# Aplikace: „Sleduj svého poslance“

## A. Cíle aplikace

Zodpovídá to běžné situaci v životě, kdy je povšechně dobré mít definované cíly, nicméně důležité je jen „rámcově“ konat v souladu s nimi a jako kontraproduktivní se jeví námaha spojená s rozlišením, který konkrétní cíl se tím, kterým činem plní. Ono se to vzhledem na hierarchickou propojenost cílů říci přesně ani nedá. Protože cíl v našm modelu není ani tak bod jako směr.

Jelikož se bude jednat o uživatelskou aplikaci, je výhodnější méně striktní filozofie, a v případě, když se něco nesvedu bude lepší, aby aplikace běžela dál.

Považuji za důležité si stanovit na začátku cíle a mít je počas celého projektu na mysli. Jak se mi ten který cíl podaří naplnit, nechávám k posouzení jiným.

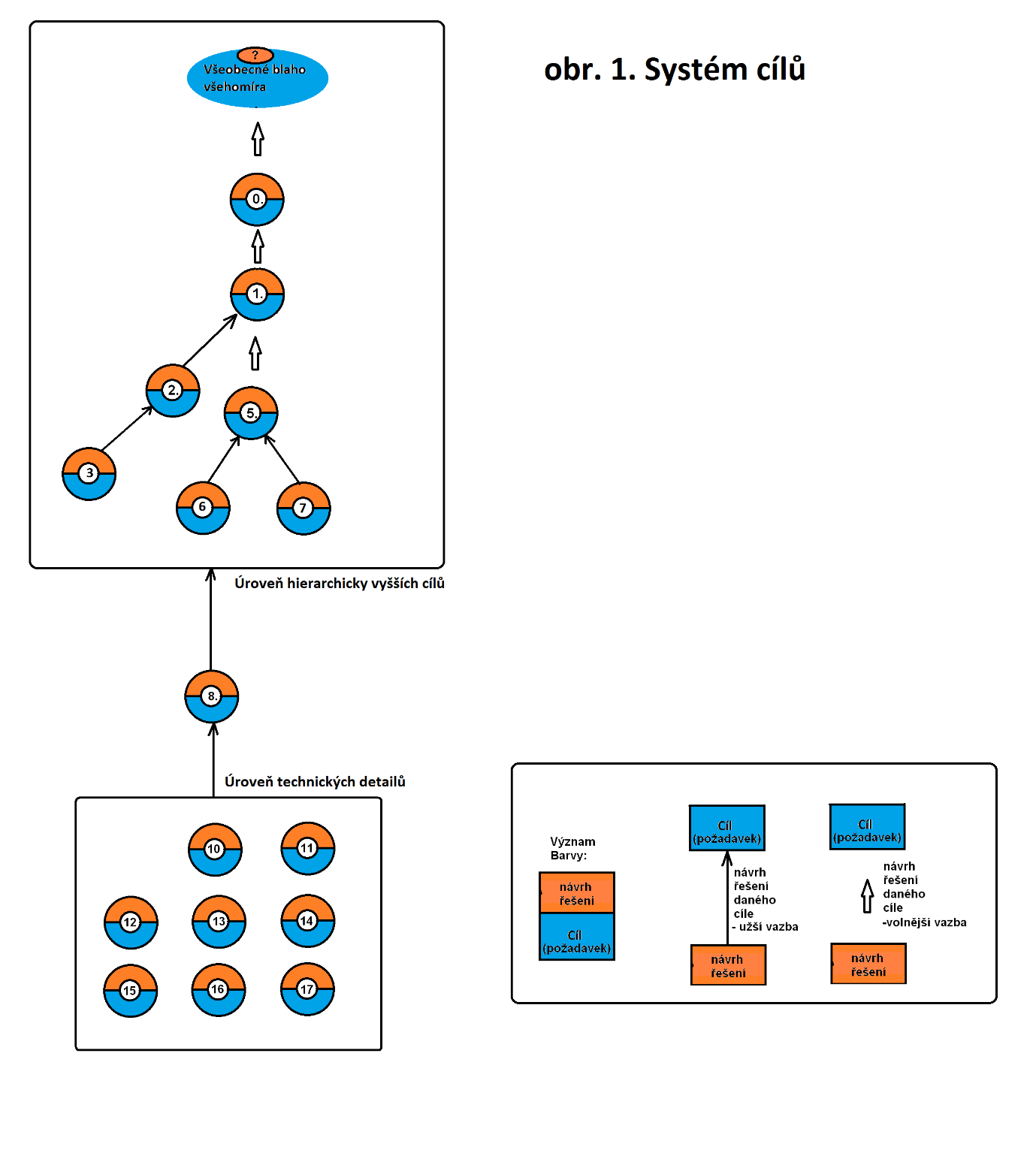
Cílů může být tolik, kolik je úhlů pohledů. Spíš než o cílech jednotlivě bude lépe, když budeme mluvit o ***systému cílů***, který má svou vnitřní strukturu a hierarchii. Následující očíslované body se náš *systém cílů* pokusí načrtnout, nicméně z povahy věci plyne, že tento nebude úplně vyčerpán.

**K ostrému dělení na požadavek / návrh řešení.** Při pozorném pohledu zjišťujeme, že neexistuje ostrá hranice mezi cílem (účelem, požadavkem) a prostředkem k dosažení cíle (návrhem řešení požadavku), ale že se spíš jedná o 2 různé aspekty té samé „věci“. Tedy jestli je nějaká věc cílem, nebo prostředkem, záleží jen na úhlu pohledu. Prostředky na dosáhnutí cílů vyšší úrovně jsou zároveň cílem (protože definují CO se má udělat), který dosáhneme zase jinými prostředky nižší hierarchické úrovně. Tedy již z podstaty věci ostré absolutní oddělení nemůže být realizováno. Tedy není možno od požadavku žádat, aby byl jenom požadavkem, protože je zároveň i řešením něčeho jiného. To samé platí i o řešení. Oddělení je možné jen pokud se týče relativního pohledu (tj. tenhle relativní cíl má takový-hle návrh řešení). Stejná dichotomie (*cíl / prostředek k jeho dosažení*) platí i pro dvojici Požadavek / Návrh řešení požadavku, protože to jsou jen jiné názvy toho samého.

Graficky je ***systém cílů*** našeho sytému je znázorněn na obrázku č.1. Šipky na obrázku třeba chápat víceméně symbolicky, protože cíle se ve skutečnosti vzájemně překrývají a naopak, dané řešení může sloužit k dosažení mnoha cílů. Nicméně je možné si všimnout, že tyto cíle/ řešení je možné smysluplně uspořádat do určitých, řekněme sfér. Jelikož zaměření této práce je spíše praktické, bude lépe, když obrátíme pozornost na nižší sféry cílů / prostředků a to konkrétně týkající se implementace, nicméně pro úplnost, abychom si lépe uvědomili z čeho vycházíme, uvedeme i cíle hierarchicky vyšší.

Nicméně parametry se nevztahují k oběma zmíněným aspektům, např. ***priorita*** se vztahuje pouze k aspektu cíle. Bude-li tedy tato uvedena, bude se tedy vztahovat vždy jen k němu.

Co se týče formálního zápisu, hierarchicky vyšší cíle budu jenom vyjmenovávat, zatím co ty elementární – hierarchicky nejnižší, budu uvádět ve formě tabulky, jak se to běžně dělá.



## Cíle-řešení:

(0). Aplikace podpoří demokratické principy v společnosti.

(1) A. Aplikace má zvýšit transparentnost konání vládních činitelů ve společnosti.

(2) B. Aplikace má být objektivní.(podporuje, (tj. je parciálním řešením) pro 1.). L2-vysoká (L1/L2 -vysvětlení viz níže).

(3) B.1. Objektivity chceme docílit tak, že aplikace bude umožňovat sbírání dat o konkrétních činech, resp. úkonech mocenského chování (tj. chování při hlasování) dané veřejně činné osoby tj. její výkonu veřejné služby (Naše řešení cíle č. 2).

(5) C. Rozšířenost. Aplikace má mít co největší dosah (tj. rozšířenost mezi lidmi). L2-vysoká.

(6) C.1. Aplikace bude uživatelsky příjemná. L2-vysoká.

Myšleno z hlediska uživatele i osvojovatele.

Z mého pozorování ve většině případů je celková nepřehlednost systému jakousi sumou mnohých drobných nepřehledností, které sice sami o sobě jdou poměrně lehce překonat, nicméně v masové kombinaci s dalšími vytvářejí onen prales, kterým se musí osvojovatel prosekávat. Je to analogie tření při pohybu tělesa o podložku, resp. přísloví „babka k babce, budú kapce“. Jakási průběžná nedbalost při odstraňování překážek pro budoucí osvojovatele (tj. Nemyslení na něj) tak vede k celkové nepřehlednosti systému. V takéto zbytečné nepřehlednosti (tj. Nepřehlednosti která nevyplýva ze složitosti samotného systému) vidím plýtvání duševními schopnostmi osvojovatele, protože jeho energie by měla být využita lépe, spíš na rozvoj systému, než na jeho osvojení. Zdá se mi, že když se k pŕehlednosti bude přistupovat systematicky, po malých krocích (kupř. přehlednost označení, názvů, etc.) může dojít snadněji k efektu „odtrhnutí laviny“ a člověk který systém studuje se do něj vtělí snáz a rychleji.

(7) C.2. Aplikace bude dostupná jako webová non-stop služba. (Řešení pro 5. a zároveň cíl pro body uvedené v části implementace). L2-vysoká.

(8) D. Realizace. Implementace. Aplikace bude schopna zabezpečit svůj vlastní chod (zde mluvíme o „fyziologických“ nevyhnutelnostech aplikace). Tento cíl je sice umělý, ale umožňuje odstínit (zkoncentrovat) všechny parciální cíle týkající se technických řešení od více abstraktních cílů (pozn. Toto dělení je taky umělé). L2-vysoká.

## Úroveň technických detailů:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| č. požadavku: | 10 |  | č. požadavku: | 11 |
| název: | Ukládání historie. |  | název: | Ukládání změn v systému. |
| uživatel: | dobrovolník, administrátor |  | uživatel: | dobrovolník, administrátor |
| popis: | Aplikace umožní uchovávat historii aktů veřejných činitelů a veřejných orgánů. |  | popis: | Aplikace umožní uchovávat historii aktů správců, tj. (uživatel dobrovolník a administrátor). Tedy ani zrušené entity se nebudou mazat, ale uchovávat. Stejně tak, jako i záznamy o každé změně. |
| priorita: | L1-vysoká |  | priorita: | L1-nízká |
|  |  |  |  |  |
| č. požadavku: | 12 |  | číslo požadavku: | 13 |
| název: | Vkládání dokumentů |  | název: | Vkládání komentářů |
| uživatel: | dobrovolník, administrátor |  | uživatel: | dobrovolník, administrátor |
| popis: | Aplikace umožní k příslušným událostem vkládání pdf dokumentů a obrázků. |  | popis: | Aplikace umožní dobrovolník-ovi vkládání komentářů k událostem. |
| priorita: | L1-vysoká |  | priorita: | L1-vysoká |
|  |  |  |  | ­­ |
| č. požadavku: | 14 |  | č. požadavku: | 15 |
| název: | Správu entit (objektů) |  | název: | prohlížení profilu veřejných osob, etc. |
| uživatel: | dobrovolník, administrátor |  | uživatel: | Všichni |
| popis: | Aplikace umožní správu (vytváření/naplňování daty/mazání) entit (objektů), které jsou popsané v DB modelu. |  | popis: | Systém bude generovat přehled údajů a historie působení jednotlivých veřejných činitelů / veřejných orgánů / a ostatních entit Tj. LOCATION, VOTE, a jiné... |
| priorita: | L1-vysoká |  | priorita: | L1-vysoká |
|  |  |  |  |  |
| č. požadavku: | 16 |  | č. požadavku: | 17 |
| název: | 3-jí úroveň zobrazování |  | název: | Přihlašovací procedura |
| uživatel: | všichni |  | uživatel: | dobrovolník, administrátor |
| popis: | Aplikace bude podporovat trojí úrovně práv v systému. (tj. např. trojí zobrazení stránek, podle typu uživatele, atd..) |  | popis: | Aplikace bude mít systém přístupu uživatelů (přihlášení, správa session, odhlášení) |
| priorita: | L1-vysoká |  | priorita: | L1-vysoká |

## Priorita cílů (požadavků)

Taky priorita je relativní pojem. Co může rozhodovat o úspěchu z dlouhodobého hlediska, nemusí být podstatné při samotném spuštění a funkci aplikace. Majíce toto na mysli, napadá mně opět pyramida, tentokrát z oboru psychologie - *Maslowová pyramida lidských potřeb*. Potřeby, které jsou u základny pyramidy, jsou sice pro život naprosto nevyhnutelné, nicméně na směrování lidského života mají jenom nepatrný vliv. Bez potřeb z vrcholu pyramidy se sice dá lokálně přežít, nicméně jsou to právě ony, které určují směr civilizací, kultur a vůbec celého lidského žití. Jako příklad, bych uvedl: můžete rokovat s vládou třeba o záchraně světa, no když se vám „chce“ – musíte si odskočit, děj se co děj - má to vyšší prioritu. Nicméně i záchrana světa má svou váhu, asi vyšší než onen úkon. Proto tohle rozdělení.

Maslow rozděluje potřeby do vícero vrstev, pater. Prozatím (v rychlosti) bych si nedovolil navrhnout (pojmenovat) více pater, a proto volím jenom dvě základní.

Úroveň 1 = základní, bez kterých aplikace nebude fungovat. (L1)

Úroveň 2, - věci důležité z dlouhodobého, strategického hlediska, bez kterých však aplikace fungovat bude (tj. nač bude fungovat, když ji nikdo nebude používat?). (L2)

V obou úrovních pak budou stupně: Vysoká - zásadní, Střední - doplňující, Nízká - drobnosti.

## B. Uživatelé a lidský faktor

Systém budou užívat 3 typy uživatelů.

0. **administrátor**. Kromě práv dobrovolníka bude mít přístup k historii změn a možnost přidávat / odstraňovat uživatele. Vyžaduje přihlášení.

1. **dobrovolník.** Členové neziskových organizací, které se zabývají kontrolou činnosti veřejně činných osob (poslanců, politiků), aby v případě jejich nekalé činnosti upozornili širší veřejnost. Bude mít kromě práv uživatele, taky možnost vkládat a odstraňovat entity, dokumenty, události , komentáře, hodnocení. Vyžaduje přihlášení.

2. **uživatel.** Role je určena pro běžného návštěvníka stránek. Má privilegia jenom k prohlížení stránek (nemá přístup do všech). Systém tedy nebude podporovat veřejnou diskusi k tématům.

## D. Fyzické umístnění

Aplikace bude pracovat jako non-stop webová služba na aplikačním serveru *Tomcat*. Server (hardware), kde je aplikace umístěna musí podporovat taky databázový server *MySQL*. Prozatím se předpokládá pronájem místa. Později, pokud bude aplikace úspěšná zakoupení vlastního serveru (hardware).

## E. Zajištění kvality

Uživatelé budou moci, pokud zaznamenají funkční chybu kontaktovat IT support a tuto jim sdělit. Jedná se o klasický post-launch support.

## F. Databázový model:

#### Navrhované entity:

1. PUBLIC\_PERSON, 2. PUBLIC\_ROLE, 3. TENURE, 4. PUBLIC\_BODY, 5. LOCATION, 6. VOTE\_OF\_ROLE, 7. VOTE, 8. SUBJECT, 9. THEME, 10. USER, 11. USER\_ROLE, 12. CHANGE, 13. ACT\_CLASSIFICATION, 14. PERSON\_CLASSIFICATION

##### 1. PUBLIC\_PERSON (veřejně činná osoba)

Představuje fyzickou osobu, tj. člověka, který počas svého života může vykonávat vícero veřejných funkcí, čili rolí.

##### 2. PUBLIC\_ROLE (veřejná funkce, resp. veřejně činná role)

Představuje konkrétní výkonní funkci působící ve veřejném životě. Tato je jednoznačně definována osobou, funkčním obdobím a veřejným orgánem (tj. PUBLIC\_PERSON, TENURE a PUBLIC\_BODY, od kterých dědí identitu).

##### 3. TENURE (její funkční období)

Funkční období ve variantě B je vztáhnuté vysloveně ke PUBLIC\_ROLE. Jedno funkční období může patřit i vícero funkcím – kdy se naplní scénář "od voleb do voleb", který se zřejmě bude týkat většiny.

Funkční období je jako samostatná entita jenom proto, aby jak zadání káže, bylo možné vyhledávat podle jednotlivých funkčních období. Jinak by zřejmě stačilo vložit do entity PUBLIC\_ROLE povinný parametr *since* a nepovinný *till* (resp. pokud se vždy dopředu ví jaké bude mít daný mandát trvání, tak může být i povinný).

##### 4. PUBLIC\_BODY (veřejný orgán, ustanovizeň)

Představuje veřejný orgán, který rozhoduje o předmětech hlasování (SUBJECT).

##### 5. LOCATION

Místo, kde se veřejný orgán nachází. V konečné verzi si to představuji graficky: nejdříve bude mapa Slovenska se třemi aktivními částmi (kraji). Po zvolení některého z nich se uživatel dostane do přiblížení s aktivními okresy daného kraje, pak obcemi, resp. v některých případech městskými částmi.

##### 6. VOTE\_OF\_ROLE (hlasování veřejně činné funkce)

Úkolem této entity bude uchovávat informace o hlasování dotyčné fyzické osoby, v rámci hlasování orgánu (VOTE) zprostředkovaně skrze funkci. Identitu dědí z entit PUBLIC\_ROLE a VOTE.

##### 7. VOTE (hlasování orgánu o věci)

Některé atributy VOTE, jako např. počet hlasů Za / Proti / etc. resp. celkový výsledek se můžou jevit jako nadbytečné, protože informaci v nich uloženou možno dopočítat z výsledků hlasování jednotlivých funkcí (a při tvorbě databáz je dobré se zdvojování informací vyhýbat, nakolik se osvědčilo skladování dané informace jen na jediném místě v DB). Na druhé straně se zvýší přehlednost a řešení případného rozporu bude lehce dohledatelné, takže se za tyto atributy přimlouvám. Atribut *link* bude odkazovat k elektronickému záznamu zápisu hlasování (např. skenu). Ačkoli není povinný (v době vytváření zápisu do systému dokument nemusí být k dispozici), kvůli důvěryhodnosti by měl být samozřejmostí. Atribut *internal\_number* bude odkazovať na číslování, které používá daný orgán, taky kvůli přehlednosti.

##### 8. SUBJECT (předmět hlasování)

Konkrétní věc, o které se hlasuje. Nepovinný je jeho vztah k témě. Zvolil jsem, že jej navrhuje, tj. předkládá na hlasování právě 1 osoba. Progamátorsky musí být zabezpečeno, že tato osoba bude členem orgánu, který jej schvaluje.

##### 9. THEME (kauza, resp. tématický okruh)

Širší tématický okruh, který se buďto řeší ve vícerých krocích hlasování - tj. kauza, nebo témata například: územní rozvoj, vzdělávaní, atd.

##### 10. USER (uživatel)

Uživatel přihlášený do našeho systému. Předpokládám, že široká veřejnost se kvůli lepší přístupnosti nebude muset přihlašovat, takže s ní počítat nebudu (nebude mít právo zasahovat). Takže návrh systému uvažuje s tím, že kdo bude přihlášen, bude mít právo vkládat údaje a zasahovat do systému. Mazat údaje bude mít právo jenom administrátor. Prozatím nebudu rozvádět do podrobností problematiku rozlišení různých úrovní přístupu a zůstanu při jenom jedné.

##### 11. USER\_ROLE(role přihlášeného uživatele)

Prozatím uvažuji jen o administrátorovi tj. uživateli se právem zapisovat.

##### 12. CHANGE (změna)

Tato entita je jednou z možností jak evidovat redakční zásahy. V této by se mohli evidovat všechny vložení od daného uživatele. Otázkou je, jak postupovat, když se má odstranit nevhodný záznam z minulosti. Myslím, že bude dobré žádné záznamy z minulosti nemazat, ale přidat do všech tabulek atribut *visible*, který se bude při "odstraňování" záznamu měnit z přednastaveného *true* na *false*. Hierarchické závislosti při změně hodnoty *visible* se budou muset řešit individuálně.

##### 13. ACT\_CLASSIFICATION (hodnocení, vyhodnocení aktu)

Implementaci této části můžeme pokládat za aktuální v pozdějším stadiu vývoje projektu, nicméně tak nějak předběžně s ní můžeme počítat již teď. Tato entita bude překládat do lidsky srozumitelného jazyka, o čem vlastně daný čin vypovídá (kupříkladu: pokud někdo bude při hlasování opakovaně trvat na vykácení všeobecně oblíbené zalesněné plochy za účelem výstavby vil prominentů, je oprávněné předpokládat, že této osobě neleží na srdci veřejné blaho, byť by byl ze strany Zelených). Hodnotil bych jedině akt VOTE\_OF\_ROLE (tj. Ne přímo veřejnou osobu. Zdá se mi, že VOTE separátně hodnotit netřeba), a taky SUBJECT, který souvisí s navrhovatelem. Protože to, že se někdo o něco pokouší (předkládá návrh hlasování) taky o dotyčné osobě něco vypovídá.

Kritéria hodnocení musí být takové, aby v co nejmenší míře mohli být považované za subjektivní a nemělo by jich být mnoho. Mohli by se např. hodnotit na škále 0 – 5. Napadají mne např.: *„soulad s programem“* a *„veřejná prospěšnost“.* U osoby by se též mohl vyhodnocovat parametr „*stabilita*“ neboli *"soulad sám se sebou"* (tj. jestli dotyčný nehlasuje podle toho, jak se vyspí, nebo jak zafouká vítr).

Každé hodnocení musí být podložené odkazy na fakta, které k tomuto hodnocení vedly.

Kvůli zvýšení objektivity vyhodnocování bych navrhoval, aby do úvahy přicházeli výhradně fakta (tj. návrhy hlasování a hlasovaní) uložené v modelu. Tedy model bude zachovávat presumpci neviny každé veřejné osoby, která bude v něm figurovat a tato může být kompromitována výhradně pouze vlastním chováním, které monitoruje náš systém. Vyhneme se tím částečně výtce, že na někoho přednostně "snášíme materiál".

Úplně ideální by bylo, kdyby tyto vyhodnocení dělal nějaký důmyslný a přitom transparentní (zveřejněný) algoritmus. V modelu předpokládám, že každá událost má nejvíce 1 hodnocení. Z těchto údajů by se jiným algoritmem mohlo / mělo v pravidelných intervalech (např. měsíčně) vypočítávat celkové hodnocení dané veřejné osoby. Tohle hodnocení bude ukládáno v entitě *PERSON\_CLASSIFICATION*.

##### 14. PERSON\_CLASSIFICATION (hodnocení veřejně činné osoby)

Představuje hodnocení veřejně činné osoby.

***Dodatek***

Ve fyzickém modelu (PH) nahrazuji složené primární klíče samostatnými ID. V ER modelu ponechávám tyto, kvůli zvýšení přehlednosti vztahů mezi entitami.

**Rozdělení tabulek.** Tabulky resp. Entity je buďto týkají samotného navrhovaného bussiness systému, nebo jsou pomocné, resp. popisují administraci. Kvůli větší přehlednosti proto tento fakt zahrnuji i do názvu tabulek. Zvýší se tím přehlednost.

# G. Technické detaily řešení

## G.1 Odstaňování / deaktivace entit

Jelikož entity nejsou na sobě nezávislé, ale mají určité vazby, bude nevyhnutné tyto vazby zohlednit i při jejich odstraňování, tj. Deaktivaci, protože pochopitelně nemá smysl držet v systému entitu, která odkazuje na jinou, již zrušenou. Využijeme k tomu stejný algoritmus jako v případě výše uvedeného filtrování.

Na obrázku č.XZY je uveden strom hierarchických vazeb entit vyskytujících se v našem systému.

Jedná se o stromový graf, kde kořeny jsou entity úplně vespod a listy úplně nahoru. V programu mám tento vztah nazvaný boss – slave.

Jeho jednotlivé vrstvy znamenají, čím nížeji je entita v hierarchii, tím má základnější postavení. Tj. Entita z 2. vrstvy nemůže existovat bez entity z vrstvy první na kterou ukazuje. V komplexnějším systému však narážíme na otázku: když odstraníme entitu A, které všechny entity by měli zmizet?

Algoritmus jejich vyhledávání při deaktivaci bude následovní:

1. Vyjdi z entity, kterou cheš odstranit.
2. Najdi všechny entity (co do počtu i typu) které na tuto entitu ukazují. Poznamenej si je.
3. Pro každou entitu z bodu 2. udělej postup popsaný v tom samým bodě (2).
4. Proces 3. opakuj dokud neskončíš na listové entitě (nikdo již na ni neukazuje).
5. Deaktivována musí být taky množina entit získaná sjednocením všech zaznamenaných entit z bodu 2.

Pozn. V některých případech se může stát, že po deaktivaci hlavní entity jsou podřízené entity z našeho systému nepřístupné (např. dokumenty dané entity). a tudíž jejich deaktivace se může jevit jako zbytečná akce, tj. porušení principu nezbytnosti akcí. Při podrobnějším pohledu zjistíme, že není tomu tak a ve skutečnosti méně energie mineme, když bezpečně deaktivujeme vše, co deaktivované býti má. Je to z toho důvodu, že one zákon nezbytnosti akcií, vychází z hlubšího zákonu "šetření energie", který je mu nadřazený. Pokud nezdeaktivujeme entity, který by stejně systém neviděl, musíme vydávat průběžně energii na to, aby se dalším vývojem aplikace nestali tyto jiným spůsobem přístupné. Tj. stále musíme zbytečně čelit této "hrozbě".

2. Hlavní výdej energie spočívá v tom, že musíme pracně rozlišovat, který strom entit deaktivovat celý a který jen částečně (protože podentity nejsou viditelné). Naopak, pokud deaktivujeme celý strom, tak jak je potřeba, administrátor při pohledu do databázy získá větší přehled o tom, co je v jakém stavu. Tj. zvýší se transparentnost, celého systému.

Vazby mezi entitami jsou dané samotnou definicí v databázy (cizí klíče). Pro urychlení však bude dobré si vtvořit strukuru, která tyto vztahy jednoznačně popisuje. Zvolil jsem formu další databázové tabulky A\_HIERARCHY. Výhoda toho způsobu je v tom, že daný vztah deklarujeme. Tím pádem získáváme větší kontrolu nad celkovým chováním filtru (pokud chceme z jakéhokoli důvodu některý vztah ignorovat, prostě ho v tabulce neuvedeme).

# Filtrování

Při řešení problematiky filtrování jsem narazil na 2 typy.

1. podle existenční závislosti, tj. Řekněme „podle potenciálního stavu“
2. podle (řekněme) ‘aktuálního stavu‘. – vysvětlím.

### Filtrování podle existenční závislosti

Při postupném zaplňování systému je třeba počítat s nepřehledným množstvím položek v jednotlivých tabulkách, nebo jednoduše potřebujeme z celého souboru entit vybrat jenom ty, které se hodí do vybraného pohledu. K tomuto účelu je nezbytné zavést filtrování. Jeho algoritmus bude založen na podobném principu jako odstraňování dokumentů z předchozí kapitoly, s tím rozdílem, že vyhledávání nepůjde od vybrané ovlivňující entity až na úroveň listů ale jen do takové hlobky, než zarazí na ovlivněnou třídu a zacházet se bude jenom se získanou množivou entit ovlivněné třídy.

Výhoda těchto postupů je v tom, že filtrování je naprosto univerzální, tj. Při jakékoli kombinaci: ovlivňující vs. ovlivněná třída. To platí i pro odstraňováni entit.

Při vkladání nových entit nás z logiky věci zajímá filtrování prvního druhu, protože když vkládáme novou entitu, zajímá nás jaké jsou všechny možnosti parametrů, které ji mohou přislouchat.

Naopak při pasivním prohlížení nás zajímají jenom parametry, které byly zvolené a zrovna v systému jsou.

### Filtrování podle aktuálního stavu

Pokud si zvolíme kupř. Veřejný orgán, který nás zajímá. Tak sice tématický okruh hlasování (školstvo, životní prostředí, etc.) je sice od tohto nezávislý, nicméně zajímají nás jenom ty na které odkazují hlasování uskutečněné v daném veřejném orgánu. Tento typ filtrování budu využívat při tvorbě jednotlivých pohledů – View.

# Ukládání změn

Při každé změně systému se tato zapíše do databáze do specielní tabulky na to určené (tabulka A\_Change). Do této tabulky se uloží datum změny (timestamp), id osoby, která změnu provedla, název tabulky, kde změna proběhla, id řádku v tabulce, kde změna proběhla, stará hodnota, nová hodnota.

Jsou dvě možnosti jak změny ukládat a to na straně Javy, nebo na straně Databáze, tj. V rámci databázového stroje. Výhodou ukládání změn na straně databáze, je určitě rychlost.

O nevýhodách tohto spůsobu bych chtěl trochu pojednat.

Pokud existují 2 nezávislé systémy, které zasahují společnou oblast, ze zkušeností platí, že je dobré pokud je jeden systém jakoby hlavní a druhý jakoby podřízený. Tedy něco, že na jednom smetišti by měl být pánem jenom jeden kohout. Pokud tomu tak není, musí být vyřešena jejich synchronizace, protože jenom tak můžeme předejít zbytečným kolizím. Z praxe vychází synchronizace téměř vždy nákladnější na prostředky v porovnáním s prostým vymezením sfér vlivu. Vymězení autonomních oblastí s pouze jedním „pánem“ bývá jednodušší, přehlednější a tímpádem více odolné vůči chybám.

Většina logiky se však řeší na straně Javy. Pokud by se část logiky přesunula na stranu DB, nutně by vznikli „kolizní plochy“. Nemuselo by tak tomu být, kdyby kupř. bylo žádoucí do DB tabulky zapsat úplně jakoukoli změnu. Jak však ukážu dále, některé změny zapsat do tabulky změn není vhodné a tudíž je situaci jednodušší řešit na straně Javy i za cenu zpomalení aplikace.

Taky by mohla vzejít námitka, že je zbytečné v A\_CHANGE ukládat novou hodnotu, když je přece tato uložena v samotné (jiné) DB tabulce. To je sice pravda, ale uložení i této hodnoty celkově zjednoduší rekonstrukci předchozího stavu a taky může představovat formu zálohy dat, takže získá se tím robustnější a v neposlední řadě přehlednější řešení.

Systém bude taky nastaven tak, aby změna objemých dat (tj. dokumenty) nemohla probíhat (aby se zbytečne nezatěžovala tato tabulka). To je týká u nás jedině tabulky T\_DOCUMENT ,tj. Pokaždé, když bude do systému vložen dokument, bude mu přiděleno nové id, a změna z hlediska uživatele, bude představovat přidání nového řádku v DB.

Univerzální formát.

Jelikož v tabulkách jsou hodnoty uloženy v nejrůznějších formátech (VARCHAR, DATE, TEXT, BIT, etc..) na to, aby mohli být ukládány v jednom stloupci, je potřebné najít univerzální formát na který bude možno transformovat jakoukoli hodnotu a taky co nejjednoduššeji provést zpětnou transformaci.

Nejuniverzálnějším fromátem se jeví proud bytů, tj. Kupříkladu BLOB. Nicméně, pokud jsme ze systému vyloučili změny dat, které tento formát vysloveně vyžadují (tj. dokumnetů). Druhým univerzálním formátem je String, tj. VARCHAR, ze kterého jde snadněji převod zpátky na potřební formát. Při použití typu BLOB by stejně jeden mezikrokem musel být pravděpodobně typ *String.*

## Ukládání dokumentů

Dokumenty je možno ukládat v databáze jako typ BLOB, nebo jako odkazy k adresárové struktúře na serveru. Výhodou dokumentů uložených v adresářové struktuře je to, že jsou dostupné i jiným způsobem, a též, že odlehčují databázový provoz. Naopak ukládáni dokumentů v databáze se mi její jako elegantnější přenostitelnější řešení.

## Návrat do bodu v minulosti

Díky struktře ukládání změn bude návrat do minuloasti(stavM) jednoduchou iterací skrz tabulku A\_change od současného dne(stav S) až do okamihu ke kterému se chceme vrátit, přičemž v každím iteračním kroku provedeme reverzní krok změny. (tj. pokud byla stará hodnota v tabulce X 5 a nová 4, změní se hodnota v tabulce X na 5.) Tímto způsobem se dostaneme postupně až do stavu, který panoval v době, kterou jsme si zvolili. Model umožní taky deaktivovat i akce vybraných jednotlivých uživatelů, ale za tímto účelem bude potřeba zabezpečit současnou deaktivaci takových změn jiných uživatelů, které jsou od na změnách našeho uživatele závislé.

Důležité je tyto změny provést mimo historii uchovávací systém, jinak se bude zbytečně jako změna počítat i návrat do minulosti – docházelo by k nežádoucímu „zrcadlovému“ množení údajů v této tabulce. To je jeden z důvodů proč jsem nezvolil sledování změn skrz triggery na databázové vrstvě. Tento stav bude potřeba reflektovat i metod ukládání do databáze, tj. Něco ve smyslu: „do simple step only“ a „do complex step“.

Všechny tyto problémy vyplývají z toho, že náš navrhovaný systém ukládaní změn umožňuje jen jednu „větvu“ historie. Pokud tedy ve stavu do kterého jsme se vrátili, vykonáme další změny (tj. Stav D) a opět se budeme chtít vrátit do „budoucnosti“(stav S), promítnou se nám tyto změny také. Toto „prolínání“ větví se mi nejeví jako dobrý nápad (s největší pravděpodobností by musel systém značně zkomplikovat, protože by museli být řešeny kolize resp. Existenční závislosti mezi tím co bylo přidáno a „budoucí“ větví). Pokud tedy nechceme systém rozšířit o možnost pracovat ve větvích (analogie s verzovacími systémy např. GIT), musíme při každém vrácení se do minulosti krok ze kterého jsme se vrátili v tabulce CHANGE vymazat. Aby poslední řádek v této tabulce zodpovídal stavu ve kterém se skutečně nacházíme. Když však tyto řádky skutečně vymažeme, ztratíme možná cennou informaci. Proč je tedy taky jenom nedeaktivovat? Taková deaktivace by však z pohledu zevnitř systému byla nezvratná. Tj. Tyto řádky by viděl jenom superdministrátor s přístupem do databáze. Uvedu jako možnost budoucího vývoje, že tyto změny by pak mohli být obhospodařovány v další tabulce, do které by sa zapisovali změny v tabulce CHANGE.

Z důvodu, že se id uživatele ukládá do tabulky změn, musí být i tato a od ní závislé administrátorské tabulky deaktivovatelné (tj. Entity se nemohou maza). Změny v nich však budou ze systému zachytávání změn vyloučeny (není potřeba tyto údaje ucovávat).

## Oživování mrtvých entit.

Systém umožní taky oživení mrtvých entit a to buďto jednotlivě nebo s celým stromem, který z tohto kořenu vyrostl, podle výběru uživatele. Tyto změny se do tabulky CHNGE zapisovat budou.

# Návrh mapy stránek

Jsou zobrazeny podle typu uživatelů. Vždy platí, že vyšší role zahrnuje, nebo překrývá stránky nižší role.

### Mapa stránek uživatele.

### View 1. Přihlášení:

Netýká se role uživatel.

### View 2. Vstupní stránka:

Vstup: View 1, nebo nic, resp. každý pokus o vstup na více privilegovanou stránku.

Výstup: View 3, View 4.

Společná pro všechny role. Na vstupní stránce bude možnost si zvolit zemepisnou oblast, kde sa nachází veřejný orgán(View. 3) , nebo konkrétní osobu(View 4), která nás zajímá.

### View 3. Veřejný orgán(E):

Vstup: view 2,

Výstup: view 2, view 5, view 6

Na této stránce budou zobrazeny základní údaje verějného orgánu (adresa, předseda, seznam aktuálních poslanců, etc..), komponenta veřejných rolí (3.a). a všech hlasování (3.b) daného veŘejného orgánu. resp. sady hlasování. Bude tu taky volba náhledu do historie, po zaškrtnutí které se objeví výběr hasovacích období, resp. kalendář které umožňují náhled do minulosti. Protože se entita volebního období nevztahuje k verejnému orgánu, ale veřejné osobě, je teoreticky možné, že v daném momentu jsou činné veřejné role, které nemají odkaz na stejnou entitu volební období (TENURE). V tomto případě bude k filtrování historie kalendář. Běžnou praxi však je, že volební období se prakticky vždy vztahuje ke veřejnému orgánu a proto ponechám filtrování (2. druhu) i přes výběr volebních období, které je uživatelsky logičtější.

**komponenta 3.a Veřejné role:** V této komponentě budou zobrazeny buďto aktuální sada aktuálních rolí (tj. např. poslanci), nebo po zaškrtnutí historie sady rolí které byli aktuální v jistém momentu historie.

**Komponenta 3.b Hlasování (plurál):** Tato komponenta zobrazí všechna hlasování, která se odehrala na půdě daného veřejného orgánu ve vybraném volebním období. Pokud nebude volební období pro všechny veřejné role daného veř. Orgánu stejné, bude tu vymezeno rozpětí od – do (kalendáře). A podívat se na jejich detailnější zobrazení (View 6.). Zde bude také grafická komponenta zobrazující rozmístnění hlasování v čase.

### View 4. Veřejná osoba(E):

Vstup ze: View 2, view 3, view 5

Výstup na: view 2, view 3, view 5, view 6

Po zvolení veřejné osoby (View 2) budou zobrazeny její základní osobní údaje a možnosti výběru její veřejných rolí (4.a), které zehrála, nebo nahlédnutí do hodnocení této osoby(4.b).

**Komponenta 4.a Veřejné role:** Zde bude možné vybrat konkrétní veřejnou roli, pro zvolené volební období.(View 5.)

**Komponenta 4.b Hodnocení veřejné osoby(E):** Zde bude k nahlédnutí hodnocení veřejné osoby.

### View 5. Veřejná role(E):

Vstup: View 3, view 4

Výstup: view 3, view 4, view 5, View 6

Zde se zobrazí základní údaje dané veřejné role a seznam hlasování ve kterých se daná osoba zúčastnila. Vedle hlasováních bude uvedeno jak se tato role v daném hlasování zachovala.

**(//variant 2: hlasovania bude combo box a po výbere sa oživí komponenta 5.a Komponenta 5.a Hlasování role(E):** //Po vybrání hlasování se zobrazí, jak tato role hlasovala.)

### View 6. Hlasování (E):

Vstup: View 3, view 4, view 5

Výstup: view 3, view 5

Zde se zobrazí podrobnosti vybraného hlasování (tj. výsledku hlasování) + zobrazení kdo jak hlasoval.

**Komponenta 6.a Předmět hlasování(E):** Podrobnosti o předmětu hlasování (k danému předmětu se může hlasovat více-krát).

**Komponenta 6.b Tématický okruh hlasování(E):** Přislouchá předmětu hlasování. Podrobnosti o něm.

**Komponenta 6.c Hlasování osob:** Zde se zobrazí kdo jak hlasoval.

**Komponenta 6.d Klasifikace Hlasování:** Zobrazení ohodnocení hlasování.

Ve všech stránkách / komponentách které zodpovídají nějaké entitě, bude komponenta zobrazující dokumenty přislouchající této entitě ke stažení.

### Mapa stránek dobrovolníka.

Změněné stránky: Na každé stránce, která představuje entitu (označení E) přibudne tlačítko: „přidej novou entitu“ a komponenta dokumentů bude editovatelná, tj. bude možnost přidávat/odebírat dokumenty. Přibudne stránka (View 7.), na které budou odkazy na univerzální správcovké rozhraní pro všechny entity. Samotný editovací stránka (view 8) bude univerzální, tj. prispůsobena všem typům entit. Umožní jak editaci, tak přidávání i mazaní entit tříd týkajících se business modelu a editace vlastního profilu (hesla a pod).

### Mapa stránek administrátora.

Všechno jak pro dobrovolníka. Editace i administračních tabulek, přidávaní a mazání nových uživatelů. Přístup k možnosti navrácení systému.

# Možnosti rozšířování systému.

**Problém vyššího počtu dobrovolnických skupin.** V případě, že se systém rozšíří, bude třeba odseparovat různé dobrovolnické skupiny, aby si „nekafrali do zelí“. Řekněme, že budou 2 dobrovolnícke skupiny, monitorující chování městkých zastupitelství v Košicích a Bratislavě. Jak by třeba vypadalo návrat do minulosti? Co by tomu řekli lidé z druhé skupiny?

Tento problém je možné řešit zahrnutím veřejného orgánu do přihlášení (je pravděpodobné, že dobrovolníci z jedné skupiny budou „obsluhovat“ jenom jeden verejný orgán) a daný dobrovolník bude mít dosah jenom na entity příslušící danému orgánu. To samé platí pro administrátora jak jsme si jej definovali výše. Jeden administrátor by měl připadat na jednu skupinu dobrovolníků, ideálně pro jeden veřejný orgán. Nicméně musí tady být potom vytvořena další role, řekněme „super-admin“, která umožní přidávání entit veřejných orgánů a tvorbu kont lokálních administrátorů. Tady by zase platilo, 1 super-amin na jednu databázi.

**Řešení 1.** Všechny tyto argumenty hovoří v prospěch oddělených databází. To by však mělo za nevýhodu to, že pokud daná osoba působila v různych veřejných orgánech, její existence v systému bude vícnásobná a tyto budou vzájemně nezávislé. Pak v jedné společné databázi(D0) budou tabulky T\_KRAJ, T\_OKRES, T\_LOCATION, T\_PUBLIC\_BODY a T\_DATABASES\_CONNECTION z těchto tabulek nebude možné mazat. V lokálních databázích(D\_loc) pak zase budou tabulky od T\_PUBLIC\_BODY včetně všechny ostatní a tabulka T\_PUBLIC\_BODY bude obsahovat právě jeden záznam. Při vložení nového záznamu do T\_PUBLIC\_BODY v D0 se spustí script na vytvoření nové databáze (název nové DB, login a heslo se uloží do T\_DATABASE\_CONNECTION), tabulkové schéma, inicializuje nového administrítora (tj. např „admin“/“admin“), a vloží 1 (jediný) údaj do T\_PUBLIC\_BODY. V Java světě se budou udržovat 2 databázová spojení do D0 i D\_loc, přičemž D\_loc se operativně nastaví při vstupu na stránky (uživatel si bude muset vybrat veřejný orgán, kam bude chtít vstoupit). Do D0 bude mít přístup jen super-admin.

**Řešení 2.** Budeme u dobrovolníku předpokládat „dobrou vůli“, tj. že nebudou vědomě zasahovat do cizích záležitostí. Do tabulky A\_CHANGE pak přidáme stloupec „public\_body\_id“, který bude evidovat příslušnost k veřejnému orgánu. Návraty do minulosti pak budu zohledňovat jenom řádky, které se týkají našeho veřejného orgánu (tj. oblasti, kde pracují dobrovolníci). Vstup do systému bude opět podmíněn výběrem veřejného orgánu (kvůli znížení pravděpodobnosti vzniku chyb. Kdyby totiž dobrovolníci pracovali na různých veřejných orgánech součastně, a chtěli se vrátit do minulosti, nesměli by zapomenout návrat skrz všechny veřejné orgány, na kterých by pracovali, takhle to bude pro ně více zřejmé, protože se budou muset přepnout).

# Závěr

Nepovažuji tuto práci za zdaleka ukončenou. Některé části, kterými se budu zaobírat, jsem již výše nastínil a mohé další na mne jistě čekají v záloze. Mým soukromým cílem není, aby aplikace byla čím dřív na světě, ale aby až vyjde, jsem s ní byl vnitřně spokojen.

OMNIS FESTINATIO EX PARTE DIABOLI EST.