrhování jednotlivých bloků stylů[19].

Adaptivní vzhled na rozdíl od responsivního nepracuje s rozsahy procentuálních velikostí prvků struktury obsahu. Místo toho je vytvořeno několik souborů s pevně nastavenou strukturou obsahu pro dané velikosti. Na serveru je prováděna kontrola rozlišení zobrazovacího okna, server následně vrací optimální kód a pouze ta data (konkrétní vzhled, styly, skripty), které se využijí. Tím se možné dosáhnout i rychlejšího zobrazení u velmi malých zobrazovacích oken, protože server posílá menší množství dat (např. velikosti obrázků), než by poslal na zařízení s velkým rozlišením zobrazovacího okna[20].
Při tvorbě adaptivního vzhledu je běžné, že se vytváří nejdříve šest základních a nejpoužívanějších rozlišení obrazovek, respektive jejich šířek, a to 320, 480, 760, 960, 1200 a 1600 pixelů. Každé mobilní zařízení tak zobrazuje takovou obrazovku, která odpovídá jeho velikosti. Na mobilních zařízeních, jako je smartphone nebo tablet, se v podstatě očekává jedna velikost zobrazovacího okna a není tedy třeba zohlednit okamžitou změnu velikosti. Podobným způsobem lze uplatnit adaptivní vzhled i u dalších zobrazovacích zařízení, jako například terminály, které slouží jako informační nebo objednávací systémy.
Problém se může objevit v situaci, kdy se adaptivní vzhled zobrazuje na počítači, respektive na monitoru. Zde není výjimkou, že se zobrazovací okno mění, protože webový prohlížeč může být uživatelem upravován tak, aby lépe vyhovoval jeho požadovanému rozložení aplikací na obrazovce. Zde adaptivní vzhled naráží na své

limity, protože je koncipován jako kolekce připravených fixních obrazovek pro různé velikosti. Okamžitá změna velikosti prohlížeče se tedy neupravuje, jako při responsivním vzhledu a dokud nedojde ke změně na již definovanou šířku, je nutné v zobrazovacím okně posouvat obsah posuvníky, aby bylo možné vidět kompletní obsah, určený pro větší šířku [21].

Při vývoji webového obsahu je třeba se rozhodnout, zda obsah řešit pomocí responsivního nebo adaptivního vzhledu. Každá volba má svá pozitiva a negativa. Responsivní vzhled je často jednoduše implementovatelný, je u něj však menší kontrola nad velikostí zobrazovací obrazovky a na mobilních zařízeních může být zátěží pomalejší načítání obsahu . Adaptivní vzhled je laděný na míru konkrétní velikosti, a proto může být uživatelský zážitek větší. Data se také načítají rychleji, protože je stahováno pouze to, co je určeno pro konkrétní velikost. Podchycení různých velikostí obrazovek však může být náročnější na vývoj. Každé nové zařízení, obsahující jiné než implementované rozlišení, může vyžadovat vytvoření dalšího vzhledu právě pro tento typ [21].

3.4 Adaptace rozhraní dle uživatele

K návrhu adaptivního uživatelského prostředí se využívá různých pohledů. Jedním pohledem je sledování chování uživatele a adaptaci organizace obsahu používané aplikace nebo adaptaci ovládání na základě fyzických omezení uživatele. Jiným pohledem může být adaptace rozhraní dle zaměření pracovní činnosti uživatele a odstínění částí, které pro svou práci nepotřebuje.

Jedním z příkladů může být sledování pohybu očí uživatele po obrazovce. Při provozu automatických vozidel se vycházelo z toho, že řidič, který v dané chvíli funguje spíše jako dozorce funkčnosti vozidla, může časem snížit svou pozornost a v takovém případě by mu mohli notifikační ikony pomoct ve sledování provozu. Z tohoto důvodu bylo realizování testování za účasti dobrovolníků. Na obrazovce viděli devět různých ikon, oznamující stav vozidla a okolí a systém sledoval, na kterých ikonách se zrak dobrovolníka ustálí. Testování napomohlo odhalit nejčastěji sledovaná místa na obrazovce pro lepší adaptaci prostředí a zobrazování ikon na obrazovce[22].

Adaptace prostředí řidiče je velice důležitá. Frustrace z nečekaných situací při spěchu na důležitou schůzku může vyvolat u řidiče osobního automobilu agresivní chování vůči ostatním účastníkům silničního provozu, které může vést až k dopravním nehodám. Frustraci mohou vyvolat situace jako nečekané zpomalení z důvodu práce na silnici a vytváření zpomalující kolony nebo z důvodu komplikací při hledání volného místa k zaparkování. Adaptací prostředí vhodnou signalizací, poskytováním informací o zdržení nebo hraním uklidňující hudby je možné frustraci postupně snižovat a zabránit tak možným problémům v průběhu jízdy [23].

Jiná studie sledovala možnosti adaptace uživatelů s poruchou hybnosti pro ovládání obrazovky kurzorem myši. Pro takové uživatele je nasměrování klasického kurzoru a kliknutí na elementy grafického rozhraní poměrně komplikované. Z tohoto důvodu byly vytvořeny další dva alternativní virtuální kurzory, které měly za cíl zjednodušit pohyb po rozhraní. První virtuální kurzor sledoval oblast pohybu a umožnil vybrání elementu, který zasahoval do dané oblasti, aniž by bylo vyžadováno přesné najetí na element. Druhý kurzor využíval pomocné numerické klávesnice, která umožnila snížení manuálního přesunu kurzoru automatickým pohybem v jednom z osmi směrů od aktuální pozice na obrazovce[24].

Pro uživatele, jejichž omezení nedovolí ovládání počítačových periférií pomocí svých končetin, jsou vyvíjeny systémy, které umožňují interakci s počítačem pomocí pohledů očí. Uživatelské prostředí obsahuje tlačítka pro ovládání aplikace a uživatel vybírá konkrétní tlačítko zastavením pohledu na vybraném tlačítku. Doba pohledu, která má být vyhodnocena jako kliknutí myší na tlačítko, se však může lišit nejen v závislosti na uživateli, ale také na typu zobrazovaného obsahu na tlačítku (text, obrázek). Z tohoto důvodu se vyvíjí také aplikace, které se adaptují na konkrétní uživatele a zaznamenávají pohledovou dobu pro každého zvlášť [25].

Zvýšení uživatelského komfortu je možné aplikovat i adaptací domácích spotřebičů, což by mohli ocenit pohybově limitované či starší osoby. Klasické pračky obsahující otáčecí hlavu jako ovladač, kterým se definuje teplota a typ materiálu praného prádla, případně přepínací tlačítka pro definici počtu otáček a podobně. Ve studii, která se problémem ovládání automatické pračky zabývala, bylo navrženo nové rozhraní, umožňující uživateli nejen nastavit všechny parametry pomocí LCD panelu, ale také definici parametrů pomocí jednoduchého posuvníku a speciální

metoda nastavuje všechny potřebné parametry. Také bylo přidáno hlasové zařízení, které umožnilo uživatelů manipulaci se zařízením pomocí hlasu. Implementovaný kontroler zajišťoval bezproblémové přepínání mezi všemi třemi vstupy. Navíc byl vyvinut software, spustitelný na mobilním zařízení nebo laptopu, které mohlo komunikovat s automatickou pračkou bezdrátově, což umožnilo ovládání pračky a sledování jejího stavu bez nutnosti být u ní přítomen [26].

Adaptace v domácnosti se nemusí zaměřit pouze na komunikaci přes povely, které obyvatel domu vydává. Architektura inteligentního kontroleru může mít více scénářů, jak pomáhat. Například pokud se obyvatel domu probudí a vydá povel na rozsvícení světel, kontroler by měl zjistit podle času, o jak hlubokou noc se jedná, a podle toho zvolit intenzitu světla. Navíc by měl být schopen zjistit, ve které části má k rozsvícení světel dojít. Jiným scénářem může být situace, kdy příchozí osoba zapomene zamknout vstupní dveře a v průběhu dne to neudělá. Večer, když se chystá jít spát a zhasne, systém by měl tento stav vyhodnotit a připomenou například zvukovým upozorněním, že by z důvodu větší bezpečnosti bylo vhodné vstupní dveře zamknout [27].

3.5 Adaptace rozhraní dle role uživatele

V rámci podnikových aplikací se vyskytuje jiná komplikace. Aplikace pro řízení vztahů se zákazníkem (CRM) nebo plánování podnikových zdrojů (ERP) obsahují velké množství nástrojů a funkcí, dostupných různým typům uživatelů, aniž by některé z nabízených funkcí pro svou oblast pracovní činnosti potřebovali. V tomto případě je tedy cílem omezit uživateli nabízené nástroje a funkce pouze na ty, které skutečně potřebuje. Potřeby každého uživatele jsou definovány uživatelskou rolí, která určuje potřebné nástroje a ostatní potlačí, čímž je dosaženo zjednodušení a větší přehlednosti uživatelského rozhraní.

Jednou z možných cest je Role-Based UI Simplification (RBUIS), který je založen na zjednodušení uživatelského rozhraní minimalizací sady funkcí a rozvržení jednotlivých elementů, optimalizovaného pro danou uživatelskou roli. Mechanizmus je využíván u studia Cedar, které využívá různé zdroje, jako jsou pravidla adaptace, uživatelská zpětná vazba a sledování chování, k uzpůsobení

uživatelského rozhraní s množstvím různých elementů tak, aby uživateli byly zobrazeny jen ty prvky formulářových oken, které skutečně potřebuje[28].

Alternativou je Adaptive UI (AUI), vývojové prostředí nabízející možnost definice pohledů uživatelského prostředí a jeho adaptace podle potřeb uživatele. Toto řešení je různě rozvíjené pro konkrétní řešení. Jedním z nich je například Self-Adaptive UI (SAUI), které nabízí integrované modelově orientované vývojové prostředí, nabízející možnost samo rozhodování o adaptaci uživatelského rozhraní podle obsahu či kontextu využití. Na demonstrované případové studii univerzitní knihovny systém prověří, zda je uživatelem zaměstnanec knihovny nebo student, případně zda je rezervace realizována přes laptop, mobilní zařízení nebo knihovní terminál. Na základě získaných zjištění pak přizpůsobí uživatelské prostředí konkrétní roli uživatele a konkrétnímu používanému zařízení[29].

Podobnou cestou se vydali tvůrci frameworku, nazvaného Component-Based Adaptive User Interface (CoBAUI), rozděleného na čtyři hlavní komponenty. Komponenta Context Provider monitoruje a získává potřebné informace z různých zdrojů, ať už je to aplikace, síť nebo senzory. Tato data získává na základě událostí v aplikaci nebo v pravidelných časových intervalech. Komponenta Rule Provider je používána pro získávání a předávání adaptativních pravidel za běhu aplikace. Umožňuje také dynamické přidávání a odstraňování pravidel v průběhu používání a změny předává dál. Komponenta Rule Evaluator vyhodnocuje aktuální informace o kontextu a poskytuje adaptačnímu kontroleru vyhodnocení informace o konkrétním pravidle a konkrétním kontextu. Komponenta Adaptive Widget je rozšířena komponenta uživatelského rozhraní, která je využívána pro definici a vzhledu zobrazovaného uživatelského prostředí, které je možné za běhu modifikovat. Každá komponenta se musí registrovat v adaptačním kontroleru (Adaptation Controller), aby mohla být adekvátně využívána ve frameworku. Právě kontroler řídí komunikace mezi jednotlivými komponentami a umožňuje flexibilní přidávání a odebírání jednotlivých komponent.

Realizované řešení se tak přizpůsobilo preferencím uživatele podle jeho potřeb, kdy podle jeho dominance ovládání bylo ovládací menu umístěné na levé či pravé straně, preference zvýraznění prostředí umožnilo aplikaci zobrazit ve světlém nebo tmavém prostředí a podle míry používání aplikace se nezkušeným uživatelům

zobrazovali ovládací prvky menu s ikonou i popiskem, zatímco zkušenějším uživatelů se již zobrazovaly pouze ikony[30].

3.6 Adaptace v dostupných řešeních

Ve firemním prostředí výrobního zaměření vzniká velké množství informací z finančního, logistického a výrobního či údržbového sektoru. Každý sektor většinou pracuje s jiným druhem dat pro svůj provoz. Vedení firmy naopak potřebuje mít všeobecný přehled o dění v podniku, což vytváří vhodné prostředí pro adaptaci reportovacích systémů, které nabízí množství různých komponent, schopných se přizpůsobit specifickému případu užití.

Mezi nejznámější řešení patří platforma Tableau, která se zaměřuje na rychlou analýzu dat v reálném čase, a to z různých zdrojů, na které je schopná se připojit. Hlavní silou je tvorba interaktivních dashboardů, na kterých je možné adaptovat vizualizační prvky prezentace dat, ať již jde o rozmanité grafy nebo barevně rozlišené tabulky. Uživatel si při tvorbě dashboardu může určit, jak velký bude vybraný graf, jakého typu a v jakém formátu bude zobrazen, kde bude umístěn na zobrazovací ploše a v případě potřeby i lehce změnit zdroj nebo úplné odstranění daného elementu ze zobrazovací plochy[31].

Na podobném principu funguje i produkt společnosti Microsoft, a to Power BI. Tento systém je velice populární ve firemním prostředí, využívající i jiné produkty firmy Microsoft (Windows, Office, a pod.), protože nabízí uživatelské rozhraní velice podobné ostatním produktům této společnosti. Ačkoliv design jednotlivých elementů se může lišit od produktu, popsaného výše, je produkt Power BI založen na podobném principu. Umí také získávat data z různých zdrojů a vytvářet interaktivní dashboardy pro vizualizaci potřebných firemních dat[31].

Velkým hráčem na trhu s prezentací uživatelských dat ve firemním prostředí je i společnost SAP. Nabízí několik různých druhů nástrojů pro vizualizaci. Mezi ně patří například SAP Fiori, systém určený pro vytváření pracovního prostředí se zaměřením na uživatelskou zkušenost. Produkt SAP Hana umožňuje vytváření systémů pro vytváření podnikových modelů a odhadů vývoje trhu. K dispozici také nabízí integrované analytické nástroje a využívá inteligentní automatizace, založené na umělé inteligenci a robotizaci. Pro analýzu dat ve stejném duchu, jako je Tableau

nebo Microsoft Power BI, je pak nabízen produkt SAP Lumira, který nabízí podobné elementy pro dashboard, jako výše zmíněné konkurenční produkty[31].

3.7 Použité technologie při vývoji webové aplikace

Aby bylo možné aplikaci dále udržovat a vyvíjet, je nezbytné, aby byla vytvořena v programovacím jazyce a prostředí, které ovládá více programátorů ve firmě. Tím se eliminuje závislost podpory pouze na jednom vývojáři a aplikace je spravovatelné i v případě personálních změn. Z tohoto důvodu jsou navrženy technologie, které jsou ve firmě již používány pro vývoj lokálních aplikací.
Firemní prostředí využívá na klientské i serverové systémy operační systém Windows od firmy Microsoft. Aplikace bude vyvíjena jako webová, tedy bude hostována na serveru, provozujícím IIS (Internet Information Services).

.Net Framework je platforma, existující již od roku 2002. Obsahuje velké množství technologií, které usnadňují vývoj aplikací. Patří sem například Windows Presentation Foundation (WPF), používaný pro vývoj grafického uživatelského rozhraní především klientských aplikací. Také sem patří podpora vývoje webových stránek se serverovým skriptováním pomocí technologie Active Server Pages (ASP), která je použita pro vývoj aplikace. Za zmínku stojí také technologie Language Integrated Query (LINQ), která je využívána ve vyhledávání a selekci položek v kolekcích, které implementují rozhraní IEnumerable.
Aby bylo možné vyvíjet aplikace v této platformě, vytvořila firma Microsoft systém Visual Studio.NET. Ten podporuje různé druhy programovacích jazyků, především Visual Basic a C++. Navíc byl vytvořen nový programovací jazyk C#, jehož první verze byla vydána společně s první verzí .Net Frameworku.
Pro platformu .Net Framework není důležité, který jazyk je pro vývoj použít, což dává vývojářům možnost pracovat v tom jazyce, na který jsou zvyklí. Při kompilaci je každý kód přeložen do jednoho společného mezijazyka Common Intermediate Language (CIL), což je objektově orientovaný jazyk zásobníkového typu. Při spuštění aplikace je převeden do byte kódu a vytváří se .Net assembly, ze kterého Just In Time (JIT) kompilátor generuje nativní kód.

Pro tvorbu a zobrazení webových stránek se využívá značkovací jazyk HyperText Markup Language (HTML). Skládá se z několika povinných značek (tagů) a kolekce dalších, které je možné pro definici webové stránky použít. Prvotní verze byly tvořeny převážně bloky kódu, ohraničenými tagy , které neměly hlubší sémantický význam a nebylo zcela zřejmé, co který blok kódu konkrétně znamená.
Značkovací jazyk se postupně vyvíjel a současná verze HTML 5 přidává nové značky, které umožňují lépe definovat strukturu dokumentu webové stránky. Mezi tyto značky tak patří například <header></header> pro definici záhlaví dokumentu, <footer></footer>pro definici zápatí, <nav></nav> pro definici navigačního menu v dokumentu a podobně.
K rozšíření došlo také u formulářového objektu <input />, který umožnil zadat textovou hodnotu bez bližšího určení. S verzí HTML5 přibyly nové typy vstupního pole, takže je možné specifikovat typ pole jako číslo, datum, čas, email nebo také barvu. Ne všechny typy jsou podporovány některými webovými prohlížeči. Například typ měsíc nefunguje u webového prohlížeče Mozilla Firefox.

Aby bylo možné oddělit strukturu a obsah webové stránky od jejího vzhledu, byl vytvořen nový jazyk pro definici vzhledu, barev nebo fontů webových objektů. Tento jazyk, nazvaný Cascading Style Sheets (CSS) nejdříve umožňovaly pouze měnit vzhled jednotlivých elementů pomocí jednoduchých selektorů. Současná verze CSS3 však umožňuje definovat styl pro daleko specifičtější výběr elementů, například podle obsahu atributu, pořadí v listu, stavu elementu (aktivní, vypnutý, zakliknutý). Je tedy možné barevně odlišit první, poslední nebo každý šestý prvek v seznamu.
Součástí jazyka CSS je i definice typu a vlastního výstupního zařízení (media queries). Vývojář je schopen pomocí typu zařízení definovat jiné druhy stylů, čímž odliší vzhled webové stránky na obrazovce od webové stránky, určené pro tisk. Podobným způsobem je možné definovat různé styly vzhledu webové stránky podle specifických vlastností výstupního zařízení, například podle šířky či výšky zobrazovacího zařízení.
S rozvojem práce s kaskádovými styly se začaly objevovat požadavky na možnosti definování proměnných, které by bylo možné používat v průběhu vytváření kaskádových stylů. Výhoda by byla především v tom, že například změna barvy či

velikost fontu by se měnila pouze na jednom místě a pomocí proměnné by se automaticky změnila ve zbytku kaskádových stylů. Na základě těchto potřeb se začaly objevovat preprocesorové skriptovací jazyky, které definované styly kompilovaly do klasického jazyka CSS. Mezi tyto preprocesory se řadí například Syntactically awesome style sheets (Sass), Leander Style Sheets (Less) nebo Stylus.

Platforma vychází z předchozí verze, a to Active Server Pages, existujícího od druhé poloviny devadesátých let 20. století. S příchodem .Net Frameworku byla platforma přetvořena. Aplikace, vytvořené v ASP.NET, jsou předkompilovány a spouštěny přes Common Language Runtime (CLR). Díky kompilaci je také velké množství případných chyb v kódu již při vývoji.
V roce 2007 byla přestavena platforma ASP.NET MVC, která dělí aplikaci do třech samostatných vrstev. Hlavní vrstvou je kontroler (nebo řadič), který funguje jako vstupní bod. Uživatel volá jednotlivé metody kontroleru, zvané Actions, a kontroler na ně reaguje odpovídající odpovědí. V platformě kontroler získává data z modelové vrstvy, která obsahuje data ve formě doménových objektů, používaných v aplikaci. Získaná data následně kontroler předává pohledové vrstvě, která vytváří výslednou prezentaci obsahu uživateli.
Pohledová vrstva je tvořena zpravidla HTML stránkou typu Master Layout, která obsahuje hlavní kostru stránky. Většina odpovědí kontrolerů je realizována přes stránku typu View, obsahující hlavní obsah, vkládaný do Master Layout stránky. V některých případech se stránka View negeneruje. Jedná se například o případy, kdy je volán dotaz na data, generovaná v textovém formátu, jako je xml nebo json. V těchto případech se nevolá ani Master Layout stránka a data jsou odesílána pouze jako text.

V devadesátých letech 20. století byl vytvořen nový objektově orientovaný a multiplatformní skriptovací jazyk, který dostal název JavaScript. Nejčastěji se tento jazyk používá na webových stránkách jako klientský skript, který je součástí stránek. Má podobnou syntaxi jako jazyky Java nebo C a je také case sensitivní. Jeho kód se nekompiluje ale interpretuje samotným webovým prohlížečem po stažení stránky i s kódem.

JavaScript je využíván především pro práci s objekty na webové stránce a pro ulehčení práce vzniklo několik desítek javascriptových knihoven, které se na tuto činnost specializují. Jednou z těchto knihoven je i knihovna jQuery.

Knihovna jQuery se poprvé objevila v roce 2006 a umožnila zjednodušenou manipulaci s objekty v rámci Document Object Model (DOM), který popisuje webovou stránku jako dokument se stromovou strukturou, z nichž každý uzel je objektem, reprezentujícím určitou část dokumentu. Zatímco v dřívějších dobách musely být události na objektech DOM explicitně vyjádřeny voláním konkrétní metody uvnitř definice objektu, knihovna jQuery umožňuje události řešit mimo objekty a události řešit separátně, čímž není narušena struktura HTML prvků.

Pomocí knihovny je možné přidávat nebo ubírat jednotlivé objekty do struktury webové stránky, ověřovat či měnit obsah a vzhled jednotlivých objektů, získávat data ze serveru a posílat data zpět na server, a podobně.

Příkladem práce s jQuery je vyhledávání maximálních hodnot v tabulce. Pokud máme u každé položky jednoho sloupce stejný název třídy, umožňuje knihovna jQuery načíst do proměnné kolekci těchto položek a následně je procházet v cyklu.

Při komunikaci se serverem a předávání či získávání dat je zpravidla potřeba odeslat požadavek na server a nové načtení webové stránky. Ne vždy je však potřeba tak velký zásah do obsahu, aby bylo nutné nové načtení kompletní struktury. Změna jazykové mutace nebo získání některých dat nemusí nutně vyžadovat, aby stránka načítala i ostatní části webové stránky, pokud na ně nemá zásadní vliv.

Pro tyto účely byla vyvinuta technologie Asynchronous JavaScript and XML (Ajax), která komunikuje se serverem pomocí metod GET a POST, aniž by bylo vyvolána action metoda kontroleru, takže se stránka nepřekreslí kompletně, ale pouze ty její části, kterých se událost, spouštějící Ajax operace, tyká. Protože některé operace mohou trvat déle, poskytuje Ajax možnost zobrazovat informační okno, které upozorňuje uživatele aplikace na fakt, že dochází k načítání či posílání dat, a nemá tak pocit, že aplikace "zamrzla"

4 Teoretický návrh řešení

Ve firemním prostředí je sledováno množství blokovaného materiálu, uloženého ve skladovacích prostorech firmy. Oddělení logistiky má za cíl pravidelně sledovat veškerý blokovaný materiál, zjišťovat příčiny blokování a určit odpovědnou osobu, která zavede opatření k co nejrychlejšímu odblokování materiálu. Současně připravuje pravidelné reporty pro sledování množství blokovaného materiálu v rámci jednotlivých oddělení (středisek).
Ke sledování blokovaného materiálu je používána aplikace Blockstock, vytvořená v roce 2015 jako klientská aplikace pro operační systém Windows.

4.1 Předchozí stav aplikace

Současná aplikace je vytvořena pomocí Windows Forms a programovacího jazyka Visual Basic.NET. Je složena celkem ze 13 formulářových objektů, z nichž každý obsahuje na pozadí třídu, obsahující obslužné metody, reagující na události formuláře (načtení, stisk tlačítka, poklepání na řádek tabulky). Dále jsou součástí moduly, obsahující podpůrné metody pro připojení na databázi, export dat do tabulkového procesoru MS Excel, a podobně.

Aplikační role jsou pouze dvě - uživatel a administrátor. Uživatel pracuje s tabulkou dat, v níž může upravovat a doplňovat kterýkoliv záznam. Protože se zobrazují záznamy všech blokovaných položek, může využít možnost definice vlastního seznamu zobrazených sloupců a omezení počtu záznamů určením konkrétních hodnot ve vybraných sloupcích. Vytvořený filtr je třeba uložit na disk používaného počítače. Pro potřeby podrobnější analýzy je možné data exportovat do MS Excel, kde si uživatel zpracovává vlastní podklady. Dalšími možnostmi je zobrazení nejdražších blokovaných položek a nejdéle blokované položky.
Administrátor importuje data z externího systému do aplikace, definuje cíle pro maximální sumu blokovaného materiálu jednotlivých oddělení, určuje disponenty, tj. osoby zodpovědné za konkrétní typ materiálu, dále podklady pro reporty vedení a seznam uživatelům, kterým je pravidelně zasílána informace o novém importu dat.

Definice administrátora je součástí zdrojového kódu. Uživatelské jméno osoby, která má dostat administrátorské oprávnění, je vloženo do metody, vracející true nebo false, pokud uživatel je nebo není nastaven jako administrátor. Každá změna administrátorského oprávnění tedy vyžaduje rekompilaci a redistribuci aplikace.

Uživatel pracuje s tabulkou, která obsahuje více jak 2 000 záznamů, najednou však může vidět nejvýše 50 záznamů v závislosti na velikosti obrazovky. Tabulku je možné seřadit podle vybraného sloupce. Pokud chce uživatel vyselektovat pouze ty záznamy, o které se zajímá, musí vytvořit filtr, který je následně uložen na pevném disku klientského počítače, používaného k práci s aplikací. Tento způsob se ukázal jako nevhodný pro ukládání českých znaků v anglickém prostředí operačního systému (docházelo k odstraňování háčků nebo čárek nad některými českými znaky). Navíc si uživatel musel pamatovat, kam filtry ukládal pro případě výměny počítače.

4.2 Zpětná vazba

Existující řešení v průběhu času začalo narážet na změny v nastavení prostředí a bezpečnosti operačního systému, nasazených centrálním IT oddělením, což vedlo k tomu, že aplikace přestala vyhovovat jako nástroj, který by ulehčoval uživatelům práci ve sledováním blokovaného materiálu. Na základě požadavku vedení oddělení logistiky coby vlastníka procesu správy blokovaného materiálu byla svolána schůzka, na které zástupci jednotlivých oddělení definovali požadavky na změny ve funkčnosti.

Ve firmě pracuje nemálo zahraničních pracovníků, kteří neovládají češtinu natolik dokonale, aby mohli vhodně pracovat s aplikaci. Jedním z hlavních požadavků tedy byla implementace i německého a anglického jazyka do aplikace. Aplikace by si měla zapamatovat, jaký jazyk byl uživatelem zvolen, a při dalším spuštění by se již uživatelské prostředí mělo zobrazit v jazyce, který byl vybrán naposledy.

Velmi diskutované téma bylo množství sloupců, které se mají zobrazovat v aplikaci. Uživatelé mají podle své role ve firemním prostředí jiné potřeby a pro svou práci

využívají jiné sloupce. Návrhy na zobrazení pouze některých sloupců se neshodly na jasném počtu a výběru, proto padl návrh na ponechání výběru sloupců každému uživateli zvlášť s tím, že by si aplikace pamatovala daný výběr a při dalším spuštění si zobrazovala poslední nastavení.

Současná verze aplikace nabízí pouze vytváření uživatelských filtrů, přes které je možné vybrat si pouze některé údaje. Pokud ovšem uživatel potřebuje narychlo vyhledat ve vybraných záznamech konkrétní hodnotu, nemá možnost vyhledat pouhým zadáním hodnoty do odpovídajícího pole daného sloupce. Každé vyhledávání nutí uživatele buď vědět dopředu, co hledá, nebo exportovat data do souboru .xlsx a vyhledávání řešit v externí aplikaci. Byl tedy vznesen požadavek na možnost doplnění rychlého vyhledávání pomocí zadání konkrétní hodnoty u daného sloupce.
Mezi hlavní požadavky na změnu patřily:

- častější import dat do aplikace
- zrychlení načítání dat v tabulce
- rychlé vyhledávání v tabulce bez nutnosti vytváření filtru
- vytváření vlastních filtrů bez nutnosti ukládání na používané zařízení
- možnost uživatelského nastavení
- rozšíření české variace popisků o anglickou a německou.

4.3 Adaptivní chování aplikace

Návrh adaptivního chování aplikace by měl vycházet z existujících řešení, což se v případě této firemní aplikace může jevit jako problém. Neexistuje žádná shoda nebo standard pro postup vývoje adaptivních uživatelských rozhraní, už i proto, že lze nalézt větší množství rozmanitých cílů, ke kterým se má adaptivní rozhraní použít. Nabízené řešení, založené na tvorbě adaptibilních komponent, je zkoumáno déle jak 30 let a stále se nejedná o běžné řešení v produkčním prostředí [30].
Existující demonstrace adaptivního chování jsou převážně realizovány na mobilních zařízeních. Jejich hardwarová vybavení nabízí senzory a prvky, které pro adaptaci uživatelského rozhraní je velmi vhodné využívat. Většina mobilních telefonů

například obsahují gyroskop, který je možné využít pro sledování práce uživatele a vypozorování, zda je mobil používán levou nebo pravou rukou. Podle způsobu uchycení a klikání na tlačítko menu je pak možné adaptovat toto tlačítko přesunutím na tu stranu uživatelského rozhraní, které je pro prsty uživatele více dostupné a tedy lépe ovladatelné. Pomocí senzoru pro sledování světla v prostředí, kde se uživatel nachází, je možné a často běžné adaptovat uživatelské prostředí v případě šera či tmy přepnout vzhled do tmavého režimu pro potlačení vyzařování jasu a úlevu očím v tmavém prostředí [30].

Některá řešení nabízí skutečně nepřeberné množství pravidel, aplikovatelných při adaptaci uživatelského prostředí. Při používání grafické a zvukové interakce uživatele by se aplikace mohla stát hůře ovladatelnou ve chvíli, kdy se uživatel dostane do prostředí se zvýšeným hlukem, proto může pravidlo v situaci, kdy vyhodnotí zvýšený hluk prostředí, automaticky přepnout uživatelské rozhraní pouze do grafického režimu. Jiné pravidlo reaguje na barvoslepé uživatele, pro které přepne prostředí do černobílého režimu. Mezi další typy pravidel je možné zařadit slabý zrak , kdy se aplikace přizpůsobí zvětšením textu, nebo problém s kognitivní funkcí, ztěžující uživateli orientaci v komplexním uživatelském řešení, což je řešeno přepnutím zobrazovacího rozhraní v jednodušším módu a rozděleného do více kroků [32].

Zásadní informací, spojovanou s velmi komplexním adaptivním chováním uživatelského rozhraní, je potřeba vytváření takového prostředí pro uživatele s opatrností. Vedle skutečnosti, že adaptace prostředí může zlepšit používání vybrané aplikace, samotná adaptivní rozhraní představují výzvu pro důvěryhodnost a akceptování adaptivních prvků uživateli. Může se totiž stát, že adaptace uživatelského rozhraní vyvolá v uživateli pocit, že nemá kontrolu nad rozhraním, pokud se změny provádí automaticky [33].

Pro navrhovanou aplikaci je využito konceptu vývoje, založeného na modelu komponent [30]. Koncept využívá programovaní jazyk AngularJs, který není možné ve firmě využít pro chybějící know-how, proto bude způsob adaptace vytvořen pomocí knihovny jQuery. Aplikace bude webová, s mobilní verzí se nepočítá, proto nebudou implementovány prvky, využívající senzorů jako gyroskop nebo ambient.

Bude však využito konceptu sledování používání aplikace uživatelem a na to navázaná pravidla.

Uživatelské rozhraní původní aplikace nabízelo pouze jeden jazyk. V jiných firemních aplikacích byla vždy nabízena primárně čeština, kterou bylo možné změnit na jiný z podporovaných jazyků. Nová aplikace byla tedy navržena tak, aby si pamatovala výběr jazyka, který měl uživatel zvolený jako poslední. Při novém spuštění se pak aplikace zobrazí v jazyce dle uložené hodnoty.
Změna vzhledu v původní aplikaci nebyla možná, vždy se zobrazovala stejně pro všechny uživatele. Současný trend však nelze přehlédnout, množství moderních aplikací nabízí dva až tři vzhledy, aby si uživatel vybral dle svých preferencí nebo aktuální nálady. Bezpečnostní politika společnosti neumožňuje využívat kamery pro sledování a pozorování nálady přihlášeného uživatele. Nová aplikace byla tedy navržena tak, aby nabízela uživateli čtyři různé možnosti vzhledu. Po výběru uživatele se uchová vybraný vzhled před opuštěním aplikace a při další spuštění se uživatelské rozhraní zobrazí ve vybraném vzhledu.
Tabulka původní aplikace je klíčová pro uživatele a schůzka zainteresovaných osob ukázala, že každý potřebuje vidět jiné sloupce. Poskládání jednotlivých sloupců tak bylo spíše věcí kompromisu než potřeb všech uživatelů. Sloupce ve většinové shodě se zobrazovaly na hlavní obrazovce a ostatní byly schovány v pořadí. Pokud je chtěl uživatel zobrazit, musel využít posuvník a přesunout tabulku tak, aby dané sloupce zobrazil. Nová aplikace byla tedy navržena tak, aby dala možnost každému uživateli vybrat si svůj seznam sloupců. Tento seznam se následně uchová před opuštěním aplikace a při dalším spuštění se zobrazí uložený seznam sloupců.
Nová aplikace také byla navržena tak, aby umožnila uživateli všechna výše zmíněná nastavení určit při prvním používání aplikace. Na začátku aplikace zjistí, zda uživatel, který právě aplikaci spustil, již někdy aplikaci používal. V případě první návštěvy se uživateli nabídne možnost počátečního nastavení, tedy v jakém jazyce chce mít aplikaci zobrazenou, jaký vzhled mají mít ovládací prvky a pozadí a jaké chce používat sloupce tabulky.
Uživatel také bude mít možnost své nastavení kdykoliv změnit, tedy nebude muset úvodní nastavení využít a seznam sloupců nebo jazyk bude moct nastavit později.

V současné aplikaci není možné vyhledávat jinak než seřazením hodnot ve vybraném sloupci od nejnižší nebo nejvyšší hodnoty, případně pomocí předdefinovaných filtrů. Ve filtru si uživatel vybral sloupce, které chtěl zobrazit, a podmínky jednotlivých sloupců pro filtraci.

Nový návrh tedy zahrnuje možnost každého uživatele, aby nad tabulkou mohl zadávat u každého sloupce hodnoty, které hledá. Tím se mu umožní seznam záznamů by se omezil pouze na na ty řádky, které by splňují danou podmínku. Uživatel následně může zadat pro další omezení počtu řádků hledané hodnoty i do dalších sloupců a byly by zobrazovány pouze ty záznamy, které by vyhovovaly logickému součinu zadaných hodnot. Aplikace zaznamená průběžné vyhledávání, při dalším spuštění aplikace se uživateli zobrazí seznam záznamů, odpovídající poslední hledané kombinaci.

Podle nastaveného pravidla se vyhodnotí nejčastěji používané kombinace filtrování dat. Po naplnění podmínek daného pravidla se vytvoří nový uživatelský filtr a připojí se k filtrům, které byly definovány uživatelem přes danou akci. Adaptace nebude funkční u uživatelů s rolí *Host*.

Všichni uživatelé současné aplikace používají stejnou sadu ovládacích prvků, přestože v některých případech používají sotva polovinu. Pro návrh nové aplikace se objevily dva návrhy, jak umožnit uživateli používání pouze těch ovládacích prvků, které skutečně potřebuje.

Jednou možností bylo vytvoření pravidla sledování tlačítek, které uživatel vybírá v průběhu používání aplikace. Vytvořené pravidlo vyhodnotí nejčastěji používané volby a tyto následně přednostně nabídne. Ostatní volby se ukryjí do bloku, který bude v základním nastavení tyto volby schovávat, avšak na zvolení uživatele tyto volby zobrazí. Uživatel tak bude mít k dispozici nejdříve své možnosti a ty, které nepoužívá, budou schovány. Pokud začne v průběhu času nějakou volbu používat častěji, volba se po naplnění podmínek pravidla objeví v seznamu používaných voleb, a naopak ta, která se vyhodnotí jako méně používaná v poměru k ostatním, se uschová a nebude viditelná.

Druhou možností bylo nastavení dostupných voleb na základě uživatelských rolí. Uživatel z oddělení logistiky nepotřebuje vidět možnosti administrátora,

v některých případech to je dokonce nežádoucí. Aplikace tedy umožní každému uživateli nastavit uživatelskou roli. Nový uživatel zpočátku by měl roli hosta s možností sledovat pouze obsah tabulky, rychlého hledání a změny vzhledu aplikace. Až s pozdějším nastavením jeho role by se mu rozšířily možnosti aplikace. Obě možnosti zredukují možnosti, které uživatel uvidí při používání aplikace, proto se aplikace stane uživatelsky přívětivější. Na základě rozhodnutí zadavatelského týmu bylo rozhodnuto o realizaci druhé možnosti adaptace ovládacích prvků.

K současné aplikaci byla vytvořena uživatelská příručka. Nebyla však příliš využívaná a uživatelé se v případě potřeby dotazovali kolegů nebo se obrátili přímo na vlastníka aplikace nebo odpovědného vývojového pracovníka.
Nová aplikace byla tedy navržena tak, aby na vybraných místech zobrazila nápovědu, která by napověděla, co se od uživatele na daném místě očekává nebo jaké možnosti uživatel má. Nastavené pravidlo následně vyhodnotí, jak často dané místo uživatel navštívil a tedy jak často mohl nápovědu vidět. V případě, že dojde k naplnění podmínky pravidla, bude uživatel považován za zkušeného uživatele a nápověda se mu na daném místě skryje.

4.4 Parametry vývoje nové aplikace

Na základě podnětů a požadavků uživatelů byly podrobně probrány možnosti stávající aplikace. Aplikace byla vytvořena pracovníkem, který již ve firmě nepracuje a nikdo jiný se již ve firmě nezabývá programováním ve Visual Basic.NET. Z tohoto důvodu bylo rozhodnuto, že aplikace bude přeprogramována do podporovaného jazyka C#, který je pro vývoj ve firmě používán častěji.
Současným trendem ve firmě je využívání především webových aplikací, které nevyžadují instalaci na konkrétní klientské zařízení, proto bylo rozhodnuto, že i současná aplikace na sledování blokovaného materiálu bude realizována jako webová.
Aplikace je vytvořena na platformě ASP.NET MVC a její serverová část je rozdělena do jednotlivých kontrolerů podle toho, pro jakou roli jsou určené nebo k čemu slouží (například správa uživatelských rolí bude řešena přes kontroler AdminController).

Aplikační a persistentní vrstva je tvořena samostatnými projekty v rámci jednoho solution.

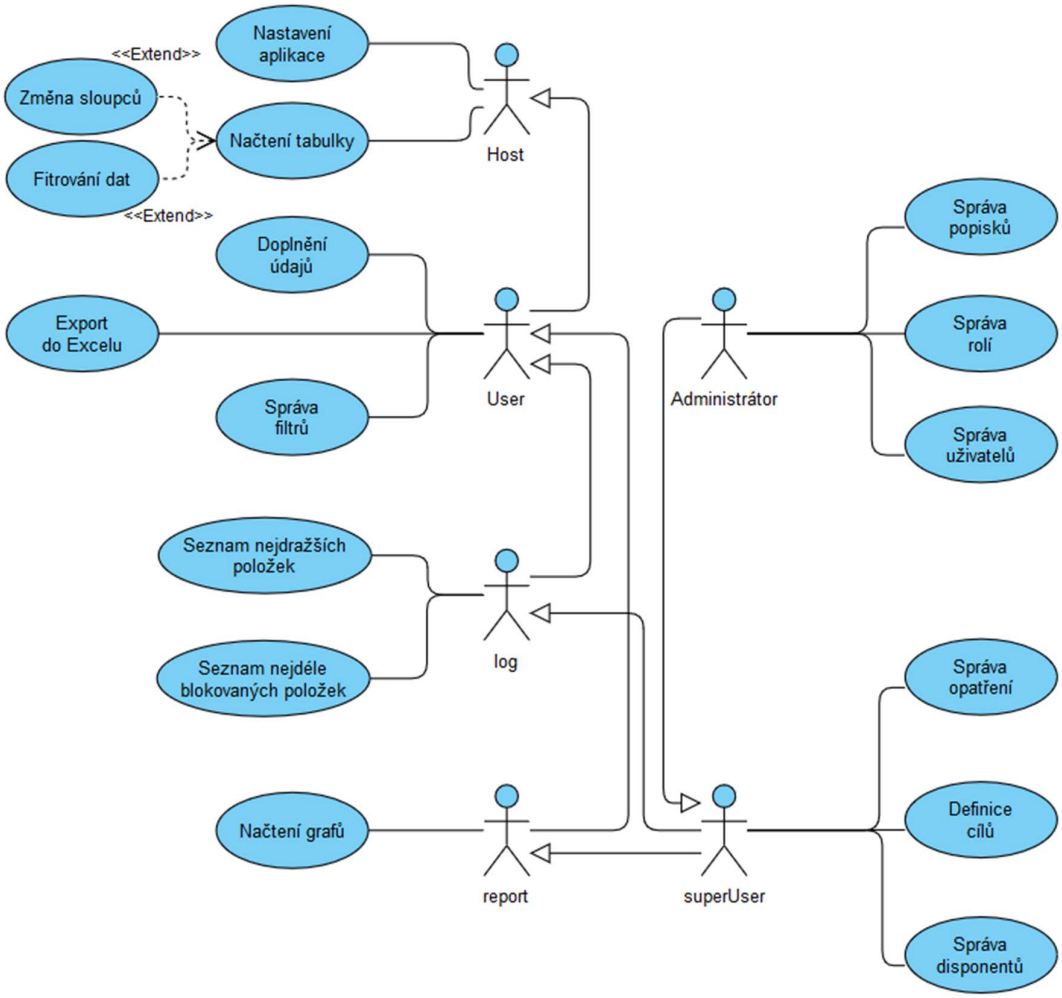
Klientská část bude tvořena v rámci jednotlivých views za využití javascriptové knihovny JQuery a jejích funkcí. Pro komunikaci klientských skriptů s databází budou pomocí AJAX metod volány actions vybraných kontrolerů k tomu určených.

Aplikace poběží na firemním serveru s nainstalovaným IIS (Internet Information Services), aby byl dostupný co nejvíce uživatelům ve firmě. Firemní bezpečnostní politika neumožňuje v současné době připojení k interním webovým aplikacím přes mobilní zařízení, proto se jejich používání prozatím neuvažuje.

Na základě komunikace s centrálním IT oddělením, zabývajícím se správou dat externího systému, se podařilo zajistit automatizaci importu dat bez nutnosti zásahu uživatele, čímž se podařilo odstranit jeden z úkolů administrátora a zvýšení pravidelné aktualizace dat v aplikaci. Jako úložiště bylo vybráno datové úložiště systému MS SQL Server, který je již ve firmě zaveden.

Pro testování byla vybrána skupina deseti uživatelů nejednotného pohlaví a různého pracovního zařazení, aby bylo možné vyzkoušet, zda je aplikace vhodně navržena a zda je možné ji rychle a intuitivně používat. Testovací skupina se skládá z uživatelů, kteří pracují na logistickém oddělení a sledují opatření, která jsou nastavena odpovědnými osobami k blokovaným zásobám na svých odděleních. Další uživatelé jsou odpovědné osoby, které sledují své konkrétní zásoby a zapisují do nich údaje jako opatření k odblokování a odpovědné osoby, které se problémem budou zabývat. Mezi další uživatele patří pracovníci IT oddělení, kteří starší aplikaci nepoužívali, ale mají dobré znalosti ovládání webových aplikací, a asistentky vedoucích pracovníků, kteří nemají velké znalosti z oblasti IT.

Na základě funkčnosti stávající aplikace a požadavky na funkčnost nového systému byly definovány případy užití, které bylo možné rozdělit do šesti samostatných rolí - administrátor, superuser, log, report, user a host.



Obr. 5 Use Case diagram.

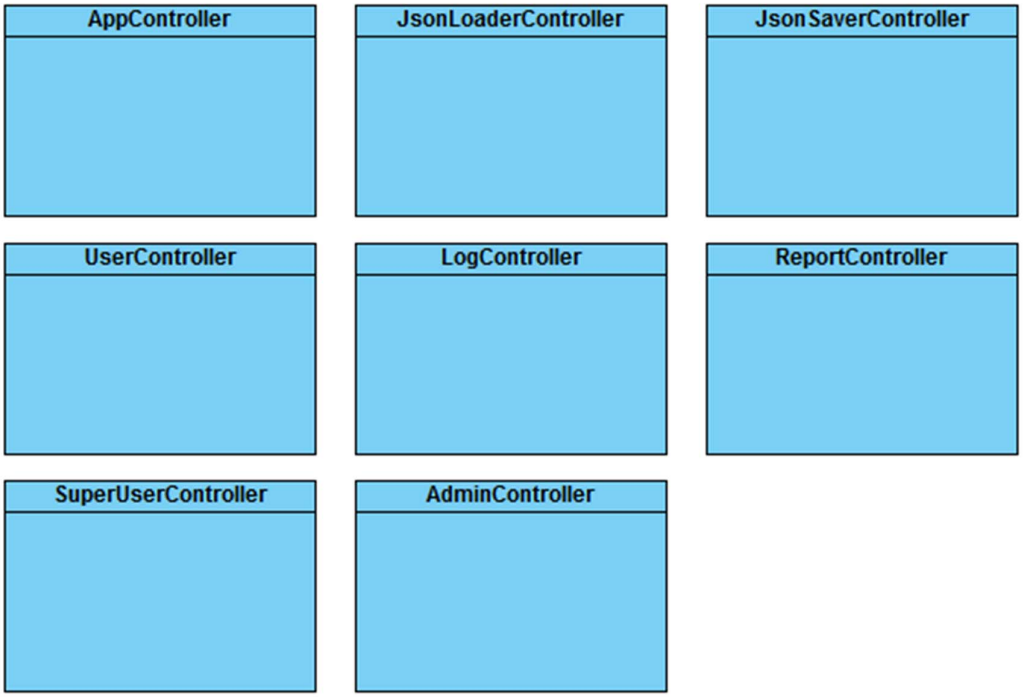
Aplikace bude přístupná každému, kdo bude mít přístup k adrese aplikace. Dokud mu však nebude přidělena žádná role, bude mít pouze defaultně nastavenou roli *host*. Při první návštěvě se uživateli zobrazí možnosti počátečního nastavení aplikace, jako je jazyk, vzhled nebo sloupce, které se mají v tabulce blokovaných materiálů zobrazit. Po nastavení a zavření úvodního okna se vybrané údaje uloží do databáze a při další návštěvě systému se aplikace zobrazí uživateli v prostředí, které

si určil. V tabulce bude mít uživatel s rolí *host* pouze možnost náhledu na tabulku, včetně rychlého filtrování údajů.
Po přidělení role *user* bude mít uživatel možnost doplnit údaje k jednotlivým záznamům, vytvářet si filtry, které bude chtít volat i při dalších návštěvách, nebo exportovat data do souboru xlsx, používaném v tabulkovém procesoru MS Excel.
Pokud bude uživatel z oddělení logistiky a bude mít přidělenou roli *log*, bude mít možnost i sledovat reporty pro sledování množství nebo doby blokování materiálu.
Uživatel s rolí *report* nebude mít stejné možnosti jako uživatel s rolí *log*, jeho možností navíc oproti roli *user* je možnost načítání grafů na základě údajů z tabulky blokovaného materiálu.
Uživatel s rolí *superUser* bude mít možnost využívat všech předchozích případů užití, s rozšířením o správu opatření k uvolnění, které se bude nabízet k vyplnění jako nejčastěji vybrané možnosti. Také bude spravovat tabulku cílů maximální ceny blokovaného materiálu pro jednotlivé produkty a správu disponentů, kteří se budou k jednotlivým materiálům přiřazovat podle údajů v záznamech.
Nejvyšší oprávnění *administrátor* pak bude mít k dispozici ještě správu jednotlivých jazykových mutací, správu rolí a přidělování či odebírání těchto rolí jednotlivým uživatelům.

Spouštěcí aplikace bude obsahovat řídící kontrolery, kterými se budou zachytávat dotazy uživatelů na aplikaci. Kontrolery nebudou obsahovat aplikační logiku, budou fungovat pouze pro zachycení dotazu na aplikaci. Následně vybraný kontroler zavolá třídy aplikační logiky a pro provedení nezbytných operací vrací odpovídající objekt typu View, který zobrazí výslednou obrazovku.
Operace každé role bude zastupovat odpovídající kontroler svými actions. Pro potřeby asynchronní komunikace klientských skriptů s aplikační logikou aplikace budou vytvořeny speciální kontrolery, jejichž actions nebudou vracet objekt typu View, ale objekt typu JsonResult (v případě dotazu na data) nebo pravdivostní hodnotu, zda se požadovaná akce zdařila nebo ne (v případě odesílání dat). Kontroler pro získávání dat je nazván JsonLoader, kontroler pro zpracovávání hodnot je nazván JsonSaver.

Hlavním kontrolerem je AppController, určený pro načtení obrazovky, ve které se zobrazují hlavní data, tedy tabulka se záznamy a ovládací prvky uživatelů. Pro speciální úkoly, pro něž je potřeba speciální objekt typu View (například export dat do excelu nebo editace uživatelských rolí), jsou vytvořené další akce v rámci kontrolerů, k nimž se akce vážou.

Kontrola oprávnění použití dané akce vybraného kontroleru je následně řešena pomocí objektů typu ActionFilter. Objekt zjistí přiřazenou roli uživatele a v případě, že požadovanou roli uživatel má, akce bude provedena, v opačném případě bude uživatel přesunut na úvodní obrazovku.

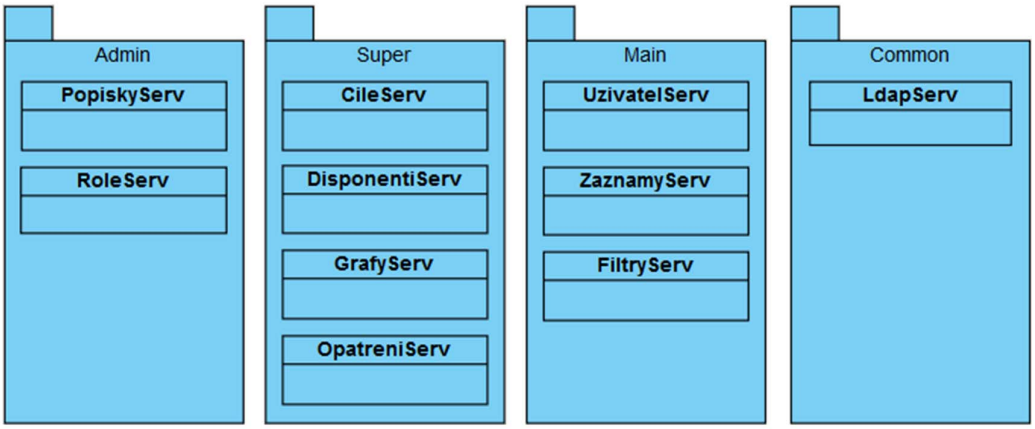


Obr. 6 Class diagram (Kontrolery).

Aplikační vrstva, jejíž třídy budou jednotlivé kontrolery volat, bude oddělena od hlavního projektu do samostatného class projektu. Jednotlivé případy užití jsou rozděleny do servisních tříd podle svého zaměření a třídy jsou sdruženy dle svého zaměření do vybraných složek (jmenných prostorů). Metody servisních tříd není možné volat samostatně z aplikace, jsou volány pouze vybranými kontrolery. V případě řešení potřebných úkonů aplikační logiky jsou tyto řešeny právě

v servisních třídách. Data jsou získávány pomocí tříd datové vrstvy, zpracovány v servisních třídách a výsledek je vracen kontrolerům.

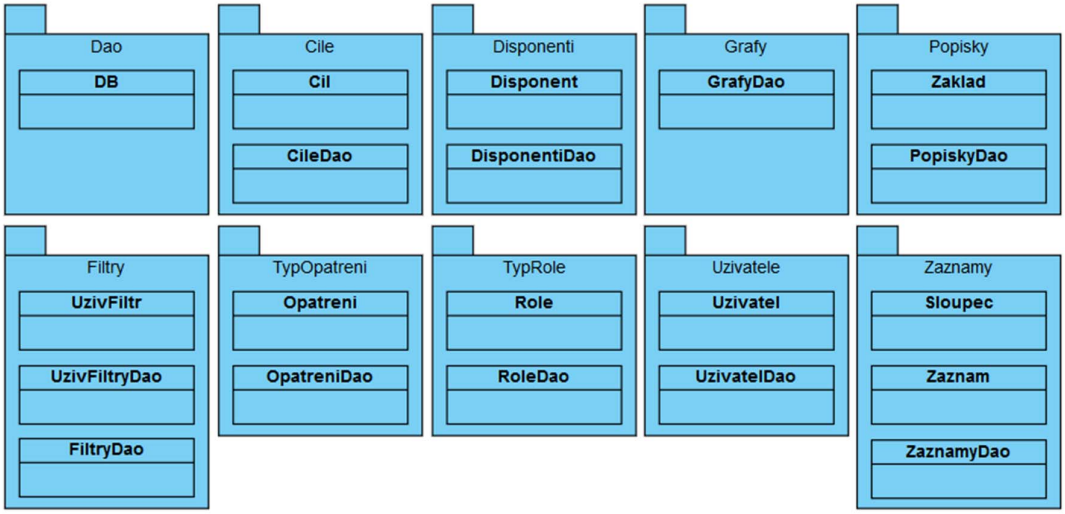
Typickým příkladem je získávání objektu, který má být použit k vygenerování prezentační vrstvy. Objekt View potřebuje pro zobrazení dat a popisků dva modelové objekty, které dodá servisní třída voláním různých obslužných tříd datové vrstvy.



Obr. 7 Class diagram (servisní projekt).

Datová vrstva, propojující aplikační vrstvu s databází, bude rovněž oddělena do samostatného class projektu. Projekt je rozdělen do samostatných složek (jmenných prostorů), které obsahují doménové třídy dle zaměření složky a obslužnou třídu, zodpovědnou za komunikaci s databází. Metody, obsažené v obslužných třídách, jsou zodpovědné za získávání dat z databázových tabulek, aktualizaci hodnot v záznamech nebo odstranění vybraných záznamů.

Výsledkem je vracený seznam hodnot nebo pravdivostní hodnota, zda se volaná metoda provedla správně či nikoliv. V případě výjimky je volána služba, která vygeneruje mail s chybovým hlášením, odesílaný definovanému správci aplikace.



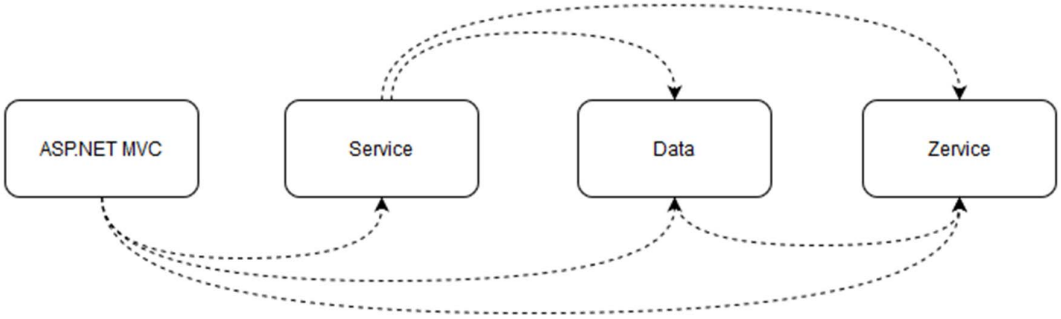
Obr. 8 Class diagram (datový projekt).

Poslední projekt je součástí aplikační vrstvy, která má na starosti především obsluhu zachycených výjimek v průběhu komunikace s databází či generování prezentační vrstvy. Vytvořená třída je využívána pro informování správce aplikace, pokud během provozu aplikace došlo k výjimce, která zabránila správnému chodu aplikace. Může se jednat o chybu, způsobenou některou ze tříd aplikační logiky, nebo o chybu při prezentaci dat. Poslední možností je zaslání chybového hlášení, které si developer sám definuje v průběhu aplikační logiky.

Součástí projektu je i základní třída služeb aplikační logiky, ze které ostatní třídy dědí. Její hlavní metodou je získávání uživatelského jména po spuštění aplikace. Díky této metodě je možné zjistit, který uživatel si aplikaci spustil, a bude možné pak zjistit i jeho definované nastavení aplikace, které si nastaví při prvotním použití aplikace nebo které si následně upraví během používání.

5 Implementace navrženého řešení

Pro implementaci navrženého řešení byl vytvořen nový projekt typu ASP.NET MVC. Do vytvořeného solution byly přidány další tři projekty typu Class Library. Aplikační knihovna Service využívá datové knihovny Data. Všechny projekty pak využívají základní knihovny Zervice. Hlavní projekt pro zpracování dotazů využívá hlavně aplikační knihovny Service, ale potřebuje také referenci na datovou knihovnu Data kvůli zasílaným objektům metodou POST a pro kontrolní filtry.



Obr. 9 Reference jednotlivých projektů.

Součástí vytvořeného projektu jsou balíčky nástrojů a služeb, instalované pomocí balíčkového manažeru, zvaného NuGet a instalovaného jako součást vývojového prostředí Visual Studio. Pro vývoj nové aplikace bylo potřeba doinstalovat některé doplňky, například knihovnu jQuery, preprocesor kaskádových stylů Sass či frameworku na podporu generování seznamu dat ve formátu Json.

Aby mohly jednotlivé projekty mezi sebou komunikovat, byly vytvořeny jednotlivé reference z hlavního projektu na zbývající projekty. Stejně tak byly propojeny i projekty mezi sebou tak, aby nedošlo k cyklické referenci, což by Visual Studio vyhodnotilo jako nepovolenou operaci.

Pro potřeby generování emailových zpráv, posílaných v případě výjimky zpracování metod hlavního nebo class projektů, byl doplněn konfigurační soubor hlavního projektu Web.config o údaje, nezbytné k propojení se SMTP serverem, a o emailovou adresu příjemce.

Struktura hlavního projektu je rozdělena do několika složek podle svého zaměření.

Tabulka 1 Adresářová struktura hlavního projektu.

ActionFilters	Třídy akčních filtrů, které jsou využívány pro ověření, že vybraná akce kontroleru spouští uživatel s odpovídající rolí.
App_Data	Šablony emailových zpráv, posílaných při výjimce aplikace.
App_Start	Konfigurační třídy ASP.NET aplikace.
Content	Soubory kaskádových stylů a preprocesoru Sass.
Controllers	Řídící třídy, zachytávající dotazy uživatele pomocí actions.
Images	Obrázky, používané v aplikaci.
Scripts	Klientské skripty jQuery.
Views	Prezentační vrstva, složená z objektů View a Master Layout.

Příchozí dotaz uživatele je zachycen odpovídajícím kontrolerem. Metoda, implementující volanou action, v případě potřeby využije vybranou třídu aplikační vrstvy class projektu Service a vrací objekt View. Tento objekt je v podstatě webová stránka, uložená ve stromové struktuře hlavní složky View. Ve složce jsou vytvořeny podsložky, pojmenované podle názvu jednotlivých kontrolerů. Pokud v sobě kontroler obsahuje nějakou action metodu, vytvořené view je vygenerováno a uchováno pod složkou, pojmenovanou podle svého kontroleru.

Adresářová struktura složky View obsahuje složku Shared, která obsahuje Html stránky typu Master Layout. Tyto stránky obsahují elementární prvky webového dokumentu a pomocí funkce RenderBody() vkládají volané objekty View. Aplikace ASP.NET MVC umožňuje mít ve složce Shared podle potřeby několik master layoutů a v objektech View pak explicitně říct, který má být při generování pohledu použit. Tím je možné odlišovat například javascriptové knihovny, které se mají použít pro správný chod aplikace, nebo vynechat prvky, které se zobrazovat nemají.

5.1 Tvorba hlavního projektu

Nejdůležitější částí aplikace jsou její kontrolery. Pokud uživatel spustí aplikaci, první kontroler, který by měl být spuštěn, je všeobecná stránka, zobrazující tabulku. Kontroler AppController obsahuje metodu HttpGet, která obsluhuje volání dotazu na aplikaci ve formátu "/App/Load". Metoda je dostupná pro všechny, proto není

třeba provádět kontrolu, zda uživatel může toto volání provést. Metoda reaguje za dotaz vrácením odpovídající stránky ve složce Views.

Webová stránka Load v úvodu volá odpovídají master layout stránku _AppLayout. Stránka je rozdělena na dva hlavní bloky – hlavička a tělo. Hlavička obsahuje název aplikace, kolekci souborů pro definování stylů a kolekci souborů s klientským skriptováním. Tělo stránky obsahuje zobrazovaný obsah, rozdělený do několika bloků, z nichž některé jsou v úvodu skryté a objevují se jen v případě, že jsou volány.

Tabulka 2 Bloky master layout stránky _AppLayout.

Obsah	Blok, zobrazující hlavní obsah. Skládá se navigačního menu pro výběr jazykové mutace, položek dle uživatelské role a uživatelských filtrů.
Loader	Informuje uživatele, že se nahrávají nebo předávají data.
Nová návštěva	Blok nastavení pro nového návštěvníka aplikace.
Nový filtr	Blok pro definici nového filtru.
Vzhled	Blok pro výběr z dostupných vzhledů aplikace.
Sloupce	Blok pro výběr z dostupných sloupců v tabulce aplikace.
Opatření	Blok pro nastavení opatření u záznamů tabulky v aplikaci.

Ve složce Content byla struktura kaskádových stylů rozdělena na *sass*, obsahující definované styly pro načítací stránku a nabízené vzhledy, a dále na *themes*, obsahující předdefinované styly prvků, využívaných knihovnou jQuery, respektive jQueryUI. Složka *sass* obsahuje soubory typu *.scss*, které jsou následně kompilovány do prohlížeči podporovaného formátu *.css*, a to včetně minimalistické verze pro případy snížení přenosové velikosti při generování stránky.

Na začátku zdrojových souborů kaskádových stylů jsou definovány proměnné s barevnými odstíny podle stylu, pro který jsou určeny. Většina definovaných objektů se odkazuje na identifikátor bloku, u kterého budou použity. V těchto objektech jsou pak dále definovány styly pro prvky, zobrazené v daném bloku. V některých případech jsou definovány styly pro samostatné třídy.

Akční metody kontrolerů aplikace ve většině případů odesílají pouze objekty typu View a do webové stránky, která se zobrazuje v master layoutu, nepředávají žádné informace. O obsah webové stránky se starají klientské skripty, uložené ve složce *Scripts*.

Hlavní složkou je složka *_main*, která obsahuje knihovny potřebné pro fungování skriptů jQuery. Jedná se o knihovny jádra frameworku a dále knihovny pro generování ovládacích prvků jQueryUI, které se používají v aplikaci. Protože se využívá i kalendářový objekt, obsahující názvy měsíců a dnů, byly připojeny i soubory, obsahující jazykovou mutaci podporovaných jazyků.

Zbytek struktury složky *Scripts* obsahuje vlastní skripty. Složka *ajax* obsahuje funkce, používané pro získávání a odesílání dat aplikace. Součástí jsou i funkce, reagující na zahájení a ukončení asynchronního přenosu ajax. Složka *app* obsahuje funkce, které jsou spouštěny po zobrazení hlavní stránky. Mezi ně patří nastavení jazykové mutace uživatele, generování nabízených možností dle role uživatele nebo načtení filtrů, které uživatel, v případě, že není pouze hostem aplikace, může používat a má definované ke svému profilu.

Ve složce role jsou definovány funkce, které jsou k dispozici podle role uživatele. Například ve složce *guest* jsou uloženy skripty, dostupné všem uživatelům, tedy skripty pro generování dialogových oken v případě návštěvy nového uživatele a pro potřeby změny vzhledu nebo seznamu sloupců, které chce uživatel vidět.

Ve složce *zaznamy* jsou pak funkce, pracující s tabulkou dat. Jedná se o generování celé tabulky, správa seznamu sloupců, načtení dat, správu filtru nebo funkce pro generování dialogového okna, určeného k zadávání opatření záznamů.

Po načtení stránky se spustí hlavní funkce, která získává popisky aplikace, nastavení uživatele, záznamy tabulky, uživatelské filtry a poslední vyhledávané informace v tabulce. Všechna tato data se získávají pomocí volání ajax metod typu GET, využívajících akčních metod kontroleru JsonLoaderController. Volaná obslužná metoda zavolá odpovídající servisní třídu a získaná data vrací zpět v textovém formátu JSON.

Detekce prvního přístupu do aplikace je prováděna dvěma způsoby. U prvního se předpokládá, že záznam o uživateli je již uveden v odpovídající tabulce, pravidelně

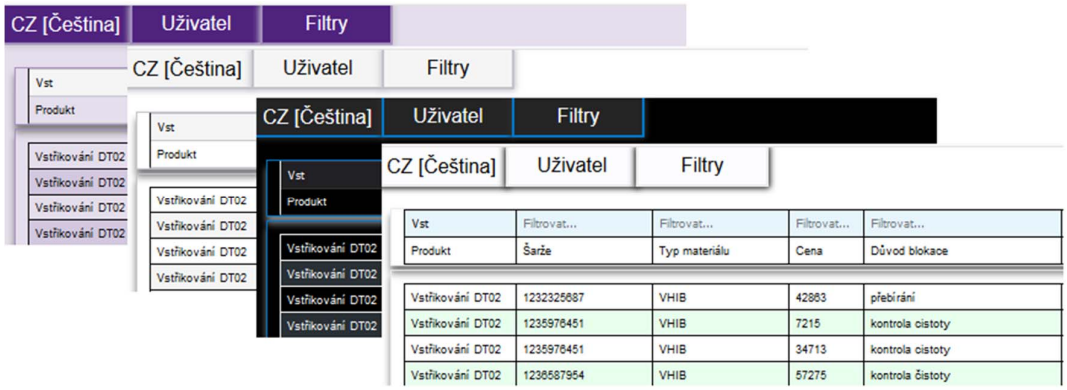
aktualizované na začátku měsíce. Uživatel, který doposud neprovedl vlastní nastavení, má v ve sloupci *novy* pravdivostní hodnotu nastavenou na *true*. Druhý způsob počítá s možností, zpravidla vyskytující se v průběhu měsíce, že při získávání údajů o uživateli datová třída, která se dotazuje do databázové tabulky, neobdrží žádný záznam, což znamená, že uživatel není lokálním zaměstnancem společnosti nebo nastoupil v průběhu měsíce a ještě nebyl zaevidován v aplikaci do tabulky zaměstnanců. V tomto případě se vytváří nový objekt typu *Uzivatel* a předává se s parametrem, oznamujícím nového uživatele systému.

Ve chvíli, kdy hlavní funkce klientského skriptu získá všechna data, místo zobrazení obrazovky s tabulkou nejdříve zobrazí úvodní dialogové okno pro nového uživatele. V něm je uživateli nabídnuto, aby si vybral jazyk, ve kterém chce mít zobrazené názvy sloupců tabulky i další prvky aplikace. Protože uživatel nemusí rozumět jednotlivým slovům na tlačítkách, každé tlačítko obsahuje pouze zkratku jazyka a kliknutím na požadovaný jazyk se všechny texty změní na popisky ve vybraném jazyce. V dalším bloku dialogového okna je k dispozici výběr ze stylů, podporovaných aplikací. Kliknutím na konkrétní styl se uživateli okamžitě změní vzhled dialogového okna a jeho prvků na daný styl, takže uživatel ihned vidí, v čem se jednotlivé styly liší. Na poslední stránce si uživatel může definovat, které sloupce chce mít zobrazeny v tabulce. Pro nového uživatele jsou předvybrány sloupce, které byly definovány v předchozí aplikaci jako nejpoužívanější. Vybrané sloupce je možné označit jako nevybrané a naopak.

Uživateli je k dispozici tlačítko pro zavření dialogového okna, a tak volbu nastavení může kdykoliv přerušit a přejít rovnou do aplikace. Hlavní funkce posbírá všechna doposud zadaná data a po kliknutí na zavření dialogového okna posílá data aplikaci na uložení, aby při další návštěvě mohl mít uživatel aplikaci nastavenou tak, jak si vybral. V případě, že uživatel již v tabulce existuje, pouze se aktualizují data daného záznamu, v opačném případě se vytváří nový záznam do tabulky uživatelů.

Data jsou posílána pomocí ajax metody typu POST, kdy se předává objekt třídy Uzivatel akční metodě kontroleru JsonSaverController.

Jakmile je ukončeno odeslání uživatelského nastavení, aplikace zobrazí hlavní obrazovku, obsahující tabulku a obslužné navigační menu.



Obr. 10 Výběr vzhledu aplikace

Hlavní náplní většiny uživatelů je především práce s daty blokovaných zásob. Pro vytvoření tabulky se skrze ajax metody získává seznam sloupců, určených z nastavení uživatele, jejich popisky v preferovaném jazyce uživatele a seznam záznamů. Součástí získávání těchto údajů je i záznam o posledním vyhledávání uživatele v tabulce. U nového návštěvníka se však záznam o vyhledávání nevrací žádný. Vygenerovaná tabulka je rozdělena na dvě části.

První částí je hlavička tabulky, obsahující jednotlivé názvy. Každý sloupec je definován popiskem a velikostí, rozdílné podle typu sloupce. Nad názvy sloupců je vytvořen řádek pro možnost zadávání hodnot k vyhledávání záznamů podle uživatelsky definovaných hodnot. Na počátku jsou všechny buňky vyhledávacího řádku prázdné, po zadání hodnoty do některé z nich se hodnota uchovává a zobrazuje u daného sloupce.

Druhou částí je seznam řádků s daty. Pokud uživatel neomezí seznam zadáním filtru, jsou vybrány záznamy bez konkrétního omezení a bez seřazení, v opačném případě se načítají pouze data, vyhovující podmínkám vyhledávání. Filtrování je rozděleno na vyhledávání v textu, kdy se vyhledává podle prvních znaků řetězců, a vyhledávání podle čísel nebo data, kdy se hledá stejné nebo větší číslo, respektive stejné nebo novější datum ve vybraných sloupcích. V případě zadání parametrů do více sloupců se pak vyhledávají hodnoty, které vyhovují všem uvedeným parametrům současně.

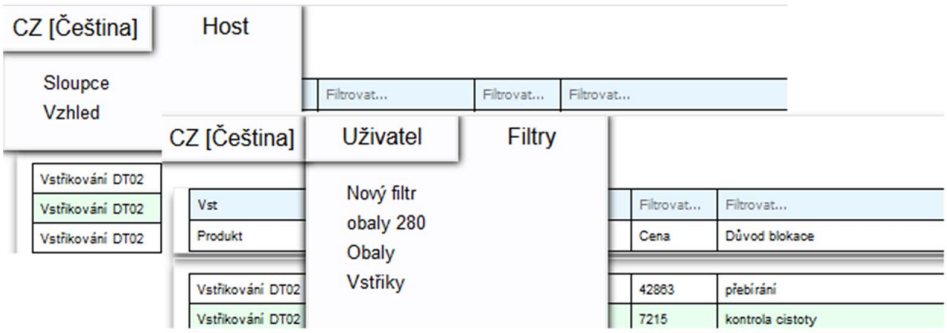
Vst	Filtrovat...	Filtrovat...	50000	Filtrovat...
Produkt	Šarže	Typ materiálu	Cena	Důvod blokace
Vstřikování DT02	1236587954	VHIB	57275	kontrola čistoty
Vstřikování DT02	1244658572	VERP	89622	moc lepidla
Vstřikování DT02	1244658572	VERP	82427	etiketa Indie
Vstřikování DT02	1236587954	VHIB	63352	kontrola čistoty
Vstřikování DT02	1235976451	VHIB	67672	kontrola cistoty
Vstřikování DT02	1244658572	FHMI	51980	DIFERENCE
Vstřikování DT02	1235976451	VHIB	71072	kontrola cistoty
Vstřikování DT02	1275784651	VHIB	90370	kontrola čistota
Vstřikování DT02	1297858974	VHIB	52701	MHD KVĚTEN 2020
Vstřikování DT02	1236587954	VHIB	78812	MHD KVĚTEN 2020
Vstřikování DT02	1235976451	VHIB	89926	kontrola cistoty
Vstřikování DT02	1244658572	VERP	79305	poškozené

Obr. 11 Tabulka blokovaných zásob.

Při získávání dat pro generování obsahu aplikace se zjišťuje nastavení uživatele. Vedle zvoleného jazyka, vzhledu, zobrazovaných sloupců či filtru dat se také kontroluje, jakou má uživatel roli. Na základě zvolené role se poté generuje nabídka pro obsluhu aplikace.

Uživatel, kterému nebyla dosud přiřazena žádná role, je veden jako host aplikace. S touto rolí si může prohlížet tabulku, vyhledávat data pomocí rychlého filtru nebo řadit jednotlivé záznamy podle vybraného sloupce. Aby si mohl aplikaci přizpůsobit svému vkusu, má k dispozici možnosti, které mohl zadat při první návštěvě systému. Může si tedy upravit seznam zobrazených sloupců a podle své potřeby některé přidat a jiné schovat. Také může změnit vzhled aplikace výběrem jedné ze tří možností, které aplikace nabízí.

Kromě výše zmíněných možností může provádět i operace s tabulkou. Jeho možnosti jsou však vzhledem k jeho roli hosta omezené. Záznamy je možné řadit podle hodnot od nejmenší po největší nebo je možné vyhledávat podle hodnot, které jsou zapsány do řádku pro rychlé filtrování dat. Vytváření vlastních filtrů však není povoleno, tato sekce je zobrazena pouze těm uživatelům, kteří mají přiřazenu některou z předdefinovaných rolí, mezi něž host nepatří.



Obr. 12 Ukázka vlastních filtrů u role Host a Uživatel

Pro každého uživatele s rolí jinou než host je mimo jiné možnosti doplněna také možnost definice vlastních filtrů, které je možné poskládat podrobnějším způsobem. Zatím co rychlé vyhledávání umožňuje vyhledávat podle začátku textu a případné kombinace je možné spojit pouze logickým součinem (tedy záznamy musí splňovat všechny podmínky najednou), u vlastních filtrů jsou možnosti rozšířené.

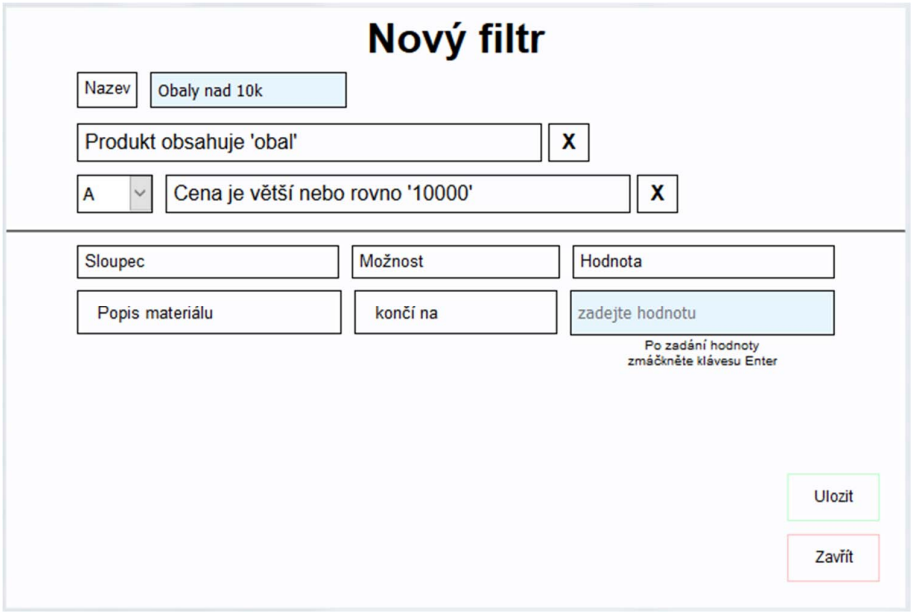
Pro vytváření vlastního filtru je zobrazeno dialogové okno, do kterého se zadává název filtru. Je možno vybrat konkrétní sloupec, který se v tabulce vyskytuje, dále operátor, použitý při filtrování dat, a vybraná hodnota. Je umožněno vyhledávat hodnoty, které se rovnají, jsou větší nebo menší, začínají nebo končí na zadaný text, obsahují nebo neobsahují zadaný text.

Po zadání hodnoty a zmáčknutí klávesy Enter je zaevidována volba filtru a je možné doplnit další podmínku. Jednotlivé podmínky je možné spojit logickým součinem (a zároveň) nebo logickým součtem (nebo).

Při uložení se ukládá vygenerovaný filtr ve dvou verzích. První verze obsahuje výraz, definovaný při vytváření filtru, a zobrazuje se jako text při náhledu v seznamu filtrů po najetí myší na název. Uživatel si tak v případě, že si nepamatuje obsah filtru a z názvu není schopen určit obsah, může nejdříve přečíst, co vybraný filtr obsahuje na podmínky pro vyhledávání dat. Druhá verze je přeložený filtr do jazyka SQL a je při volbě doplněn do vyhledávání záznamů, jakmile je vybrán pro načtení.

V případě výběru některé z možností vlastního filtru je schováno rychle filtrování a nahrazuje ho název vybraného filtru. Výběrem jiného filtru je původní nahrazen a

načítají se data na základě nového filtru. Zrušení filtru opět načítá data bez filtru, ovšem s posledními údaji pro rychlé vyhledávání.



Obr. 13 Tvorba vlastního filtru

Za každý blokovaný materiál je zodpovědný konkrétní uživatel, jehož úkolem je definovat přijatá opatření pro odblokování položky a termín, dokdy bude opatření zavedeno, a uživatele, jehož úkolem bude zajištění, že opatření zavedeno bude.

Aby mohl uživatel použít volbu pro doplnění opatření, musí být alespoň jeden záznam označen, jinak je uživatel upozorněn, že nedošlo k vybrání žádného ze záznamů. Výběr záznamu je realizováno kliknutím na vybraný řádek, opětovné kliknutí naopak řádek odznačí. Předchozí verze aplikace umožnila výběr více řádků najednou. V této aplikaci je to umožněno přes podržené tlačítko myši a posun po tabulce se záznamy. Každý označený řádek je zvýrazněn, aby bylo viditelné, které záznamy jsou vybrány a které se nezahrnují.

Pro zadání opatření je zobrazeno dialogové okno, obsahující pole pro vybrání odpovědné osoby, plánovaného opatření a datum, do kterého se opatření zavede. Pro výběry jsou implementovány komponenty knihovny jQueryUI, a to DatePicker a Autocomplete. Do polí odpovědných osob a plánovaných opatření jsou načteny

možné návrhy hodnot a uživatel při vypisování údajů dostává k dispozici zobrazený seznam možností, které odpovídají jeho výběru. Pole pro zadání data zobrazí kalendář, ve kterém je možné vybrat potřebné datum.

Po vybrání všech potřebných údajů a uložení jsou vybrané údaje uvedeny u všech označených záznamů.

Obr. 14 Zadávání opatření k odblokování

Uživatelé z oddělení logistiky potřebují pro svou práci i sledování těch blokovaných položek, které jsou buď nejdražší nebo jsou blokované nejdéle, a to u každého produktu, který je blokován. V tomto případě si uživatel nevystačí s uživatelským filtrem, ale potřebuje konkrétní reporty.

Pro tyto účely má uživatel s rolí Logistika dodatečné volby v menu pro zobrazení těchto reportů. Ve chvíli, kdy některý z reportů vybere, dojde ke schování voleb, určených pro práci s tabulkou záznamů. Uživatel tedy nemá možnost měnit sloupce nebo vytvořit uživatelský filtr a může použít pouze zobrazení jednoho z reportů a změnit vzhled aplikace. Pro možnost návratu na tabulku blokovaných zásob je přidána možnost pro zobrazení tabulky.

Vygenerované reporty zobrazí několik tabulek podle zaměření reportu. Uživatel si tedy může zobrazit buď tři nejdražší blokované položky u každého produktu nebo pět nejdéle blokovaných položek reportu. V případě, že pro daný produkt není blokováno více jak požadovaný počet, zobrazí se pouze seznam dostupných blokovaných zásob daného produktu.

Produkt	Pořadí	Materiál	Název materiálu	Cena
Čerpadlo X3	1	MAT-8942.3605	Kabeláž	3437
Čerpadlo X3	2	MAT-78239.68885	Pružina	2684
Čerpadlo X3	3	MAT-50777.65039	Palička	2668

Produkt	Pořadí	Materiál	Název materiálu	Cena
Čerpadlo Y1	1	MAT-58522.62125	Nálepky	3019
Čerpadlo Y1	2	MAT-76913.44797	Kabeláž	2814
Čerpadlo Y1	3	MAT-27984.70026	Nálepky	2781

Produkt	Pořadí	Materiál	Název materiálu	Cena
Čerpadlo Y7	1	MAT-69207.5934	Zárovka	315

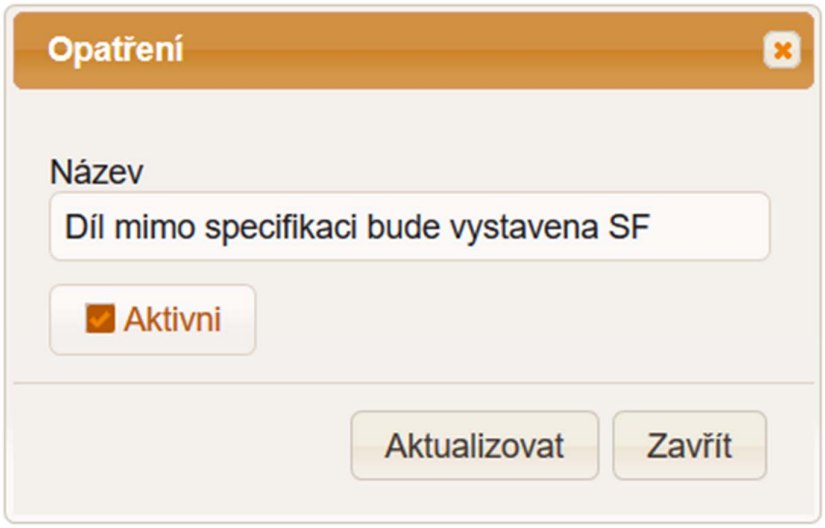
Obr. 15 Report nejdražších položek

Uživatel s oprávněním superuser nebo administrátor mají k dispozici volby pro správu tabulek, používaných pro správných chod aplikace. Mezi tyto volby patří například správa disponentů, seznam opatření nebo i seznam popisků, které se vyskytují v aplikaci.

Ve chvíli, kdy aplikace ověří uživatele a zjistí, že jeho role je superuser nebo administrátor, je doplněn další blok menu, zvaný *Nastavení*, obsahující seznam voleb pro správu podpůrných tabulek. Výběrem některé z voleb dojde k úpravě položek menu podobným způsobem jako v případě výběru reportů u uživatele s rolí logistiky. Jsou tedy schovány volby pro vytváření uživatelských filtrů, změny sloupců nebo nastavení opatření.

Uživateli jsou zobrazeny údaje konkrétní tabulky ve vygenerované tabulce a jsou k dispozici tlačítka pro přidání nového záznamu nebo hromadnou úpravu dat na základě importované tabulky. Úpravu jednotlivých záznamů je pak možné provádět přes dialogové okno, které se zobrazí po poklepání myší na konkrétní řádek tabulky. Dialogové okno je vytvořeno přes komponentu knihovny jQuery UI a naplněno údaji z vybraného záznamu. Uživatel má možnost údaje změnit nebo (de)aktivovat

záznam, což zaručí, že na vybraném místě aplikace se upravovaný údaj zobrazí nebo bude skryt.



Obr. 16 Aktualizace volby ze seznamu opatření

5.2 Podpůrné prvky aplikace

Kromě hlavní části aplikace jsou vytvořeny i další objekty, které se podílejí na chodu aplikace. Jedná se například o samostatné projekty pro aplikační logiku a datovou vrstvu, avšak i v hlavním projektu jsou vytvořeny podpůrné třídy, zajišťující správný chod aplikace.

Uživatel může využívat pouze ty volby, které mu aplikace nabídne. Může však nastat situace, kdy uživateli někdo pošle link na stránku, která je mu skryta. Tím by mohla nastat situace, že by například host mohl navštívit správu uživatelů a nedopatřením nebo záměrně pozměnit údaje či oprávnění některých osob v aplikaci.
Aby bylo zajištěno, že je uživatel oprávněn volat vybranou akční metodu, umožňuje ASP.NET MVC využívat tak zvané akční filtry. Ty je možné volat buď před provedením akční metody nebo až po jejím provedení. V tomto případě je třeba přepsat metodu *onActionExecuting*, která zjistí, jakou má uživatel roli. Pokud nemá nastavenou roli potřebnou pro spuštění dané akční metody, bude přesměrován na základní kontroler a základní akční metodu aplikace. V opačném případě je provedení akční metody povoleno.

Při vývoji aplikace byla vytvořena samostatná složka *ActionFilters* a do ní vloženy třídy, jejichž název odpovídá názvům jednotlivých rolích, používaných v aplikaci. Název tříd je následně v kontrolerech deklarován u vybraných akčních metod pro určení, kde se kontrola provádět má a kde ne.
Přepisovaná metoda onActionExecuting přijímá proměnnou třídy ActionExecutingContext. Pokud se jedná o kontrolu oprávnění a oprávnění není uděleno, přepisuje se vlastnost *Result*, které se předají parametry přesměrování.

Pokud dojde k výjimce v aplikaci, a to jak na straně serverového skriptování, tak na straně klienta, logovaly se informace v jiných aplikacích do logovacích souborů na serveru, kde aplikace byly nasazeny.
V této aplikace je nastaven jiný postup. Aby byl správce aplikace okamžitě upozorněn, kdykoliv dojde při jejím používání k výjimce, v okamžiku problému je vygenerována mailová zpráva, která se zasílá na předem definovanou emailovou adresu s informací, u kterého uživatele se výjimka vyskytla, název servisní či datové třídy včetně názvu metody, a dále chybové hlášení. Uživateli je mezitím zobrazena stránka, informující o dočasné výjimce aplikace, a tak není obtěžován chybovou hláškou přímo.
V rámci hlavního projektu byly vytvořeny textové šablony zpráv, které se zasílají na emailovou adresu. Obslužná třída projektu Zervice následně požadovanou šablonu vybere, doplní do ní potřebné údaje a odešle správci.

5.3 Implementace adaptivních prvků aplikace

Pro implementaci adaptivních prvků aplikace byly přidány do tabulky uživatelů sloupce, určené pro uchování údajů uživatele. Adaptace uživatelského rozhraní je proto realizována uchováváním jednotlivých změn nastavení aplikace na základě volby uživatele. V okamžiku změny jazyka, vzhledu nebo seznamu sloupců je volba uživatele uložena v tabulce a při dalším spuštění aplikace je poslední nastavení načteno.
Pro uchování poslední volby rychlého vyhledávání v tabulce blokovaných zásob je vytvořena samostatná tabulka, která eviduje všechny uživatelské kombinace pro

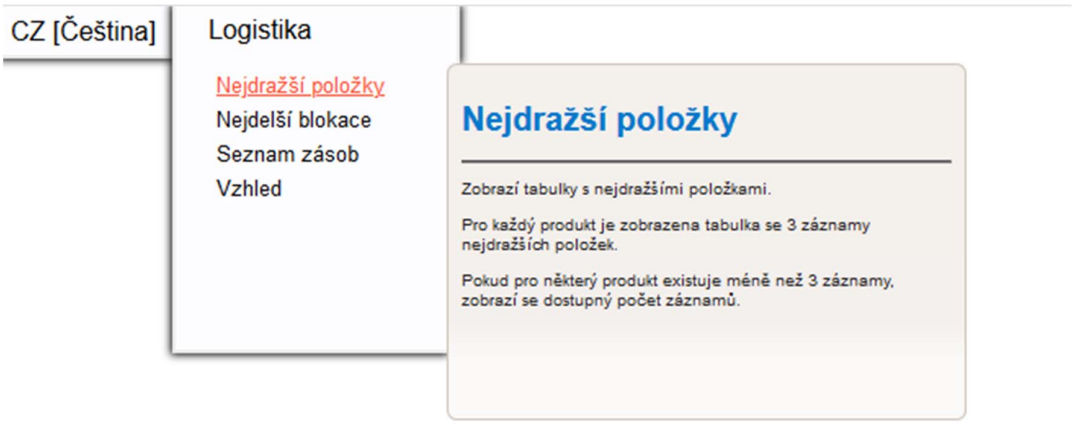
filtraci údajů, které uživatel zadal do aplikace. Při novém spuštění aplikace a zobrazení tabulky je načítána poslední uložená kombinace rychlého vyhledávání a následně jsou zobrazeny pouze ty záznamy, které dané kombinaci vyhovují.

Na databázovém serveru je v pravidelných denních intervalech spouštěna podpůrná aplikace. Jejím úkolem je provést kontrolu nejčastějších vyhledávaných kombinací všech uživatelů. Pokud je nalezena kombinace některého uživatele, která byla použita častěji, než počet definovaný v konfiguračním souboru, podpůrná aplikace vytvoří uživatelský filtr pro daného uživatele. Uživatel tak nemusí následně danou kombinaci ručně zadávat, ale může ji zvolit ze seznamu uživatelských filtrů.

Definované volby, kterými lze aplikaci ovládat, byly rozděleny do skupin podle předpokládaných rolí uživatelů. Při přihlášení aplikace kontroluje uživatelskou roli a na základě této role zobrazuje povolené volby, určené k dané roli. Položky menu jsou tak adaptovány pro práci uživatele a nezobrazují se ty volby, které uživatel nepotřebuje nebo mu nejsou povoleny. Ve chvíli, kdy je uživateli přiřazena jiná role, aplikace upraví seznam voleb.

Přes snahu vytvořit aplikaci velmi intuitivní, pro nového uživatele může být komplikované se v aplikaci orientovat. Pro lepší orientaci je mu možné poskytnout uživatelskou příručku, druhou možností je vkládat nápovědu přímo do aplikace. Například v dialogovém okně je možné zobrazit ikonu otazníku, která po najedí zobrazí nápovědu a ze které uživatel zjistí, k čemu slouží jednotlivé části dialogového okna a jak ovládací prvky používat. Z tohoto důvodu byla implementována komponenta knihovny jQuery UI, zobrazována jako tooltip ve chvíli, kdy uživatel najede myší na vybranou položku, využívající tuto komponentu. Pro nového uživatele je tak komponenta implementována u jednotlivých voleb uživatelského menu nebo u dialogových oken, které běžný uživatel používá, jako již zmíněné okno pro definici uživatelského filtru.

Aplikace uchovává počet zobrazení hlavní stránky aplikace každého uživatele. Ve chvíli, kdy počet zobrazení překročí předdefinovanou hodnotu, je uživatel vyhodnocen jako zkušený a nápověda je schována.



Obr. 17 Nápověda k volbě uživatelského menu

5.4 Testování vytvořené aplikace

Po vytvoření aplikace bylo potřeba ověřit, zda nová aplikace naplňuje cíl práce a očekávání uživatelů. Pro testovací účely byli vybráni uživatelé, kteří mají zkušenosti s předchozí verzí aplikace a mohou tedy posoudit, zda implementované změny jsou přínosem pro práci s blokovanou zásobou, dále uživatelé, kteří se s aplikací nesetkali, ale mají zkušenosti s jinými tabulkovými aplikacemi, a také byli vybráni i uživatelé, kteří nemají s podobnými aplikacemi zkušenosti.

Testující uživatelé byli vybráni i podle rozličného profesního zařazení ve společnosti, různého věku a také pohlaví. Různé zkušenosti s aplikací a podobnými aplikacemi měly prověřit, jak se dokáže v prostředí orientovat zkušený člověk nebo osoba naprosto neznalá procesu blokovaných zásob.

Celkem bylo vybráno 10 testujících uživatelů. Každý z nich měl za úkol splnit 10 úkolů a byl sledován při orientaci v aplikaci, jak rychle je schopen úkoly splnit bez předchozích rad nebo následně s pomocí.

Tabulka 3 Seznam testovacích osob

Tester	Pohlaví	Věk	Pracovní zařazení	Zkušenosti s původní aplikací
1	Žena	30 – 39	Asistentka	Velké
2	Muž	40 – 49	Technolog	Velké
3	Muž	60 – 69	Vedoucí	Žádné
4	Žena	40 – 49	IT partnerka	Žádné
5	Muž	40 – 49	Informatik	Žádné
6	Muž	30 – 39	Informatik	Nízké
7	Žena	20 – 29	Logistka	Velké
8	Muž	30 – 39	Programátor	Žádné
9	Muž	20 – 29	Technolog	Střední
10	Žena	30 – 39	Logistka	Velké

Testujícím uživatelům byly poskytnuty základní informace o účelu aplikace a co obsahují data. Následně se aplikace uživatelům spustila a každý uživatel dostal úkoly, které měl vykonat. Při plnění těchto úkolů byli uživatelé sledování a zaznamenávala se rychlost splnění úkolů. Současně byly vyhodnocovány jejich pohyby po aplikaci, což vedlo k námětům pro zlepšení aplikace.

Pro testovací účely byly schovány všechny nápovědy. Cílem bylo zjistit, jaké jsou rozdíly v pochopení vzhledu aplikace a názvy jednotlivých voleb uživatele. Pro pracovníka, znalého problematiku blokovaných zásob a používajícího předchozí aplikaci by měla být aplikace jednodušší na pochopení, než pro uživatele, který aplikaci nikdy nepoužíval a s tabulkou podobného typu pracoval poprvé.

Další skupinou byli informatici, kteří pracují s podobnými typy aplikací a někteří se dostali do kontaktu s předchozí aplikací při instalaci či řešení komplikací, spojených s fungováním aplikace. Důležitá byla i zpětná vazba od programátora, který aplikace podobného typu vytváří v rámci společnosti.

Tabulka 4 Seznam požadovaných úkolů

Č. úkolu	Požadovaný úkol
1	Proveďte úvodní nastavení svého účtu (volba jazyka, vzhledu, sloupců)
2	Po zavření okna pro nové uživatele změňte vzhled na jiný
3	Přidejte nebo uberte sloupec tabulky
4	Vyhledejte všechny produkty s názvem „čerpadlo“ a s důvodem blokace, obsahujícím slovo „poškození“
5	Z vyhledaného seznamu čerpadel nalezněte nejdražší záznamy
6	Pro první tři nejlevnější záznamy zadejte opatření a odpovědnou osobu
7	Vymažte hodnoty rychlého vyhledávání, aby se zobrazily všechny záznamy
8	Vytvořte uživatelský filtr pro vyhledání všech obalu s typem materiálu „HALB“
9	Zobrazte opět všechna data v tabulce
10	Exportujte data do tabulky v MS Excel

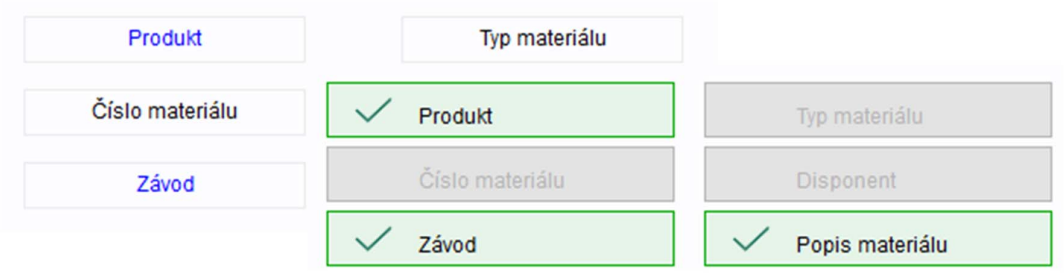
Po provedení všech úkolů byly testující uživatele vyzvání k tomu, aby vyplnili jednoduchý formulář s dotazy na orientaci v aplikaci, zda provádění požadovaných úkolů bylo snadné, zda pochopili, k čemu sloužily jednotlivé volby v menu aplikace, zda hodnotí vzhled aplikace pozitivně a zda kladně hodnotí celkový dojem z používání aplikace. Také dostali k dispozici pole pro vyplnění, co by ocenili v dané aplikaci navíc, co by navrhovali změnit a možnost napsat další náměty nebo poznámky.

6 Diskuze výsledků včetně přínosů a omezení

Aplikace byla testována tak, aby zpětná vazba byla získána nejdříve od uživatelů, kteří se s předchozí aplikací vůbec nesetkali a kteří se správou blokovaných zásob nemají žádné zkušenosti. Následně se do testování zapojili uživatelé se zkušenostmi s aplikacemi podobného typu nebo částečnou zkušeností s původní aplikací. Na závěr se zapojili uživatelé zběhlí v problematice a aktivně používající předchozí aplikaci. Na základě sledování uživatelů při práci s aplikací a jejich zpětné vazby byly vybrány body, které objevily ve výraznějším počtu u uživatelů než ostatní případy.

6.1 Vybrané prvky aplikace

Při výběru zobrazovaných sloupců tabulky více jak polovina uživatelů byla zmatená při určení, jaké sloupce jsou vybrány a jaké ne. Všechny sloupce jsou zobrazeny najednou a pouze zvýraznění pozadím nebo barvou textu jednoznačně neurčuje, zda se jedná o vybraný sloupec nebo naopak zda daný sloupec v tabulce zobrazen nebude, zvláště ve chvíli, kdy téměř polovina sloupců je předvybraných. Po konzultaci byl vznesen požadavek na úpravu zvýraznění sloupců tak, aby bylo jednoznačně rozeznatelné, který sloupec vybrán byl a který ne. Na základě požadavku byla následně přepracovaná volba pro výběr sloupců, aby bylo vizuálně lépe rozeznatelné, které sloupce vybrány jsou a které naopak ne.



Obr. 18 Úprava vzhledu vybraných sloupců tabulky

Podobná uživatelská zkušenost se objevila u rychlého vyhledávání v tabulce záznamů. Více jak polovina testujících uživatelů vykazovala pomalejší plnění úkolu na smazání vybraných hodnot, kterými filtrovali záznamy tabulky. Nepoužívané vyhledávací pole nad názvy sloupců obsahovaly šedivý text, oznamující, že je možné

zapsat údaj, avšak zadaná hodnota byla zvýrazněna pouze černým textem, což vybrané uživatele nutilo procházet sloupec po sloupci a sledovat, zda pole pro zadání hodnoty obsahuje text či nikoliv. Požadavek na zvýraznění vyplněných hodnot v rychlém vyhledávání se také objevil u několika zpětných vazeb testujících uživatelů. Na základě tohoto požadavku proto bylo zvýrazněno každé pole, obsahující hodnotu pro filtrování dat v tabulce blokovaných zásob.

Při plnění úkolu na smazání hodnot nad hlavičkou tabulky, používaných pro filtrování záznamů, se většina s testujících uživatelů shodla na tom, že postupné umazávání vyhledávaných hodnot se může hodit, pokud uživatel zkoumá výsledky postupně. Uživatelsky nepřívětivá je však nutnost ruční vymazání všech hodnot, pokud by uživatel chtěl zobrazit všechny údaje. Z této zkušenosti vyvstal požadavek na vytvoření tlačítka, pomocí kterého by uživatel vymazal veškeré hodnoty, zadané během rychlého vyhledávání, bez nutnosti mazat je jednotlivě. Na základě požadavku tak bylo přidáno tlačítko, které tuto funkcionalitu realizovalo.
Vytváření uživatelských filtrů se zdálo být nejkomplikovanějším úkolem. Někteří uživatelé přehlíželi nápovědu pro zadávání hodnot filtru, někteří vytkli stejný vzhled pro nadpisy polí i pro pole samotná. Na základě této zkušenosti byla hlavička barevně odlišena. Uživatelům následně byla představena i verze s nápovědou, která se zobrazuje při prvotních návštěvách aplikace.
Více jak třetina testujících uživatelů vyjádřila také očekávání, že po vytvoření nového uživatelského filtru a zavření dialogového okna bude uživatelský filtr ihned použit. Tato možnost prozatím nebyla implementována a je pouze vedena v patrnosti a bod bude vznesen během vyhodnocení dalšího testovacího kola.

6.2 Všeobecná zpětná vazba

Závěrečné hodnocení uživatelů mělo odpovědět na otázky k orientaci v aplikaci, srozumitelnost jednotlivých voleb v aplikaci, a hodnocení aplikace ve srovnání s předchozí aplikací.
Ze získané zpětné vazby vyplynulo, že většina uživatelů se v aplikaci spíše dobře orientovala, provádění úkolů bylo snadné, hodnotila kladně nový vzhled a celkově hodnotila práci v aplikaci pozitivně. Nejlepší hodnocení byla poskytnuta uživateli,

kteří měli velkou zkušenost s předchozí aplikací, kteří především ocenili rychlost aplikace a jednoduchost uživatelského prostředí. Mezi návrhy změn převládaly výše zmíněné návrhy, týkající se především zvýraznění výběrů nebo úprav uživatelských filtrů. Někteří uživatele si všimli, že přes uchované hodnoty v rychlém vyhledávání se neuchovává seřazení podle velikosti hodnoty ve sloupci. Mezi jednotky návrhů lze zařadit také návrh na větší velikost fontu v tabulce nebo možnost rozšíření rychlého vyhledávání po vzoru v aplikaci MS Excel, tedy zobrazovat seznam zaškrtávacích polí se všemi hodnotami v daném sloupci, aby si uživatel mohl vybrat konkrétní hodnotu, respektive hodnoty.

Ve zpětné vazbě se také objevilo několik protichůdných hodnocení, poukazující na různé preference vzhledu aplikace. Například jednotlivé volby uživatele, schované pod nabídkou role, hodnotili někteří uživatelé pozitivně. Přesto se objevili jiní, preferující spíše zobrazení lišty se zobrazenými možnostmi najednou, aby nebyly schovány pod hlavní položkou nabídky.

Možnosti adaptace uživatelského rozhraní tak bude dále diskutováno na základě zpětné vazby z dalších testování .

7 Závěr

Cílem práce bylo prozkoumat existující řešení, věnující se adaptaci uživatelského rozhraní, na základě získaných poznatků navrhnout a vytvořit aplikaci pro zvýšení uživatelského komfortu při práci s aplikací a následně vytvořenou aplikaci podrobit testování a vyhodnotit zpětnou vazbu od testujících uživatelů.

Nejdříve byla provedena analýza druhů uživatelského rozhraní a jeho návrhu a tvorby. Dále byly analyzovány možnosti adaptace uživatelského rozhraní podle vybraného přístupu a formy adaptace v existujících řešeních.

Dále byl popsán návrh firemní aplikace s prvky adaptace uživatelského rozhraní a seznam technologií, použitých při vývoji. V implementační části byly rozebrány zásadní prvky hlavního projektu a podpůrné objekty pro správu dat a kontrolu oprávnění dle uživatelských rolí. Závěr implementační části byl věnován adaptivním prvkům aplikace a návrh testování vytvořeného řešení.

V poslední části práce byly porovnány výsledky testování a uvedeny příklady nalezených bodů ze zpětné vazby, které hodnotily výsledné řešení adaptace uživatelského rozhraní a navržena opatření pro zlepšení a další vývoj aplikace.

Na základě zpětné vazby od uživatelů lze usuzovat, že se podařilo zvýšit uživatelský komfort a s adaptací uživatelského rozhraní se uživatelé lépe orientovali v aplikaci při jejím používání. Cíle práce tedy bylo dosaženo.

8 Seznam použité literatury

[1] *Multics* [online]. [vid. 2020-02-03]. Dostupné z: https://www.multicians.org/index.html

[2] *The UNIX® Standard | The Open Group* [online]. [vid. 2020-02-03]. Dostupné z: https://www.opengroup.org/membership/forums/platform/unix

[3] *The History of Development of Norton Commander* [online]. [vid. 2020-02-03]. Dostupné z: http://www.softpanorama.org/OFM/Paradigm/Ch03/norton_commander.shtml

[4] *QuickBASIC - Wikipedia* [online]. [vid. 2020-02-04]. Dostupné z: https://en.wikipedia.org/wiki/QuickBASIC

[5] DIMYADI, Johannes a Robert AMOR. REGULATORY KNOWLEDGE REPRESENTATION FOR AUTOMATED COMPLIANCE AUDIT OF BIM-BASED MODELS. In: [online]. 2013. Dostupné z: doi:10.13140/2.1.1774.6886

[6] *What is Windows, Icons, Menus And Pointing Device (WIMP)? - Definition from Techopedia* [online]. [vid. 2020-02-04]. Dostupné z: https://www.techopedia.com/definition/4687/windows-icons-menus-and-pointing-device-wimp

[7] *Natural user interface- next mainstream product user interface - IEEE Conference Publication* [online]. [vid. 2020-02-04]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/5681374

[8] *3D Touch - User Interaction - iOS - Human Interface Guidelines - Apple Developer* [online]. [vid. 2020-02-04]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/ios/user-interaction/3d-touch/

[9] Siri. *Apple* [online]. [vid. 2020-02-04]. Dostupné z: https://www.apple.com/siri/

[10] *Microsoft HoloLens* [online]. 2020 [vid. 2020-11-05]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Microsoft_HoloLens&oldid=18229800

[11] *What is Human-Computer Interaction (HCI)? | Interaction Design Foundation* [online]. [vid. 2020-02-04]. Dostupné z: https://www.interaction-design.org/literature/topics/human-computer-interaction

[12] FOJTŮ, Andrea a Lenka NĚMEČKOVÁ. *HCI v kontextu informační vědy: postup návrhu uživatelských rozhraní* [online]. B.m.: Praha. 2011. Dostupné z: https://sites.ff.cuni.cz/uisk/wp-content/uploads/sites/62/2016/01/HCI-v-kontextu-informa%C4%8Dn%C3%AD-v%C4%9Bdy-postup-n%C3%A1vrhu-u%C5%BEivatelsk%C3%BDch-rozhran%C3%AD_Fojt%C5%AF.pdf

[13] *User Interface Design Basics* [online]. 21. květen 2014 [vid. 2020-02-04]. Dostupné z: /what-and-why/user-interface-design.html

[14] *UCD vs UX: What's the difference?* [online]. [vid. 2020-02-05]. Dostupné z: https://www.justinmind.com/blog/ucd-vs-ux-whats-the-difference/

[15] *User-Centered Design Basics | Usability.gov* [online]. 3. duben 2017 [vid. 2020-02-06]. Dostupné z: /what-and-why/user-centered-design.html

[16] INFO@LUNDEGAARD.EU, Lundegaard-e-business solutions provider, www lundegaard eu. Human-centered design. *Webová integrace* [online]. 31. červenec 2018 [vid. 2020-02-05]. Dostupné z: http://www.web-integration.info/cs/blog/human-centered-design/

[17] *User Experience Design* [online]. [vid. 2020-02-12]. Dostupné z: http://semanticstudios.com/user_experience_design/

[18] *Optimizing the UX honeycomb - UX Collective* [online]. [vid. 2020-02-12]. Dostupné z: https://uxdesign.cc/optimizing-the-ux-honeycomb-1d10cfb38097

[19] *What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care?* [online]. [vid. 2020-02-12]. Dostupné z: https://medium.com/@space.alpaca/so-what-exactly-is-the-difference-between-fixed-fluid-adaptive-and-responsive-layouts-and-why-3773272d8481

[20] *Responsivní či adaptivní design? | DesignDev - Děláme internet lepší* [online]. [vid. 2020-02-12]. Dostupné z: https://designdev.cz/responsivni-ci-adaptivni-design

[21] SOEGAARD, Mads. Adaptive vs. Responsive Design. *The Interaction Design Foundation* [online]. [vid. 2020-07-28]. Dostupné z: https://www.interaction-design.org/literature/article/adaptive-vs-responsive-design

[22] ULAHANNAN, Arun, Paul JENNINGS, Luis OLIVEIRA a Stewart BIRRELL. Designing an Adaptive Interface: Using Eye Tracking to Classify How Information Usage Changes Over Time in Partially Automated Vehicles. *Ieee Access* [online]. 2020, **8**, 16865–16875. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2966928

[23] LÖCKEN, Andreas, Klas IHME a Anirudh UNNI. Towards Designing Affect-Aware Systems for Mitigating the Effects of In-Vehicle Frustration. In: *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications Adjunct* [online]. New York, NY, USA: Association for Computing Machinery, 2017, s. 88–93. AutomotiveUI '17. ISBN 978-1-4503-5151-5. Dostupné z: doi:10.1145/3131726.3131744

[24] PÉREZ, J. Eduardo, Xabier VALENCIA, Myriam ARRUE a Julio ABASCAL. Evaluation of two virtual cursors for assisting web access to people with motor impairments. *International Journal of Human-Computer Studies* [online]. 2019,

132, 81–98 [vid. 2020-07-13]. ISSN 1071-5819. Dostupné z: doi:10.1016/j.ijhcs.2019.08.001

[25] NAYYAR, Aanand, Utkarsh DWIVEDI, Karan AHUJA, Nitendra RAJPUT, Seema NAGAR a Kuntal DEY. *OptiDwell: Intelligent Adjustment of Dwell Click Time* [online]. 2017. Dostupné z: doi:10.1145/3025171.3025202

[26] MILECKI, Andrzej a Roman REGULSKI. Washing Machine Controller with a New Programming Method. *Acta Mechanica et Automatica* [online]. 2017, **11**, 328–332. Dostupné z: doi:10.1515/ama-2017-0051

[27] CHAHUARA, Pedro, François PORTET a Michel VACHER. Context-Aware Decision Making Under Uncertainty for Voice-based Control of Smart Home. *Expert Systems with Applications* [online]. 2017, **75**. Dostupné z: doi:10.1016/j.eswa.2017.01.014

[28] AKIKI, Pierre A., Arosha K. BANDARA a Yijun YU. Engineering Adaptive Model-Driven User Interfaces. *Ieee Transactions on Software Engineering* [online]. 2016, **42**(12), 1118–1147. ISSN 0098-5589. Dostupné z: doi:10.1109/TSE.2016.2553035

[29] YIGITBAS, Enes, Stefan SAUER a Gregor ENGELS. Adapt-UI: an IDE supporting model-driven development of self-adaptive UIs. In: [online]. 2017, s. 99–104. Dostupné z: doi:10.1145/3102113.3102144

[30] YIGITBAS, Enes, Klementina JOSIFOVSKA, Ivan JOVANOVIKJ, Ferhat KALINCI, Anthony ANJORIN a Gregor ENGELS. *Component-Based Development of Adaptive User Interfaces* [online]. New York: Assoc Computing Machinery, 2019. ISBN 978-1-4503-6745-5. Dostupné z: doi:10.1145/3319499.3328229

[31] GOUR, Rinu. Top 5 BI Tools that You must use for Data Visualization. *Medium* [online]. 23. květen 2019 [vid. 2020-07-14]. Dostupné z: https://towardsdatascience.com/top-5-bi-tools-that-you-must-use-for-data-visualization-7ccc2a852bd3

[32] HUSSAIN, Jamil, Anees Ul HASSAN, Hafiz BILAL, Rahman ALI, Muhammad AFZAL, Shujaat HUSSAIN, Jae BANG, Oresti BANOS a Sungyoung LEE. Model-based adaptive user interface based on context and user experience evaluation. *Journal on Multimodal User Interfaces* [online]. 2018, **12**. Dostupné z: doi:10.1007/s12193-018-0258-2

[33] RAHEEL, Saeed. Improving the User Experience using an Intelligent Adaptive User Interface in Mobile Applications. In: [online]. 2016. Dostupné z: doi:10.1109/IMCET.2016.7777428

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor: **Bc. Karel Kobliha**

Studium: I1800321

Studijní program: N1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název diplomové práce: **Možnosti zvýšení uživatelského komfortu adaptací uživatelského rozhraní aplikací**

Název diplomové práce AJ: The possibilities of increasing user comfort by adaptation of applications UI

Cíl, metody, literatura, předpoklady:

Cíl: Seznámit se s problematikou adaptace uživatelského rozhraní a navrhnout a vytvořit řešení pro existující firemní aplikaci.

1. Teoretický úvod do problematiky

2. Cíle práce

3. Rešerše existujících řešení a literatury

4. Teoretický návrh řešení

5. Implementace navrženého řešení

6. Diskuze výsledků včetně přínosů a omezení

7. Závěr

příloha - extended abstrakt v rozsahu 6 stran včetně schémat, grafů a tabulek, jako sumarizace práce

dle dohody s vedoucím

Garantující pracoviště: Katedra informačních technologií, Fakulta informatiky a managementu

Vedoucí práce: prof. Ing. Ondřej Krejcar, Ph.D.

Oponent: Ing. Ondřej Grycz

Datum zadání závěrečné práce: 21.10.2018