Záverečná Správa

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO

BRATISLAVA, 2016–2017

Portál pre vernostné programy

Vypracovali

Ladislav Bilisics
Martin Bohumel
Martin Kellner
Ivan Latták

Obsah

Kapitola 1

Úvod

Tento dokument popisuje kompletnú aplikáciu Portál pre vernostné programy. Dokument opisuje požiadavky na funkcionalitu, použité technológie, návrh softvéru z technického i vizuálneho hľadiska. Obsahuje taktiež návod na inštaláciu aplikácie.

Kapitola 2

Špecifikácia požiadaviek na softvér

2.1 Úvod pre špecifikáciu

2.1.1 Predmet špecifikácie

Táto špecifikácia požiadaviek na softvér popisuje používateľské, funkčné a parametrické požiadavky na webovú aplikáciu zobrazujúcu stromovú štruktúru používateľov. ŠPS je súčasťou dohody medzi objednávateľom a dodávateľom, a bude slúžiť ako východisko pre vyhodnocovanie správnosti fungovania predmetnej webovej stránky.

2.1.2 Rozsah projektu a funkcie systému

Webová aplikácia bude pozostávať z časti pre bežné webové prehliadače a pre zariadenia android. Našou úlohou je taktiež generovať náhodné dáta do už existujúcej databázy. Aplikácia bude prehľadne zobrazovať stromovú štruktúru používateľov a vzťahy medzi nimi v oboch implementáciach.

2.1.3 Slovník pojmov

- Strom, dátová štruktúra v ktorej sú uložení používatelia.
- Otec, je taký používateľ, ktorý pozval aspoň jedného ďalšieho používateľa.
- Syn, je každý používateľ okrem koreňa stromu a má práve jedného otca.

2.1.4 Prehľad nasledujúcich kapitol

Tento dokument ďalej popisuje perspektívu a funkcie produktu, charakteristiky používateľov, všeobecné obmedzenia, predpoklady a závislosti a požia-

davky na funkčnosť produktu, ktoré môžeme nájsť v druhej kapitole.

V tretej kapitole nájdeme informácie o generovaní dát na účely testovania. Informácie ohľadom vizualizácie a optimalizácia pre android.

2.2 Všeobecný popis

2.2.1 Perspektíva projektu

Produktom je webová aplikácia, ktorá vytvorí prehľad informácii o používateľoch v databáze a vzťahy medzi nimi. Vzťah je definovaný ako kto bol kým pozvaný do hierarchie. Táto hierarchia vytvára stromovú štruktúru (ďalej už len strom), kde každý okrem tvorca služby má priradenú osobu, ktorá ho do hierarchie pozvala. Nazývajme pozvanú osobu „synom“ (ďalej už len syn) a osobu ktorá pozývala „otcom“ (ďalej už len otec) pre zjednoduhšenie. Keď syn zaplatí za nejaké služby vrámci už existujúcej aplikácie, jeho otec dostane podiel podľa vopred určenej tabuľky. Taktiež ak syn dostane od svojich synov peniaze, jeho otec dostane podiel. Naša aplikácia aj tieto tranzakcie zachytáva vo forme ziskov otca od jednotlivých synov, ktoré v rámci stromu zapísané nad jednotlivými synami.

2.2.2 Funkcie produktu

Ako už bolo vyššie spomínané, produkt zobrazí data z databázy a prevedie ich do prehľadného stromového grafu. Používateľ aplikácie si bude môcť rozkliknúť jednotlivých synov a zistiť svoje zisky, ktoré on od nich dostáva a ich synov. Spracovanie grafu bude vo forme webovej aplikácie, ktorá by mala fungovať v prehliadači ale aj na mobilnom telefóne. Pre nízke zaťaženie zariadení a ich optimalizáciu si bude môcť používateľ pozerať strom po desiatich synoch s možnosťou prepnúť na ďalších desať ak bude chcieť a existujú. K produktu bude taktiež pribalený generátor náhodných údajov pre testovanie produktu. V generátore sa vygeneruje niekoľko desaťtisíc používateľov, ktorým bude priradený ako otec, tak aj ich synovia a ich fiktívne zisky. Generátor má za povinnosť vytvárať stromy čím viac rôznorodé, čo znamená aby boli hlboké, plytké, široké aj úzke.

2.2.3 Charakteristika používateľov

Používateľmi stránky sú podnikatelia a zákazníci, ktorí pozývajú ďalších používateľov. Majú možnosť si prezerať svojich synov,ich synov a ich zisky pomocou nášho produktu.

2.2.4 Obmedzenia projektu

Stránka/aplikácia bude obmedzená len pre už zaregistrovaných používateľov, ktorí boli na stránku pozvaní. Otec smie vidieť len a iba svojich synov, nesmie mať sprístupnenú inú časť stromovej štruktúry.

2.2.5 Predpoklady a závislosti

Predpokladom je už existujúci používateľ (zadávateľ projektu), ktorý pozve prvých synov do projektu. Taktiež pedpokladom je, že zadávateľ už má systém pre nakupovanie produktov zákazníkmi a vytvorenú databázu z ktorej bude produkt spracovávať data. Medzi závyslosti rátame funkčnosť JavaScript-ového stromu nielen v prehliadači, no aj jeho mobilnom prevedení pre mobilné zariadenia s použitím WebView, ktorý sa využíva hlavne na zobrazovanie stránok bez javascriptu. Je pravdepodobné, že táto technológia nebude fungovať. V tom prípade budeme vytvárať jednoduchšiu javascriptovú aplikáciu, alebo použijeme čisto HTML+CSS. Ďalej môžeme implementovať aj natívnu vizualizáciu. Taktiež musí podporovať prehliadače Google Chrome (verzia 54+), Microsoft Edge (verzia: 38+), Opera (verzia: 40+) a Mozilla Firefox (verzia: 49+).

2.3 Špecifikácia požiadaviek

2.3.1 Generovanie pseudonáhodných dát do dátovej štruktúry používateľov

Generovanie používateľských dát

Program generuje dáta všetky potrebné dáta pre 50 tisíc používateľov. Používatelia sú entity, ktoré na seba nadväzujú a vytvárajú stromovú štruktúru. Generované dáta nadobúdajú v rôzne tvary stromovej štruktúry

2.3.2 Vizualizácia dátovej štruktúry v podobe prehľadného stromu

Nástroj na vykresľovanie stromovej štruktúry vo webovom prehliadači

Zvoliť existujúcu knižnicu alebo aplikáciu v programovacom jazyku JavaScript, ktorá umožňujú vykresľovanie stromovej štruktúry vo webovom prehliadači.

Konfigurácia nastroja na vykresľovanie stromovej štruktúry

Nastroj na vykresľovanie musí byť nakonfigurovaný, tak aby zobrazil rôzne dáta stromovej štruktúry. Tieto dáta musia byť pre koncového používateľa

priehľadne.

Optimalizácia pre systém Android

Dátová štruktúra musí byť prispôsobená pre mobilné zariadenia s operačným systémom Android.

2.3.3 Implementácia vizualizácie dátovej štruktúry do desktopových prehliadačov

Graf, ktorý je predmetom bodu 3.2, sa bude zobrazovať na neskôr bližšie určenej webovej stránke.

Technické požiadavky na prehliadače

Stránka bude zobraziteľná a plne funkčná na internetových prehliadačoch Microsoft Edge verzie 38 alebo vyššej, Mozilla Firefox verzie 49 alebo vyššej, Google Chrome verzie 54 alebo vyššej, a Opera verzie 40 alebo vyššej.

Prístupové práva

Pre správcu (administrátora) aplikácie bude prístupný na zobrazenie celý graf užívateľov. Pre používateľov bude prístupný na zobrazenie iba podstrom, ktorého sú oni koreňom, teda graf iba tých používateľov, ktorým je daný používateľ predkom.

2.3.4 Implementácia vizualizácie dátovej štruktúry do aplikácie systému Android

Graf, ktorý je predmetom bodu 3.2, sa bude takisto zobrazovať v mobilnej aplikácii pre systém Android. Túto časť budeme implementovať pomocou WebView, prípadne CSS+HTML, alebo implementujeme natívnu vizualizáciu. Táto funkcionalita softvéru má nižšiu prioritu.

Technické požiadavky na operačný systém telefónu

Aplikácia, v ktorej bude graf zobrazovaný, bude spustiteľná iba na operačnom systéme Android verzie 4.4 (Android KitKat) alebo vyššej.

Prístupové práva

Prístupové práva pre prehliadanie grafu cez Androidovú aplikáciu sú rovnaké ako cez desktopový internetový prehliadač. Pozri bod 3.3.2.

Kapitola 3

Návrh

3.1 Úvod návrhu

Táto kapitola obsahuje podrobný popis a komplexný návrch webovej aplikácie pre vernostné programy, ktorá umožňuje jej používateľom prezerať svoje zisky z ich synov ktorých do hierarchie pozvali. Kapitola obsahuje:

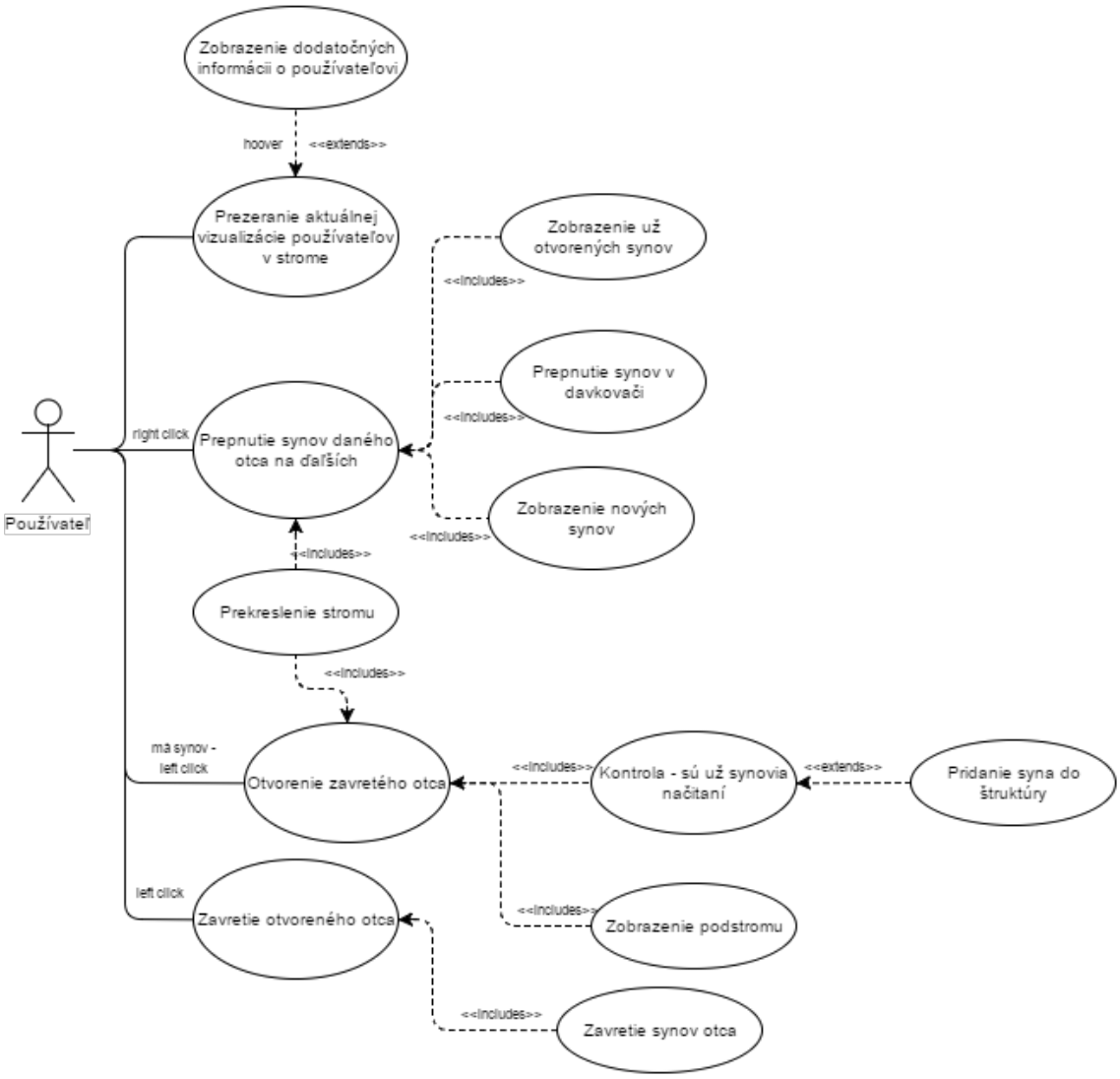
- Popis modelu stavového diagramu
- Analýzu používateľov aplikácie a -case diagram
- Analýzu generovaných dát pre účel rôznych testov
- Popis a zhrnutie testovacích scenárov
- Popis stavového diagramu
- Analýza používateľského rozhrania aplikácie pre používateľa a vzhľad zobrazeného grafu
- Popis triedneho diagramu
- Popis data model diagramu
- Presný opis funkcionality - Object design
- Popis entitno-relačného diagramu a komponenty aplikácie

3.2 Analýza používateľov aplikácie

3.2.1 Používatelia

Používateľmi aplikácie sú podnikatelia a zákazníci, ktorí pozývajú ďalších používateľov. Majú možnosť si prezerať svoje zisky z používateľov ktorých pozvali.

3.2.2 Use-case diagram



Obr. 3.1: – use-case diagram používateľa systému, vytvorené v online službe: www.draw.io

Use-case diagram popisuje interakciu používateľa so systémom. Používateľovi poskytuje možnosť prezerať si stromovú štruktúru a ak prejde myšou nad daného syna stromu, tak sa mu vypíšu dodatočné informácie o ňom. Zobrazovanie synov je obmedzené na počet, nastavený v systéme. Ak si chce používateľ prezrieť ďalších synov, musí kliknúť na otca pravým, a sytém automaticky zobrazí nový strom. Pri tejto akcii musí systém prepnúť synov v dávkovači, zobraziť nových synov a načítať všetkých synov načítaných synov, ktorí už boli niekedy zobrazení. Ľavým kliknutím na otca sa buď otvoria jeho synovia, alebo zavrú, podľa jeho aktuálneho stavu.

3.3 Generované dáta

Budeme náhodne generovať okolo 40 000 jedincov v strome, ktorí budú zoradení do stromovej štruktúry. Údaje budú meno, zárobok, id a id otca. Jedinci budú rovnomerne rozmiestnení do rôznorodých stromov s rôznymi šírkami a hĺbkami.

3.4 Testovacie scenáre

3.4.1 Testovanie zobrazenia hlavných údajov synov

Test spočíval v tom, či dávkovač správne dodával a vypisoval údaje používateľov, ktorí boli do neho pridaní.
Tento test pobieha nasledovne:

- Dávkovač je naplnený testovacími dátami
- Vypýtame si aktuálny strom
- Porovnávame data s očakávanými

3.4.2 Testovanie hoover funcií

Test spočíva v tom, či sa data používateľa načítali a zobrazili korektne pre syna, nad ktorým sa používateľ nachádzal. Vytvorenému stromu skontrolujeme, či sa naozaj po vykreslení vypisujú a priadzujú správnemu synovi. Test je potrebné vykonať aj po prepnutí synov, aby bolo isté, že tieto dve funkcie sú kompatibilné.

- Dávkovač je naplnený testovacími dátami
- Vypýtame si aktuálny strom a vložíme ho do stromu
- Porovnávame data v html vytvorené eventom s očakávanými

3.4.3 Testovanie prepnutia synov otca

Testuje sa, či sa korekne po pravom kliknutí otca prepli jeho synovia a či sa funkcia nevykonávala pre používateľov bez synov. Taktiež ak aktuálni synovia boli poslední z výberu, musel dávkovač korektne dodať prvých synov otca. Tiež sa testuje korekné rekurzívne načítanie synov synov tohto otca, ak boli už otvorení.

- Dávkovač je naplnený testovacími dátami
- Prepneme jeho synov
- Vypýtame si aktuálny strom
- Porovnávame data s očakávanými

3.4.4 Testovanie otvorenia otca

Testovanie načítania synov

Ak otec už mal načítaných synov, kontroluje sa, aby sa nenačítavali duplicity ale len sa vykonalo ich zobrazenie.

- Dávkovač je naplnený testovacími dátami
- Prepneme jeho synov
- Vypýtame si aktuálny strom
- Porovnávame data s očakávanými

Testovanie pridania syna, ak nie je v štruktúre

Testovanie kontrolovalo, kedy sa zavolá funkcia na pridanie synov otca, pri presiahnutom počte a kliknutí na otca v spodnej časti načítanej štruktúry.

- Dávkovač je naplnený testovacími dátami
- Simulujeme kliknutie na syna, ktorého synovia nie sú načítaní v štruktúre
- Vypýtame si aktuálny strom
- Kontrolujeme, či boli pridaní očakávaní synovia

Testovanie obmedzenia na hĺbku stromu

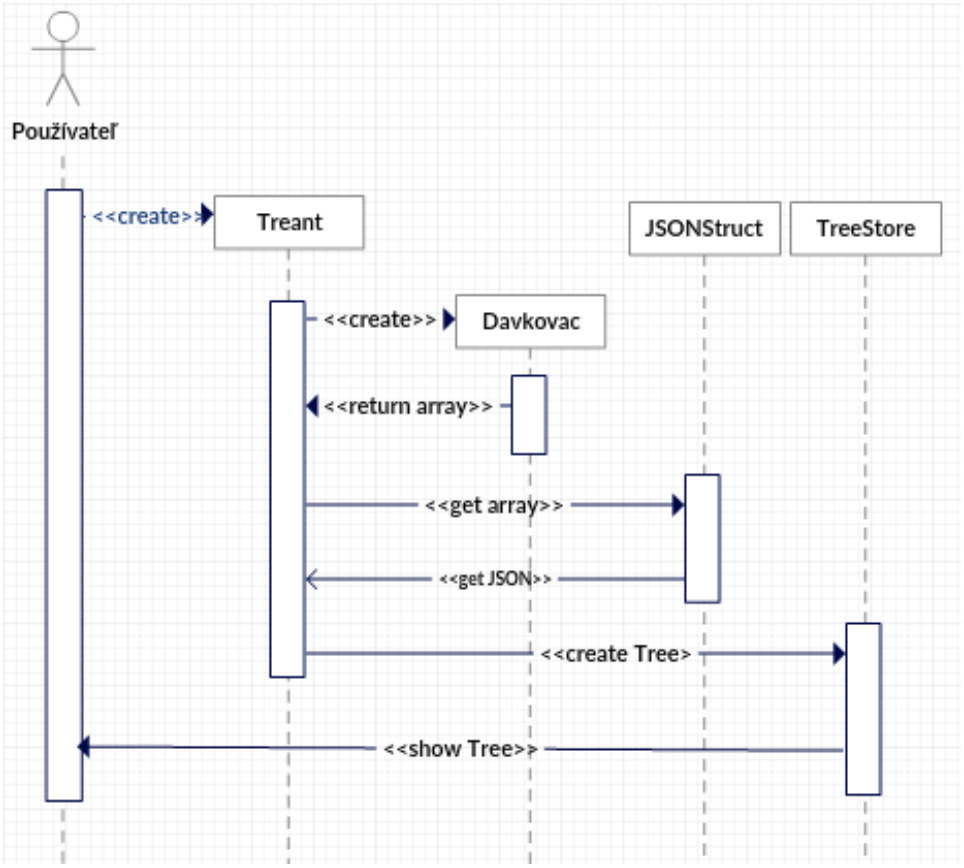
Tento test bol pomerne jednoduchý, naším cieľom je, aby neboli všetci synovia načítaní ale skrytý v grafickom zobrazení. Preto je nastavené obmedzenie, ktoré určuje, koľko chceme mať približne synov načítaných, keď strom prvýkrát otvoríme. Pri rekurzívnom načítaní synov sa má funkcia teda zastaviť nielen ak dosiahne všetky listy stromovej štruktúry, ale ja keď počet už pridaných synov je väčší ako tento počet. Kontrolujeme finálny počet synov, či presiahol túto hodnotu, dokončil úroveň načítania a pridal očakávaných synov.

- Dávkovač je naplnený testovacími dátami
- Nastavíme obmedzenie na hĺbku na nižsie, napr. 5
- Vypýtame si aktuálny strom
- Kontrolujeme, či aktuálny strom je obmedzený do očakávanej hĺbky a listy v tejto hĺbke sú všetky načítané

Taktiež ak syn, ktorý má synov ale nemá ich načítaných, musí byť kliknuteľný, aby sa dala vykonať funkcia pridania. Synovia ktorí synov nemajú túto vlastnosť mať nesmeli. Tento test prebieha nasledovne:

- Dávkovač je naplnený testovacími dátami
- Nastavíme obmedzenie na hĺbku na nižsie, napr. 5
- Vypýtame si aktuálny strom
- Kontrolujeme, či listky v aktuálnom strome, ktorí majú synov, sú kliknuteľní a synovia, ktorí nemajú synov, nie sú kliknuteľní

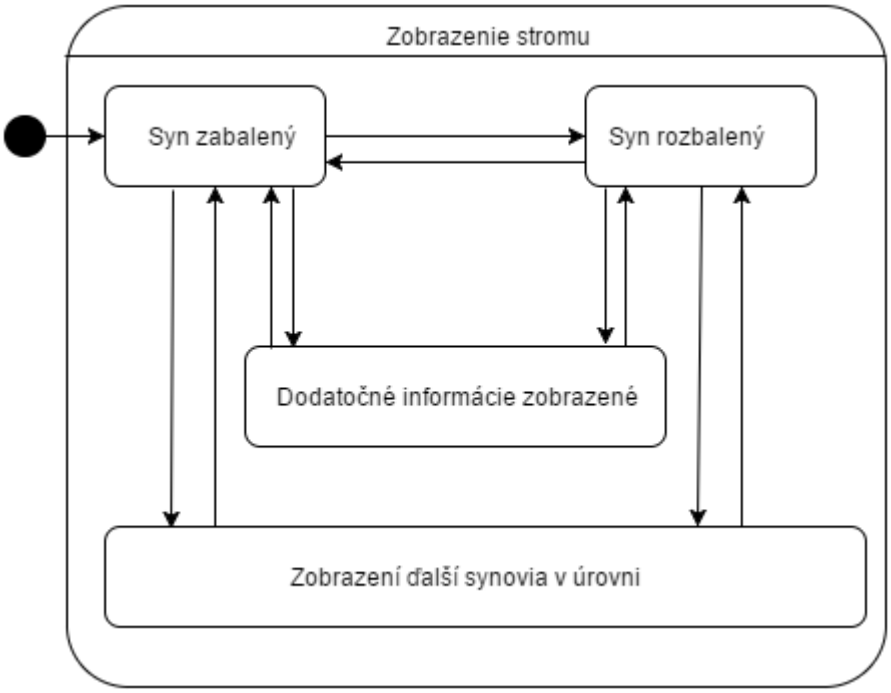
3.5 Model sekvenčného diagramu



Obr. 3.2: – Sekvenčný diagram, vytvorené v online službe: www.draw.io

Sekvenčný diagram sa týka scenára `Prezeranie aktuálnej vizualizácie používateľov v strome`. Konkrétne prvé zobrazenie, teda moment keď sa používateľovi zobrazí stromová štruktúra prvýkrát. `Treant` vytvorí `Davkovac` a ten mu poskytne pole, ktoré obsahuje údaje pre jeden vrchol stromu. Ďalej sa vytvorí JSON štruktúra a následne sa na základe tejto štruktúry vytvorí strom a zobrazí používateľovi.

3.6 Model stavového diagramu



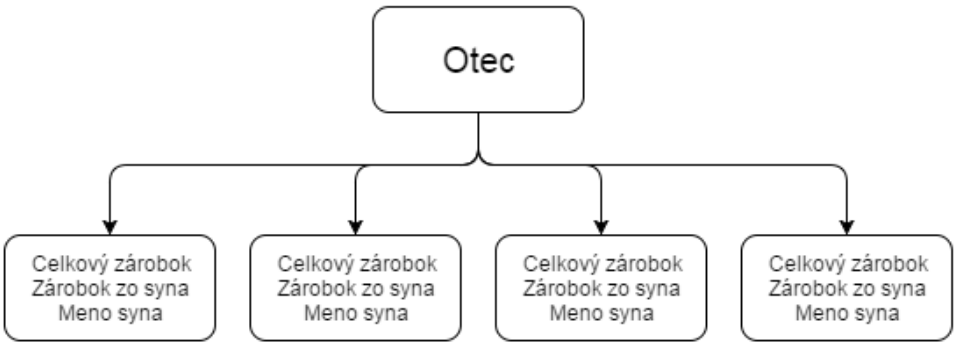
Obr. 3.3: – Stavový diagram systému, vytvorené v online službe: www.draw.io

Stavový diagram znázorňuje: Najprv sa podľa otca načítajú len jeho synovia z databázy do pamäte a jediný vykreslený je samotný otec. Následne po kliknutí na otca sa zavolá funkcia, ktorá dávkuje jeho synov. Zobrazí najprv prvých desať a pri prepnutí sa uloží stav otvoreného otca a vygeneruje sa nanovo strom s ďaľšími jeho desiatimi synami, ak ich má. V prípade že nie, prepne sa na prvých desať synov. Po kliknutí na jedinca ktorý už má roztvorených synov sa musí zavolať ich deštrukcia spolu s deštrukciou ich synov do hĺbky.

3.7 Analýza používateľského rozhrania aplikácie pre používateľa

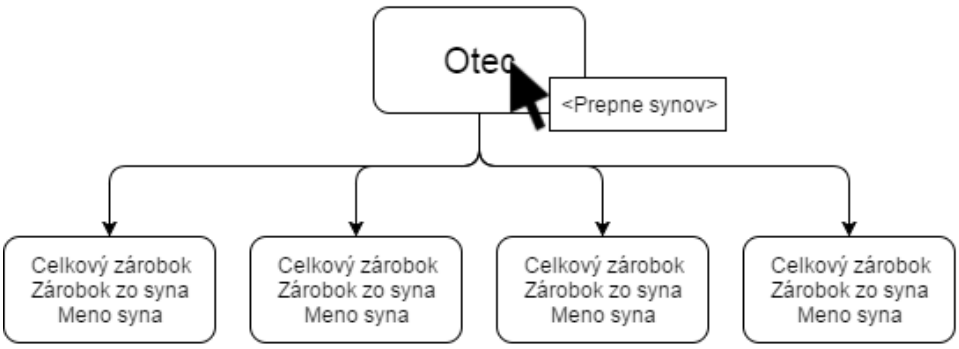
Táto časť definuje, ako aplikácia vyzerá z pohľadu používateľa. Taktiež neexistuje administrátor ani náhľad pre osobu ktorá nie je zaregistrovaný používateľ.

3.7.1 Vzhľad grafu



Obr. 3.4: –vizuálne zobrazenie grafu, vytvorené v online službe: www.draw.io

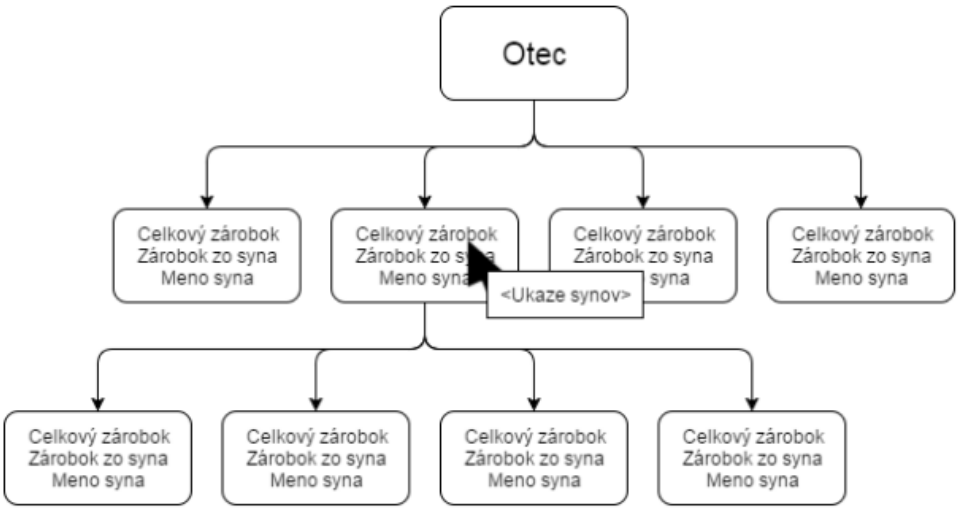
Základný vzhľad grafu je otec (prihlásený používateľ), ktorý prezerá svojich synov a celkový zárobok (z neho a jeho synov a ich synov rekurzívne) ale aj zárobok z jednotlivého syna.



Obr. 3.5: – vizuálne zobrazenie činnosti prepnutia synov otca, vytvorené v online službe: www.draw.io

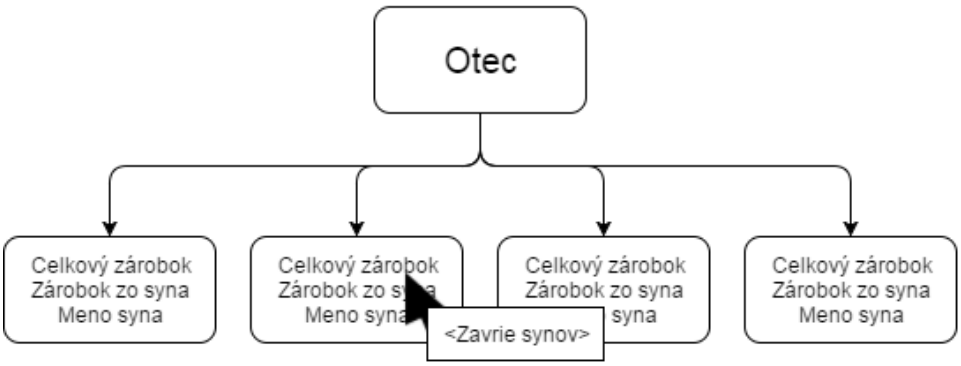
Pri pravom kliknutí na otca alebo syna (ktorý je otcom svojich synov) sa prepne na ďalších desať (alebo vopred nastavený počet) synov, ak ich má.

Ak už ďalších synov nemá, prepne na prvých desať synov.



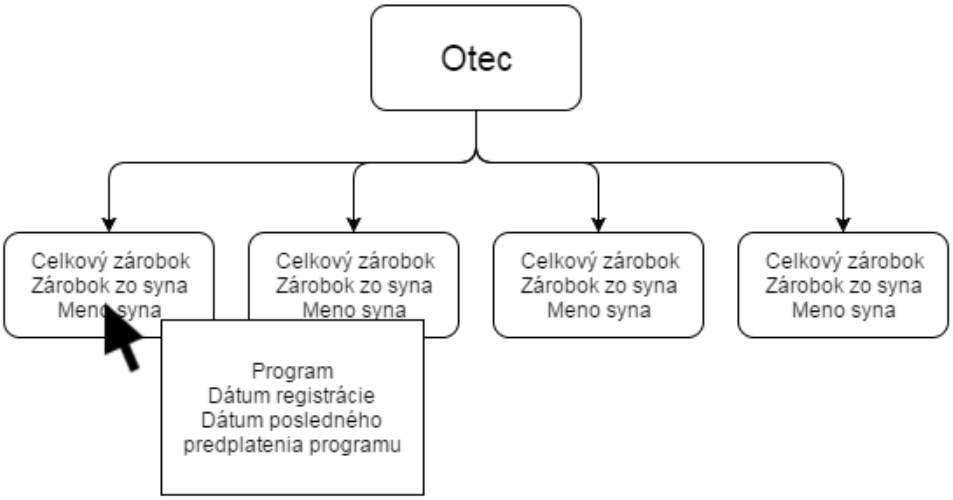
Obr. 3.6: – vizuálne zobrazenie činnosti otvorenia synov otca, vytvorené v online službe: www.draw.io

Po ľavom kliknutí na otca sa zobrazia jeho synovia a identické údaje ako v prvom grafe.



Obr. 3.7: – vizuálne zobrazenie činnosti zatvorenia synov otca, vytvorené v online službe: www.draw.io

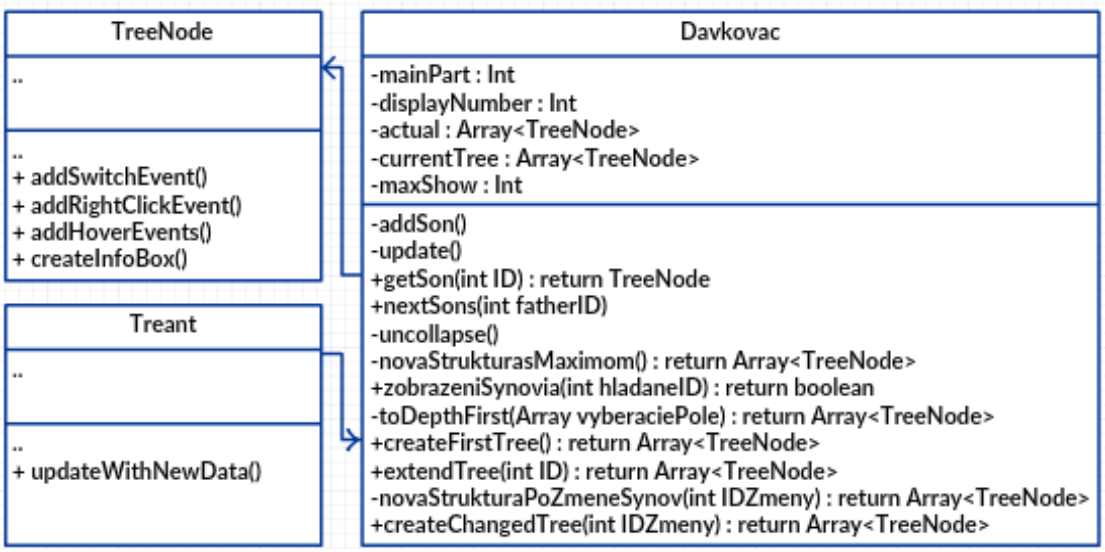
Keď používateľ klikne ľavým tlačíkom myši na už otvoreného syna, zavrie sa zobrazenie všetkých jeho synov a ich synov rekurzívne.



Obr. 3.8: – vizuálne zobrazenie činnosti hoover na získanie dodatočných informácii o synovi, vytvorené v online službe: www.draw.io

Pri prejdení myšou nad syna sa zobrazí dočasné okno s ďaľšími podrobnostami ohľadom syna.

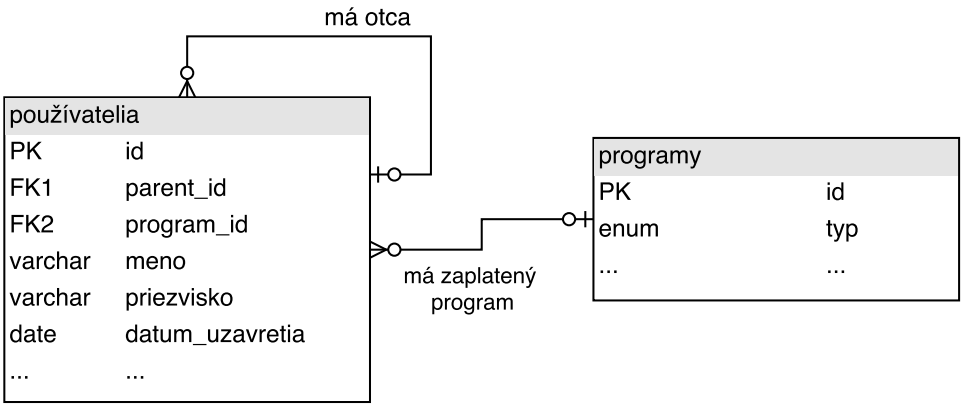
3.8 Triedny diagram



Obr. 3.9: – Triedny diagram systému, vytvorené v online službe: www.draw.io

Tento triedny diagram zobrazuje konkrétne triedy, ktoré budú implementované v našej aplikácii. Obsahuje všetky potrebné premenné a metódy potrebné na funkcionalitu aplikácie. Niektoré triedne polia a metódy boli vynechané z diagramu, lebo nie sú predmetom návrhu.

3.9 Data model



Obr. 3.10: – Data model diagram systému, vytvorené v online službe: www.draw.io

Trieda Dávkovač aj všetky generátory náhodných stromov si uchovávajú informácie iba v operačnej pamäti, a nepotrebuju ukladať žiadne dáta medzi spusteniami. To znamená, že prídavok do systému, ktorý je predmetom tejto záverečnej správy, nevyžaduje ani nepoužíva žiadne úložisko perzistentných dát.

Dávkovač získava vstupné dáta z databázy používateľov, ktorá je externou súčasťou projektu a ktorej väčšina tabuliek je mimo rámec tejto správy. Diagram tabuliek, ktoré priamo ovplyvňujú projekt, teda tabuliek „používateľ" a „program", je zobrazený nižšie, pre lepšie pochopenie formátu, v akom dávkovač získava dáta z databázy.

Skutočný stav databázy – názvy tabuliek, názvy stĺpcov, non-null atribúty stĺpcov, atď. – sa môžu líšiť od tohto diagramu. Diagram zachytáva iba rámcový výzor databázy.

Tabuľka „používatelia" znázorňuje používateľov výslednej aplikácie. Títo majú atribúty ako meno a priezvisko, dátum uzavretia zmluvy, dátum vypršania zmluvy, apod. Môžu mať naviazaný na seba platobný program. Môžu mať priradeného až jedného otca.

Tabuľka „programy" znázorňuje platobný program. Sú iba 4 platobné programy: Bronze, Silver, Gold a VIP.

3.10 Object design

Dávkovacia trieda Davkovac:

- int: mainPart
  - id hlavného otca zo vstupu.
- int: clicked
  - aktuálny kliknutý prvok v strome.
- int: displayNumber
  - maximálny počet zobrazených synov jedného otca v jednej úrovni.
- boolean: actual
  - hovorí o tom, či je strom aktualizovaný.
- array: parts =
  - tu sú uložené všetky prvky stromu v asociatívnom poli podľa ID.
- array: currentTree
  - aktuálne zobrazovaný strom.
- int: maxShow
  - hranica načítaných prvkov stromu pri prvom volaní (po úrovniach, vždy dokončí úroveň...) optimalizčná premenná.
- config
  - konfiguračná premenná pre Treant.
- addSon(myID, fatherID, meno, zarobok, datumUz, prog, datumKon):
  - Pridá syna do dávkovača a štruktúry, ktorá sa stane neaktuálnou.
- update():
  - Táto metóda sa volá autmaticky pri vytvorení stromu, aktualizuje vzťahy medzi synmi.
- getSon(id):
  - Metóda po zavolaní vráti syna s požadovaným ID.
- novaStrukturasMaximom():
  - Metóda sa používa výhradne vnútorne, lebo túto funkciu si volá vo svojom tele metóda createFirstTree, vytiahne synov od hlavného otca po limit maxShow.
- zobrazeniSynovia(hladaneID):
  - Vráti true, ak sú zobrazení synovia osoby s daným ID, hladaneID.
- toDepthFirst (vyberaciePole):
  - Zoradí synov v premennej vyberaciePole podľa štruktúry depth first.

- createFirstTree():
  - Volá sa pri konštrukcii v treant.js. Je to strom do takej hĺbky, ktorá je tesne nad maxShow prvkov.

- extendTree(id):
  - Pri leftclicku na syna ktorý nemá načítaných synov, vracia nový strom aj s jeho synmi.

- novaStrukturaPoZmeneSynov(IDZmeny):
  - Metóda sa používa výhradne vnútorne, lebo túto funkciu si volá createChangedTree, vytiahne od hlavného otca všetkých synov ktorí boli už predtým zobrazení.

- createChangedTree(IDZmeny):
  - Volá sa, keď sa prepnú synovia otca s IDZmeny... ID musí byť ID v dávkovači!

Tree: Trieda Tree je trieda knižnice Treant.js, ktorej dve metódy budú modifikované, aby lepšie spĺňala požiadavky zadania:

- dávkovac: Davkovac
  - Pole bude nastavené konštruktore triedy. Obsahuje dávkovací objekt, ktorý abstrahuje od vyberania používateľských dát z databázy a podáva synov ľubovoľného vrchola po dávkach maximálne desiatich synov naraz.

- addHoverEvents(nodeEl): void
  - Callback funkcia eventu "onHover". Položka nad ktorú sme nadišli myšou zobrazí dodatočné data.

- addRightClickEvents(nodeEl): void
  - Callback funkcia eventu "onRightClick". Synovia otca, na ktorého bolo kliknuté, sa nahradia za novú dávku, maximálne desiatich synov, a následne sa strom prekreslí.

Kapitola 4

Inštalácia a spustenie

4.1 Inštalácia pre weby

Inštalácia zahŕňa jednoduché kopírovanie priečinka "`scr`" na server, kde bude internetová stránka bežať. Pre zmenu zobrazovaného stromu je potrebné zmeniť pole `config` v súbore `index.html`, pomocou ktorého sa stránka aj spúšťa.

4.2 Inštalácia pre mobilnú aplikáciu

- Je potrebné stiahnuť si Node.js napríklad na stránke: `https://nodejs.org/en/download/`.
- Inštalácia Cordovy pomocou jedného z nasledujúcich príazov (Podľa operačného systému):
  - OS X and Linux: `$ sudo npm install -g cordova`
  - Windows: `C:\>npm install -g cordova`
- Teraz je potrebné vytvorenie priečinka , v ktorom bude zdrojový kód udržovaný.
  `$ cordova create hello com.example.hello HelloWorld`
- Všetky potrebné príkazy je potrebné spúšťať v projektovom priečinku.
  `$ cd hello`
- Pre pridanie platformy ios:
  `$ cordova platform add ios -save`
- Pre pridanie platformy Android:
  `$ cordova platform add android -save`
- Nasledujúcim príkazom skontrolujeme či sú všetky požadované časti nainštalované:

```
$ cordova requirements
Requirements check results for android:
Java JDK: installed .
Android SDK: installed
Android target: installed android-19,android-21,android-22,android-23,Google Inc.:Google APIs:19,Go
Gradle: installed

Requirements check results for ios:
Apple OS X: not installed
Cordova tooling for iOS requires Apple OS X
Error: Some of requirements check failed
```

- Následne aplikáciu zbuildujeme pre všetky platformy:
  `$ cordova build`
- Alebo iba pre konkrétne:
  `$ cordova build android`
- Nakoniec aplikáciu spustíme:
  `$ cordova emulate android`

Obšírnejší návod na inštaláciu môžete nájsť na stránke:
`https://cordova.apache.org/docs/en/latest/guide/cli/index.html`.