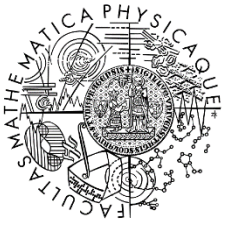


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Marián Dolník

Správa dát fariem

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D.

Studijní program: Informatika

Studijní obor: Programování a SW systémy

Praha 2019

Chcel by som poďakovať RNDr. Janovi Kofroňovi, Ph.D. za poskytnuté rady a čas, ktorý mi venoval pri písaní tejto práce. Ďakujem tiež rodičom, ktorí boli pre mňa oporou počas štúdia na vysokej škole, a mojej priateľke Martine Mirtovej, ktorá mi poskytovala spätnú väzbu pri písaní tejto práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V …..... dne..... Podpis autora

Názov práce: Správa dát fariem

Autor: Marián Dolník

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedúci bakalárskej práce: RNDr. Jan Kofroň, Ph.D., Katedra distribuovaných a spolehlivých systémů

Abstrakt: Cieľom práce je navrhnúť a implementovať softvér pre uľahčenie organizácie práce na farme spôsobom zaznamenávania údajov o zberačoch, ich odovzdávkových dátach a následným spracovaním nazbieraných údajov.

Softvérový projekt pozostáva z troch častí a to z mobilnej a webovej aplikácie a synchronizačnej jednotky.

Úlohou synchronizačnej jednotky je synchronizácia lokálnej mobilnej databázy s hlavnou databázou aplikácie.

Mobilná časť má za úlohu zbierať dáta. Zber týchto dát je vykonávaný za pomoci UI rozhrania, kde je možné dáta definovať (výberom hodnôt povinných atribútov). Mimo iné, mobilná časť aplikácie umožňuje aj aktuálne pridané dáta spravovať (filtrovanie pomocou viacerých parametrov, odstraňovanie, zoskupovanie za účelom štatistiky, synchronizácia dát s hlavnou databázou).

Webová časť zobrazuje a spravuje nazbierané dáta. Vytvára štatistické záznamy o výkonnosti jednotlivých pracovníkov a o úrodnosti jednotlivých polí. Definuje rôzne prístupové práva, podľa ktorých majú užívatelia právo na editáciu jednotlivých zložiek aplikácie, poprípade môžu do aplikácie nahliadať iba v read-only móde.

Kľúčové slová: aplikácia, Android, web, správa dát

Title: Farm data management

Author: Marián Dolník

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D., Department of Distributed and Dependable Systems

Abstract: The purpose of this bachelor is to design and implement an application which would simplify work organization on a farm by recording information about pickers, their logging data, and by further processing of these data.

This software project consists of a mobile and a web applications and a synchronization unit.

The purpose of the synchronization unit is to synchronize data of the local mobile database with the data from the main database.

The mobile application has the purpose of storing the data. Collecting the data is executed by user interface, allowing one to define data by choosing a value of the required attributes. Among other functions, the mobile application allows the user to work with already existing data (filter data by multiple conditions, remove data, collect data in order to obtain statistics, synchronize data with the main database).

The web application displays and manages worker's data. It creates the statistics about worker's efficiency and field's fertility. It defines access rights, which provides the users with access to specific parts of the web application, optionally allowing them to access the data in read-only mode.

Keywords: application, Android, web, data administration

Obsah

1 Úvod

Každoročne študenti odchádzajú na letnú brigádu, aby pokryli svoje náklady na štúdium. Jednou z vyhľadávaných prác je sezónny zber ovocia alebo zeleniny na farmách, kde je pracovník platený od množstva nazbieraných plodov.

Aj napriek vymoženostiam modernej techniky sa stále stretávame s tým, že veľké množstvo poľnohospodárskych podnikov nespravuje svoje dáta v digitálnej podobe. To môže mať niekoľko nevýhod ako zo strany vedenia, tak aj zo strany jednotlivých pracovníkov. V prípade, že na uchovávanie dát je stále využívaný spôsob zaznamenávania údajov na papier, môže ľahko dochádzať k strate alebo degradácii informácií rôznymi faktormi. Uchovávanie informácií v tejto podobe je veľmi nesystematické a neprehľadné a zároveň zbytočne vo veľkom množstve plytvá neudržateľnými zdrojmi. Ďalšou nevýhodou je, že jednotliví pracovníci nemajú prístup k informáciám týkajúcich sa ich pracovného výkonu, a tak nemajú možnosť kontroly ich zárobku.

Vytvorením aplikácie pre uchovávanie dát v digitálnej podobe uľahčíme správu dát, čím predídeme potrebe ich ukladania prostredníctvom papierovej formy, ktorá je v dnešnej dobe značne neefektívna. Zároveň tak sprístupníme jednotlivým pracovníkom dáta o ich pracovnom výkone, z ktorého následne môžu kontrolovať svoje zisky počítané z odovzdávkových dát.

Ciele

Cieľom práce je navrhnúť a implementovať aplikáciu na spravovanie dát sezónnych pracovníkov. Táto aplikácia má pomôcť zjednodušiť organizáciu práce s dátami pracovníkov na farme a udržiavať všetky potrebné informácie a dáta v digitálnej podobe. Výsledná aplikácia by mala umožňovať:

- vytváranie, editáciu a odstraňovanie pracovníkov,
- vytváranie, editáciu a odstraňovanie dát reprezentujúcich jednotlivé odovzdávkové dáta pracovníkov,
- zobrazovanie štatistických informácií o výkonnosti jednotlivých pracovníkov a o úrodnosti polí,
- ohodnocovanie polí (výška zárobku za jednotku na danom poli).

Popis jednotlivých komponentov aplikácie spolu s členením na jednotlivé časti popíšeme neskôr v texte.

Štruktúra práce

Text tejto práce je členený do šiestich kapitol. V úvode vymedzujeme ciele našej práce spolu s rozhodnutím, ktoré nás viedlo k vytvoreniu aplikácie týkajúcej sa správy dát fariem. V druhej kapitole analyzujeme jednotlivé rozhodnutia, ktoré sme museli vykonať pred samotným návrhom a ktoré majú priamy vplyv na špecifikovanie použitých technológií a zároveň obmedzenia výslednej aplikácie. V tretej, štvrtej a piatej kapitole postupne približujeme výsledný návrh mobilnej aplikácie, synchronizačnej jednotky a webovej aplikácie z pohľadu štruktúry jednotlivých aplikácií. V závere definujeme výsledky implementácií spolu s prostrediami, na ktorých prebiehal proces testovania funkčnosti. Na záver pripájame prílohy definujúce detailný popis práce s jednotlivými aplikáciami, ktoré slúžia ako ich užívateľská dokumentácia.

2 Analýza

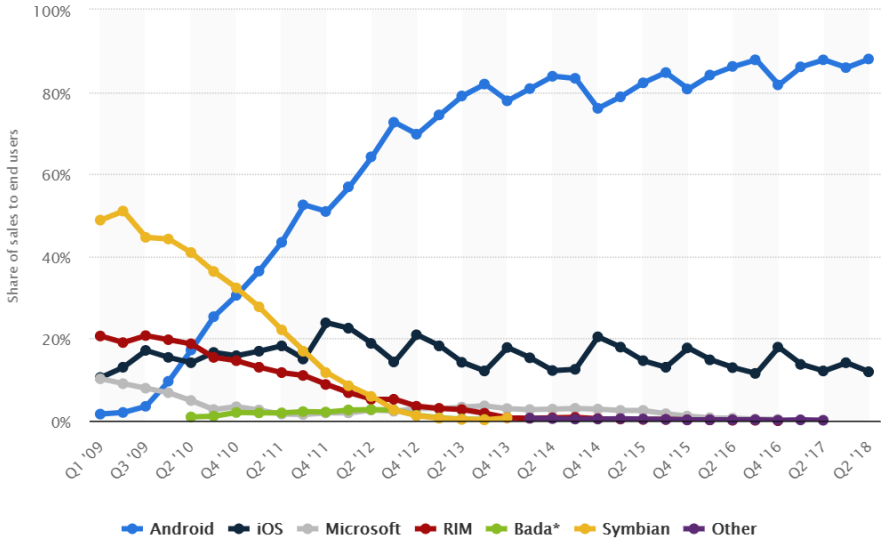
V tejto kapitole sa nachádza rozpis jednotlivých častí aplikácie spolu s popisom rozhodnutí, ktoré sme museli vykonať pri vývoji. V kapitole budú postupne zmienené dôvody výberu jednotlivých riešení, poprípade nevyhnutných požiadaviek na systém, ktoré musia byť špecifikované.

2.1 Analýza mobilnej aplikácie strawPick

V tejto podkapitole uvedieme rozhodnutia, pomocou ktorých sme sa dopracovali k výslednému popisu, ktorý predstavuje výber použitej platformy, programovacieho jazyka, minimálnej verzie API, dátového úložiska a komponent na definovanie používateľského rozhrania mobilnej aplikácie.

2.1.1 Platforma

Pred vývojom mobilnej aplikácie je potrebné rozhodnúť, pre aké mobilné zariadenia, v závislosti na operačnom systéme, bude mobilná aplikácia určená. V dnešnej dobe máme na výber z viacerých variant, pričom medzi popredné a najviac využívané platformy mobilných zariadení patria Android a iOS. Podľa portálu Statista [1], ktorý sa zaoberá prieskumom trhu a online štatistikou, zmienené platformy pokrývajú v priemere okolo 99.7% všetkých predaných mobilných zariadení za rok 2018. V rámci definovanej percentuálnej miery má Android v porovnaní s iOS značne vyššiu mieru predaja (viz Obrázok 2.1).



Obrázok 2.1: Graf portálu Statista určujúci vývoj predaja mobilných telefónov založený na ich operačnom systéme

Dôležitým faktorom, ktorý výrazne vplýva na výber platformy je cieľová skupina užívateľov. Naša práca je zameraná na sekciu poľnohospodárskej produkcie a teda je nevyhnutné prispôsobiť efektivitu zariadenia i programu na dané prostredie. Je potrebné myslieť na dostatočnú odolnosť zariadenia voči manuálnemu poškodeniu a zároveň by počiatočné náklady na kúpu nemali byť príliš vysoké. Vhodným riešením sa ukazuje výber odolných telefónov, vzhľadom na ich odolnosť a cenu. Keďže tieto zariadenia využívajú platformu Android, rozhodli sme sa aj my pre jej použitie.

Značnou nevýhodou platformy iOS, ktorá nás zároveň utvrdila vo výbere platformy Android je fakt, že iOS potrebuje minimálne v konečnej fáze vývoja počítač s operačným systémom OS X. Na druhej strane pre vývoj Android aplikácie nie je potrebný nijaký špecifický operačný systém, a teda je možné použiť Windows, Linux, OS X či iné [2].

2.1.2 Programovací jazyk

Tak ako pri voľbe platformy, tak aj pri výbere programovacieho jazyka máme viacero možností. Jednou je jazyk C#, ktorý je možné použiť vo frameworku Xamarin. V zmysle určenia efektivity jazyka C# pre vývoj mobilnej aplikácie je potrebné analyzovať klady a zápory frameworku Xamarin. Podstatnou výhodou Xamarinu je multiplatformita vyvíjaných aplikácií, ktorá umožňuje využitie aplikácie viacerými platformami, napríklad Android a iOS. V predošlej sekcii sme však obmedzili spustiteľnosť aplikácie na platformu Android a z tohto hľadiska sa pre nás multiplatformita Xamarinu javí ako nepodstatná. Keďže Xamarin umožňuje vyvíjať aplikácie bez ohľadu na platformu, výsledná aplikácia je pomalšia ako natívna aplikácia, ktorá je vyvíjaná priamo pre Android.

Medzi ďalšie programovacie jazyky sa zaraďujú Java a Kotlin, ktoré sú v súčasnosti oba vedené ako oficiálne programovacie jazyky pre vývoj mobilných aplikácií pre platformu Android. Java je dlhodobo oficiálnym jazykom pre vývoj Android aplikácií, a teda existuje veľké množstvo voľne dostupnej literatúry, čo pre nás predstavuje veľkú výhodu. Týmto spôsobom sa môžeme dostať k množstvu podrobných informácií a širokej škále autorov, ktorých je možné využiť pre porovnanie. Tento fakt zároveň predstavuje i výraznú nevýhodu, ktorá spočíva v tom, že množstvo nekvalifikovaných autorov publikuje svoje názory a šíri tak materiál, ktorý nie je možné aplikovať v praxi. Celý súbor literatúry o Jave teda môže slúžiť ako učebný materiál či už pre začiatočníkov, alebo pokročilých programátorov. Pred

výberom je však nutné dôkladne uvážiť, či je daná literatúra vhodná. Na druhej strane Kotlin je veľmi mladý jazyk, ktorý vyšiel v roku 2016. Dostať sa k voľne dostupnej literatúre nie je také jednoduché ako pri Jave a je teda nutné sa uspokojiť s dokumentáciou a s menším počtom existujúcej literatúry. Kotlin ma oproti Jave kratšiu syntax na definovanie jednotlivých časti kódu, a môže tým ušetriť čas, ktorý je potrebný pri vývoji. Avšak vzhľadom na to, že autor práce jazyk Kotlin neovláda, nedá sa povedať, že jeho skrátená syntax by nám ušetrila čas potrebný pre vývoj aplikácie, keďže by sme museli vložiť viac úsilia do výučby a pochopenia jazyka. Na základe vyššie spomenutých dôvodov a v rámci udržania koherentnosti aplikácií (všetky časti sú písané v jazyku Java) sme sa rozhodli pre výber programovacieho jazyka Java.

2.1.3 Verzia API

Pri vývoji mobilnej aplikácie je potrebné definovať minimálnu úroveň API verzie Androidu, na ktorej bude aplikácia schopná fungovať, a cieľovú úroveň, pre ktorú je aplikácia navrhnutá. Výberom nízkej úrovne API verzie Androidu, získame vysoké percento mobilných zariadení, na ktorých môže byť aplikácia spustená, pripravíme sa však o možnosť využitia nových vlastností, definovaných v novších verziách API. Programátorom, ktorí si pre vývoj svojej aplikácie zvolia nižšiu verziu API, je navrhnutá knižnica *Support Library*, ktorá umožňuje používať ekvivalentné funkcie na starších verziách API. Pri použití spomenutej knižnice ostane funkčnosť programu zhodná aj s nižšími verziami API zariadenia. Vzhľadom na to, že žiadna z použitých knižníc v našej mobilnej aplikácií nedefinuje požiadavky na špeciálne funkcie, ktoré sú pridané v najnovších API verziách, rozhodli sme sa pre výber API verzie 15, s ktorou je kompatibilných najväčší počet Android zariadení, a s ktorou zabezpečíme vysokú pravdepodobnosť spustiteľnosti aplikácie na rôznych mobilných zariadeniach. Na základe informácií, ktoré poukazujú na aktuálny počet zariadení fungujúcich na jednotlivých API verziách, vyplýva, že okolo 99,8 % zariadení je kompatibilných s nami vybranou API verziou [3] (viz Obrázok 2.2).

Android Platform Version	API Level	Devices
4.0 Ice Cream Sandwich	15	0.3%
4.1 Jelly Bean	16	1.1%
4.2 Jelly Bean	17	1.5%
4.3 Jelly Bean	18	0.4%
4.4 KitKat	19	7.6%
5.0 Lollipop	21	3.5%
5.1 Lollipop	22	14.4%
6.0 Marshmallow	23	21.3%
7.0 Nougat	24	18.1%
7.1 Nougat	25	10.1%
8.0 Oreo	26	14.0%
8.1 Oreo	27	7.5%

Obrázok 2.2: Počet zariadení bežiacich na danej úrovni API podľa prístupu na Google Play v období písania práce

2.1.4 Ukladanie dát

Hlavnou úlohou mobilnej aplikácie je vytváranie, editácia, odstraňovanie a následná synchronizácia dát s hlavným dátovým úložiskom. S ohľadom na tieto funkcie je potrebné definovať akým spôsobom sa budú všetky potrebné dáta aplikácie v zariadení uchovávať. Dáta aplikácie môžeme na základe ich funkčnosti rozdeliť do dvoch kategórií. Na jednej strane sú to konfiguračné dáta, ktoré zabezpečujú korektný chod aplikácie a na druhej strane odovzdávkové dáta, ktoré reprezentujú výkon jednotlivých pracovníkov.

Konfiguračné dáta majú výzor dvojice kľúč-hodnota a je nevyhnutné, aby pri každom načítaní aplikácie boli prítomné. Odvíjajúc sa od týchto vlastností a požiadaviek je potrebné, aby boli dáta uskladnené na mieste, na ktoré má aplikácia za akýchkoľvek okolností prístup. Z tohto dôvodu sa ako najlepšia možnosť ukladania konfiguračných dát javí použitie *SharedPreferences*, poprípade ukladanie dát do súboru na internom úložisku zariadenia. *SharedPreferences* objekt umožňuje čítanie a editáciu dvojíc kľúč-hodnota pomocou jednoduchých metód, a je tak pre nás lepším

riešením, ako si tento existujúci model ukladania dvojíc kľúč-hodnota znovu implementovať za použitia súboru na internom úložisku.

Odovzdávkové dáta držia oproti konfiguračným dátam širšiu škálu informácií a zároveň množstvo uložených odovzdávkových dát bude podstatne rozsiahlejšie ako pri konfiguračných dátach. Na ukladanie takýchto dát nám Android ponúka interný, poprípade externý súborový systém a vstavaný SQLite relačný databázový systém. Na základe charakteristiky aplikácie nie je použitie externého úložiska dobrou voľbou, vzhľadom na to, že externé úložisko nemusí byť vždy k zariadeniu pripojené. Pri výbere ukladania dát pomocou súborov do interného úložiska by bolo potrebné implementovať sadu tried, ktoré budú slúžiť na serializáciu a následnú deserializáciu objektov, pomocou ktorých je umožnené zapisovanie a načítavanie objektov zo súboru. Pri použití SQLite by sme sa týmto implementačným častiam vyhli a zároveň by sme mali možnosť pristupovať k uchovávaným dátam rýchlejším spôsobom.[1] Na základe vyššie spomenutých aspektov sme sa pre ukladanie odovzdávkových dát rozhodli použiť SQLite.

2.1.5 Návrh používateľského rozhrania

Používateľské rozhranie mobilnej aplikácie je realizované pomocou definovania jednotlivých elementov v XML súbore. Hlavným stavebným prvkom, ktorý je v aplikácii použitý je *constraint layout*, pomocou ktorého môžeme jednotlivé komponenty relatívne polohovať voči polohám ostatných elementov [5]. *Constraint layout* taktiež umožňuje definovať percentuálne rozdelenie stránky pomocou tzv. *guidelines*. To nám umožní vytvárať responzívny dizajn aplikácie. Rovnakého výsledku by sme mohli dosiahnuť aj použitím iných stavebných prvkov, ako napríklad *linear layout*, avšak definovaním percentuálneho rozdelenia stránky na viacerých úrovniach vnorených *linear layout* elementov bude výpočet exponenciálny s ohľadom na počet vnorení [6]. Na definovanie elementov, ktoré sa môžu dynamicky meniť počas chodu aplikácie, je použitá ich inicializácia za behu programu.

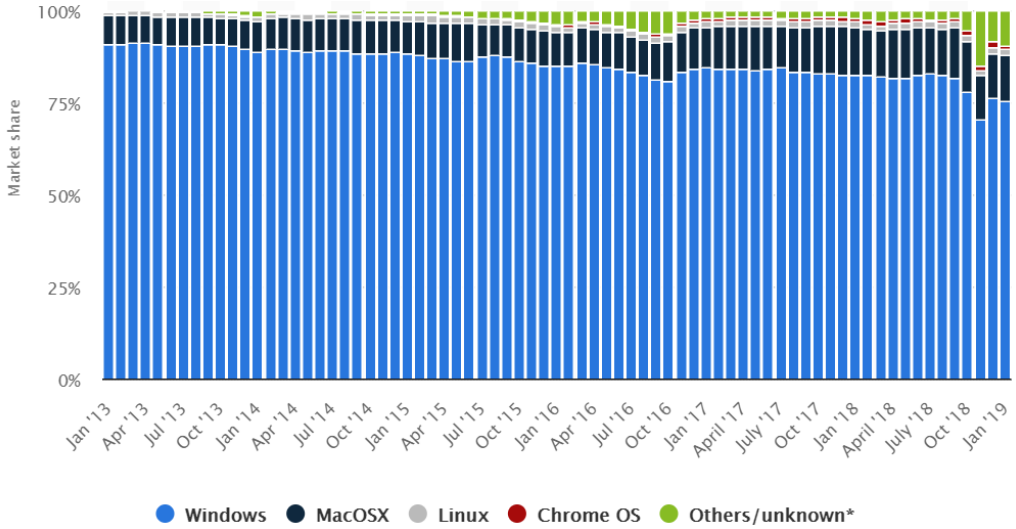
[1] Podľa informácií o SQLite relačnom databázovom systéme v zariadení Android platí, že SQLite umožňuje čítať a zapisovať dáta rýchlejšie ako metódy *read* a *write* pre prácu so súborom [4].

2.2 Analýza synchronizačnej jednotky strawSync

V tejto podkapitole sa budeme venovať výberu jednotlivých platforiem a programovacieho jazyka. Následne rozanalyzujeme možnosti, ktoré sme mali na výber s cieľom získania konfiguračných atribútov zo súboru. Taktiež si priblížime možnosti, ktoré sa ponúkajú na ukladanie dát získaných z mobilnej a webovej aplikácie.

2.2.1 Platforma

Medzi tri najvyužívanejšie operačné systémy sa dnes zaraďujú Windows, Mac OS a Unix (viz Obrázok 2.3).



Obrázok 2.3: Graf portálu Statista určujúci vývoj predaja desktopových počítačov založený na ich operačnom systéme [7]

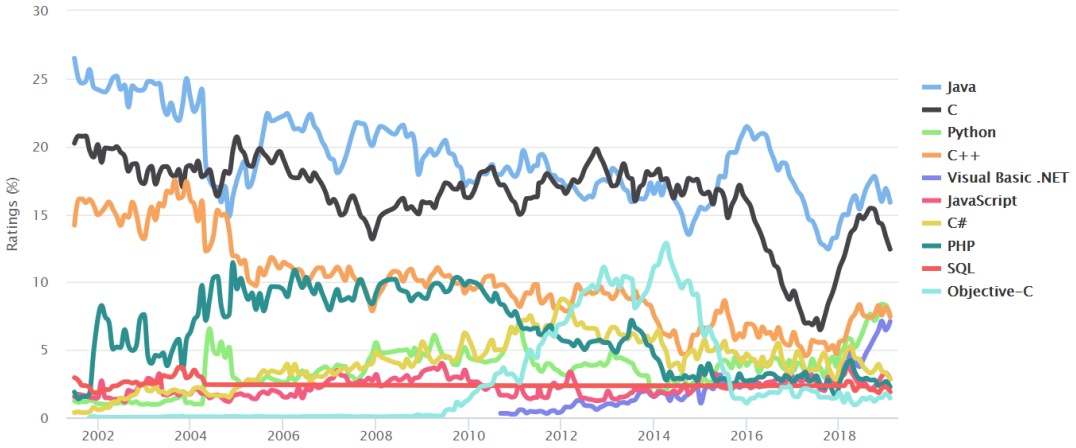
Ako môžeme vidieť na grafe, Windows sa dlhoročne drží na prvej priečke, a teda spustiteľnosť aplikácie na zariadeniach s operačným systémom Windows je z tohto hľadiska nevyhnutné. Zároveň však platí, že množstvo aplikácií, ktoré využívajú MySQL relačný databázový systém, s ktorým v aplikácii pracujeme aj my, beží práve na operačnom systéme Linux vzhľadom na jeho natívnu implementáciu MySQL systému. Ďalšou výhodou, ktorú Linux poskytuje, je mimo iné aj to, že je vyvíjaný ako open-source softvér.

Vzhľadom na charakter aplikácie, je nutné podotknúť, že aplikácia nemusí byť obmedzená na použitie konkrétnej platformy. Keby sme spustiteľnosť programu obmedzili iba na určitú platformu, zabránili by sme užívateľom možnosti prechodu medzi nimi, a tým by sme užívateľom taktiež odobrali možnosť používať špecifické vlastností jednotlivých platforiem, ktoré môžu byť pre užívateľa prínosné. Preto sme

rozhodli, že aplikáciu nebudeme obmedzovať platformou systému, na ktorom môže byť spustená.

2.2.2 Programovací jazyk

Výber programovacie jazyka je o niečo zložitejší, pretože v dnešnej dobe existuje množstvo programovacích jazykov, ktoré majú rôzne výhody aj nevýhody, pričom nie je možné objektívne určiť, ktorý jazyk je lepší. Jazyky, medzi ktorými sme sa primárne rozhodovali sú Java a C#, vzhľadom na to, že autor práce oba jazyky ovláda na pokročilej úrovni. Podľa autorovho názoru neexistuje dostatočný argument, ktorý by mohol jeden z uvedených jazykov klásť do popredia. Výhody jednotlivých jazykov môžu byť rôzne, v závislosti od typu vyvíjanej aplikácie. Vzhľadom na to, že požadujeme spustiteľnosť aplikácie na rôznych platformách, je nevyhnutné, aby sme sa zamerali primárne na túto vlastnosť. Pre aplikácie, ktoré sú multiplatformné, je vo väčšine prípadov používaný jazyk Java. C# taktiež umožňuje vyvíjanie multiplatformných aplikácií, avšak za použitia softvérovej platformy Mono. Táto platforma je vyvíjaná treťou stranou, a teda otázka kompatibility nemusí byť vždy zaručená [8]. S ohľadom na tieto skutočnosti, je pre našu aplikáciu najvýhodnejšie vybrať jazyk Java, keďže výsledný program bude v tomto zmysle kompatibilný s každou platformou. Zároveň sa nebudeme musieť spoliehať na riešenia tretej strany. Pri výbere jazyku Java sme rozhodli, že budeme aplikáciu vyvíjať prostredníctvom verzie Java 8. Na záver pridávame ešte štatistiku obľúbenosti jednotlivých jazykov na základe tiobe indexu spoločnosti Tiobe, ktorý meria popularitu jednotlivých programovacích jazykov [9] (viz Obrázok 2.4, Obrázok 2.5).



Obrázok 2.4: Graf určujúci popularitu programovacích jazykov podľa roku spracovaný spoločnosťou Tiobe

Feb 2019	Feb 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.876%	+0.89%
2	2		C	12.424%	+0.57%
3	4	▲	Python	7.574%	+2.41%
4	3	▼	C++	7.444%	+1.72%
5	6	▲	Visual Basic .NET	7.095%	+3.02%
6	8	▲	JavaScript	2.848%	-0.32%
7	5	▼	C#	2.846%	-1.61%
8	7	▼	PHP	2.271%	-1.15%
9	11	▲	SQL	1.900%	-0.46%
10	20	⇈	Objective-C	1.447%	+0.32%

Obrázok 2.5: Percentuálny nárast, pokles popularity programovacích jazykov

2.2.3 Načítanie vstupu

Synchronizačná jednotka slúži primárne k zabezpečeniu prenosu dát medzi mobilnou aplikáciou a hlavnou databázou. Pre zaistenie korektného chodu aplikácie je potrebné inicializovať všetky povinné atribúty, ktoré sa podieľajú na definovaní vytváraného spojenia. Vzhľadom na tento fakt je potrebné konfiguračné atribúty spolu s ich hodnotami z konfiguračného súboru správne načítať. Jednou z možností ako postupovať je vytvoriť si vlastnú triedu, ktorá by tento proces mala na starosti. Druhou možnosťou je použiť niektorú z už existujúcich tried, ktoré sú k danému zámeru určené. Jednou z nich je trieda *Properties* [10], ktorá reprezentuje sadu vlastností zapísaných, respektíve načítaných zo súboru. Práca s danou triedou je veľmi jednoduchá a jej použitím ušetríme čas, potrebný pre vytvorenie vlastnej triedy na načítanie konfiguračných atribútov.

2.2.4 Ukladanie dát

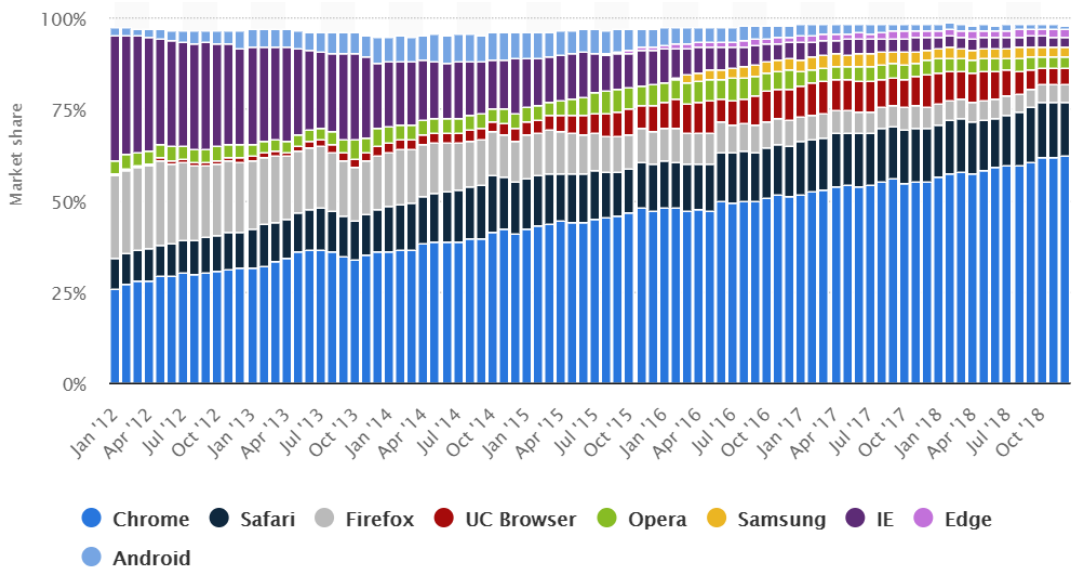
Ukladanie dát získaných aplikáciami je realizované pomocou MySQL relačného databázového systému. Hlavné výhody MySQL vidíme v jednoduchosti jeho použitia a nulových obstarávacích nákladoch. Vzhľadom na veľkosť našej aplikácie je tak vhodným riešením na otázku ukladania dát. MySQL databáza si udržuje svoju vlastnú vyrovnávaciu pamäť, v ktorej uchováva tzv. skompilované procedúry, pomocou ktorých zvyšuje svoju výkonnosť pri opakovanom použití danej procedúry. V závislosti na popisovanej vlastnosti je ale nutné používať skompilované procedúry s rozvahou. V prípade že by sme tak nevykonali, pamäťové vyťaženie jednotlivých spojení by rýchlo narastalo a výhoda tejto vlastnosti by sa ľahko stala jej nevýhodou.

2.3 Analýza webovej aplikácie strawManager

V tejto podkapitole sa budeme zameriavať na existujúce technológie, ktoré môžeme pri vývoji webovej aplikácie využiť. Postupne budeme analyzovať jednotlivé webové prehliadače, a rozhodneme pre ktoré je aplikácia primárne určená. Rozoberieme použitie a funkčnosť jednotlivých technológií, ktoré sú aplikovateľné pri vytváraní Front-endu a Back-endu aplikácie.

2.3.1 Webové prehliadače

Aby sme boli schopní zobrazovať a vykonávať interakciu s HTML dokumentami webových serverov, je potrebné použiť aplikačný softvér, ktorý je pre túto funkciu vytvorený. Existuje veľké množstvo webových prehliadačov, ktoré sa snažia poskytnúť užívateľom čo najlepšie prostredie a efektivitu. Medzi najznámejšie a najviac využívane webové prehliadače sa zaraďujú: Google Chrome, Safari, Internet Explorer, Firerox, Opera a mnoho ďalších (viz Obrázok 2.6).



Obrázok 2.6: Graf portálu Statista určujúci popularitu webových prehliadačov [11]

Každý zo spomenutých prehliadačov nám ponúka rôzne možnosti. Niektoré sa snažia byť čo najjednoduchšie a zobrazovať užívateľom iba webovú stránku, niektoré naopak ponúkajú rôzne navigačné menu, spolu s ďalšími funkciami, ktoré majú užívateľom uľahčiť prácu. Preto je veľmi ťažké objektívne rozhodnúť, ktorý zo spomenutých webových prehliadačov je pre našu aplikáciu najvhodnejší. Každý webový prehliadač si môže získať priazeň užívateľov svojou funkčnosťou, ktorú iní môžu považovať za negatívnu a zbytočnú. V tomto zmysle sme sa rozhodli pre výber webového

prehliadača, ktorý je medzi užívateľmi najpopulárnejší. Tým získame najväčšiu pravdepodobnosť, že nami podporovaný webový prehliadač bude v preferencii užívateľov. Ako môžeme vidieť na grafe popularity jednotlivých webových prehliadačov, je zrejmé, že dlhodobo najúspešnejším prehliadačom je Google Chrome, ktorý bude zároveň podporovaným prehliadačom našej vyvíjanej webovej aplikácie. Výberom tohto webového prehliadača zároveň nevylučujeme možnosť použitia ostatných prehliadačov na zobrazovanie HTML dokumentov webovej aplikácie, avšak v tomto prípade nezaručujeme funkčnosť, poprípade vizuálnu konzistenciu aplikácie.

2.3.2 Front-end technológie

Medzi základné a najpoužívanejšie Front-endové technológie, ktoré slúžia na vývoj užívateľského rozhrania výslednej aplikácie patrí HTML, CSS a JavaScript. Každá zo spomenutých technológií hrá určitú úlohu pri definovaní výzoru webovej aplikácie. HTML je značkovací jazyk, ktorý obohacuje výzor stránky, umožňuje pridávanie dodatočných informácií o význame jednotlivých komponent, z ktorých sa stránka skladá. Variantom k HTML je jazyk XHTML, ktorý neprináša žiadne nové rozšírenia oproti HTML a pre našu aplikáciu nepredstavuje teda nijaký prínos. Pomocou CSS môžeme následne vytvoreným prvkom pridávať doplňujúce vlastnosti, ktoré definujú ich umiestnenie, vzhľad a iné charakteristiky. Týmto spôsobom definuje, ako sa má obsah stránky zobrazovať. Trojicu neoddeliteľných súčastí modernej webovej aplikácie uzatvára JavaScript, ktorý umožňuje vytvárať dynamický obsah stránky a je teda nevyhnutný pre vytvorenie webovej aplikácie. JavaScript a CSS sú jediné svojho typu, ktoré umožňujú vykonávanie vyššie spomenutých funkcií.

2.3.3 Back-end technológie

Pri vývoji webovej aplikácie je nutné vybrať technológiu, ktorá bude tvoriť Back-end aplikácie. Na výber máme z niekoľkých rôznych variant ako napríklad Java, PHP, ASP.NET a ďalšie. Rovnako ako pri výbere programovacieho jazyka synchronizačnej jednotky, tak ani tu neexistuje objektívne rozhodnutie o tom, ktorá z technológií je lepšia. Preto sa zameriame na špecifiká, ktoré sú pre nás esenciálne a pri výbere na ne budeme prihliadať. Hlavným rozdielom medzi vyššie spomínanými technológiami je možnosť variabilnosti platformy serveru, na ktorej webová aplikácia beží. Pre PHP a Javu platí, že nie sú nijak obmedzené platformou serveru, a teda môžu byť aplikované na operačných systémoch Windows, Linux, Mac OS a ďalších. Toto ale

pre aplikáciu vyvíjanú pomocou technológie ASP.NET neplatí. Pokiaľ by sme sa rozhodli pre technológiu ASP.NET, obmedzili by sme platformu serveru, na ktorom môže aplikácia fungovať iba na použitie operačného systému Windows. Je taktiež možné použiť softvérovú platformu Mono, aby sme obišli obmedzenia na operačný systém pri použití ASP.NET. Avšak rovnako ako v sekcii venovanej platforme mobilnej aplikácie, toto nepovažujeme za najlepšiu možnosť.

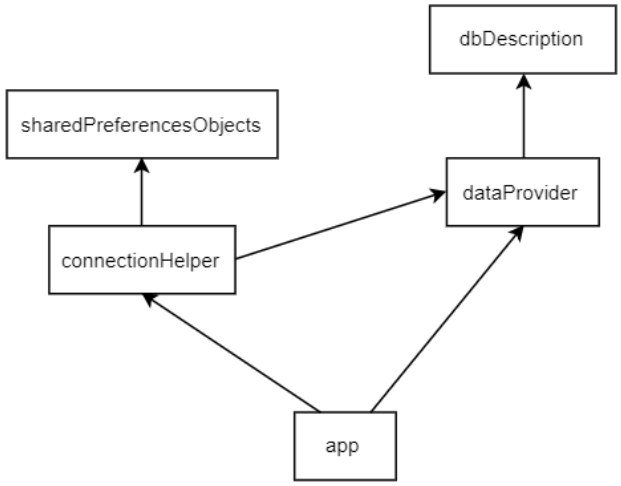
Autor práce za značnú výhodu pri porovnaní Javy a PHP pokladá fakt, že webová aplikácia implementovaná použitím jazyka Java môže byť rýchlejšia, ako tá naprogramovaná v PHP. Dôvodom je, že PHP je skriptovací interpretovaný jazyk a pri každej HTTP požiadavke sa aplikácia znovu interpretuje. Na druhú stranu kód napísaný v Jave je najprv skompilovaný do bite kódu a následne je pomocou JRE interpretovaný. To umožňuje, že pri nasadení na server sa vykoná kompilácia kódu a následne je aplikácia inicializovaná a beží po celú dobu, pokým nie je zo serveru odstránená [12]. Vzhľadom na vyššie spomenuté rozdiely medzi jednotlivými technológiami považujeme pre vývoj našej aplikácie ako najrozumnejšie riešenie použitie programovacieho jazyka Java. Pri výbere tohto jazyka sme rozhodli, že aplikáciu budeme vyvíjať prostredníctvom verzie Java 8.

3 Popis riešenia mobilnej aplikácie strawPick

V tejto kapitole popíšeme navrhnuté riešenie jednotlivých častí mobilnej aplikácie spolu so vzájomnou interoperabilitou tried a taktiež rozoberieme priebeh komunikácie s SQLite relačným databázovým systémom, ku ktorému definujeme schému použitej databázy.

3.1 Štruktúra mobilnej aplikácie

Mobilná aplikácia je definovaná kódom, ktorý je spustiteľný na Android zariadeniach. Kód aplikácie môžeme rozdeliť na jednotlivé logické celky, ktoré umožňujú udržanie podobných tematických časti aplikácie na jednom ucelenom mieste. Nasledujúci obrázok definuje teoretickú štruktúru mobilnej aplikácie spolu s označením komunikácie medzi jednotlivými časťami.



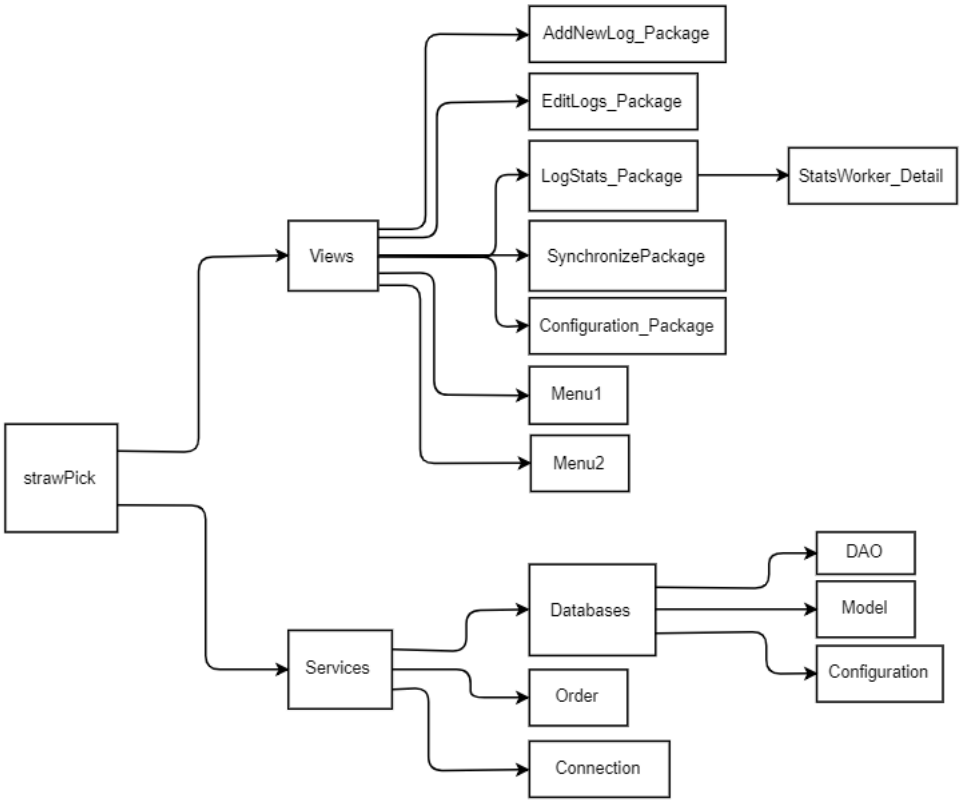
Obrázok 3.1: Rozdelenie aplikácie na funkčné celky a ich vzájomné závislosti

- **dbDescription** slúži k popisu databázy a obsahuje triedy, ktoré reprezentujú použitú databázu; triedy, ktoré tvoria implementáciu *room database* a triedu, ktorá definuje štruktúru databázy použitím konštantných statických premenných, ktoré reprezentujú názvy tabuliek a názvy ich stĺpcov
- **dataProvider** obsahuje triedy, ktoré slúžia k získavaniu potrebných dát pre jednotlivé stránky mobilnej aplikácie
- **sharedPreferencesObjects** združuje triedy, ktoré reprezentujú objekty pracujúce s triedou *SharedPreferences* a zároveň poskytuje konfiguračné parametre pre nadviazanie spojenia so synchronizačnou jednotkou

- **connectionHelper** obsahuje triedy, ktoré zabezpečujú nadviazanie spojenia so synchronizačnou jednotkou a následný proces synchronizácie
- **app** obsahuje triedy, ktoré definujú vnútornú logiku aplikácie a taktiež triedy, ktoré definujú užívateľské prostredie

3.1.1 Rozdelenie na balíčky

Rozdelenie kódu na implementačnej úrovni je zabezpečené vytvorením balíčkov, pričom každý balíček slúži k udržiavaniu tried, ktoré sú funkčne prepojené. Hlavným balíčkom celej aplikácie je balíček *com.mdolnik.strawPick*, ktorý sa vo svojej vnútornej hierarchii následne delí na ďalšie časti (viz Obrázok 3.2).



Obrázok 3.2: Štruktúra balíčkov aplikácie

- **Views** balíček obsahuje jednak triedy implementujúce užívateľské rozhranie aplikácie jednak triedy, ktoré s nimi úzko súvisia. Jednotlivé vnorené balíčky špecifikujú konkrétne časti aplikácie a udržujú triedy podieľajúce sa na implementácii daného pohľadu stránky
- **Services** balíček zastrešuje triedy, ktoré slúžia na vykonávanie funkcií aplikácie a podľa nich sa delí na jednotlivé sekcie
- **Databases** obsahuje všetky triedy, ktoré sa podieľajú na definovaní spojenia a komunikácii s databázou

- **Order** balíček obsahuje triedy starajúce sa o proces radenia zobrazených dát
- **Connection** balíček definuje triedy starajúce sa o riadenie prenosu dát so synchronizačnou jednotkou

3.2 Návrh riešenia mobilnej aplikácie

Mobilná aplikácia je tvorená pomocou XML dokumentov, ktoré definujú užívateľské prostredie aplikácie a tried zabezpečujúcich vykonávanie funkcií poskytovaných mobilnou aplikáciou. Pri návrhu mobilnej aplikácie sme primárne kládli dôraz na jednoduchosť a efektívnosť. Hlavnou úlohou mobilnej aplikácie je vytváranie odovzdávkových dát pracovníkov a ich následná synchronizácia s hlavnou databázou, ktorej návrh bude popísaný v ďalšej kapitole. Vzhľadom na efektivitu funkcie vytvárania odovzdávkových dát, je potrebné, aby bol tento proces pre užívateľa čo najjednoduchší a aby definovanie a vytváranie nového záznamu nezaberalo dlhý časový úsek.

Vzhľadom na jednoduchosť danej funkcie je vizuálny návrh stránky riešený pomocou výberu povinných údajov odovzdávkových dát z pevne stanovených hodnôt. Zároveň je užívateľovi poskytnutá možnosť filtrovania týchto hodnôt pomocou filtračného dialógu, ktorý v prípade veľkého množstva poskytnutých hodnôt zjednodušuje ich výber. Vzhľadom na efektívnosť vytvárania odovzdávkových dát je k aktivite poskytujúcej danú funkciu vytvorený fragment, ktorý udržiava jej dáta. Fragment zároveň udržiava aktuálny stav stránky, a teda zabezpečuje vizuálnu a funkčnú konzistenciu komponent pri zmene stavu zariadenia. Inými slovami, nám fragment slúži na to, aby sme pri zmene orientácie obrazovky, prípadne inej zmene stavu mobilného zariadenia, nestratili definované hodnoty a v tomto zmysle ich nebolo nutné znovu vytvárať.

Medzi ďalšie funkcie mobilnej aplikácie taktiež patrí editácia vytvorených odovzdávkových dát, zobrazenie štatistických údajov z odovzdávkových dát a zmena konfiguračných parametrov pre proces synchronizácie. Pri návrhu riešenia daných funkcií boli zachované predošlé požiadavky na jednoduchosť a efektívnosť spolu s ich implementačnými vlastnosťami popisovanými vyššie.

3.2.1 Prihlásenie do aplikácie

Aby sme zamedzili neoprávnenému prístupu k funkciám aplikácie, je potrebné, aby sa užívateľ, ktorý chce s aplikáciou pracovať, preukázal korektnými

prihlasovacími údajmi. Na základe poskytnutých prihlasovacích údajov aplikácia vykoná proces autentifikácie užívateľa, ktorý je zabezpečený metódou *supervisorExists*. V prípade korektného overenia je užívateľovi umožnený prístup k jednotlivým funkciám aplikácie. V opačnom prípade je užívateľovi prístup zamietnutý. Po úspešnej autentifikácií užívateľa, je pomocou objektu *SharedPreferences* uložená informácia o aktuálne prihlásenom užívateľovi, ktorá je uchovávaná pokým nie je užívateľ z aplikácie odhlásený. To umožňuje kontrolovať a riadiť tok požiadaviek na zobrazenie jednotlivých stránok aplikácie.

3.2.2 Predávanie parametrov medzi aktivitami

Pri prechodoch medzi stránkami mobilnej aplikácie je v niektorých prípadoch nutné zdieľať určité hodnoty, ktoré prispievajú ku správnemu zobrazeniu stránky aplikácie, prípadne sú potrebné k zabezpečeniu funkčnosti. Pre prenos hodnôt medzi jednotlivými aktivitami využíva aplikácia metódu *putExtra*, ktorá umožňuje pridávanie dodatočných dát do objektu typu *Intent.* Zároveň využíva metódy *getIntExtra* a *getStringExtra*, ktoré umožňujú získavanie dodatočných dát z objektu získaného metódou *getIntent* aktuálnej aktivity.

Stránky, ktoré potrebujú dodatočné určenie dát sú nasledovné:

- *Menu1* definuje dodatočný parameter *REQUEST*, ktorý slúži k identifikovaniu stránky, ktorá požaduje zobrazenie daného menu,
- *Menu2* definuje dodatočný parameter *REQUEST*, ktorý slúži k identifikovaniu stránky, ktorá požaduje zobrazenie daného menu, a parameter *DATE*, ktorý slúži na ohraničenie zobrazených dát daného menu,
- *EditPage* a *LogStatsPage* definujú dodatočné parametre *DATE*, *FIELDID* a *FIELDNAME*, ktoré slúžia na obmedzenie zobrazených dát,
- *LogStatsDetailPage* definuje dodatočné parametre *DATE*, *WORKERID* a *FIELDID*, ktoré slúžia na obmedzenie zobrazených dát.

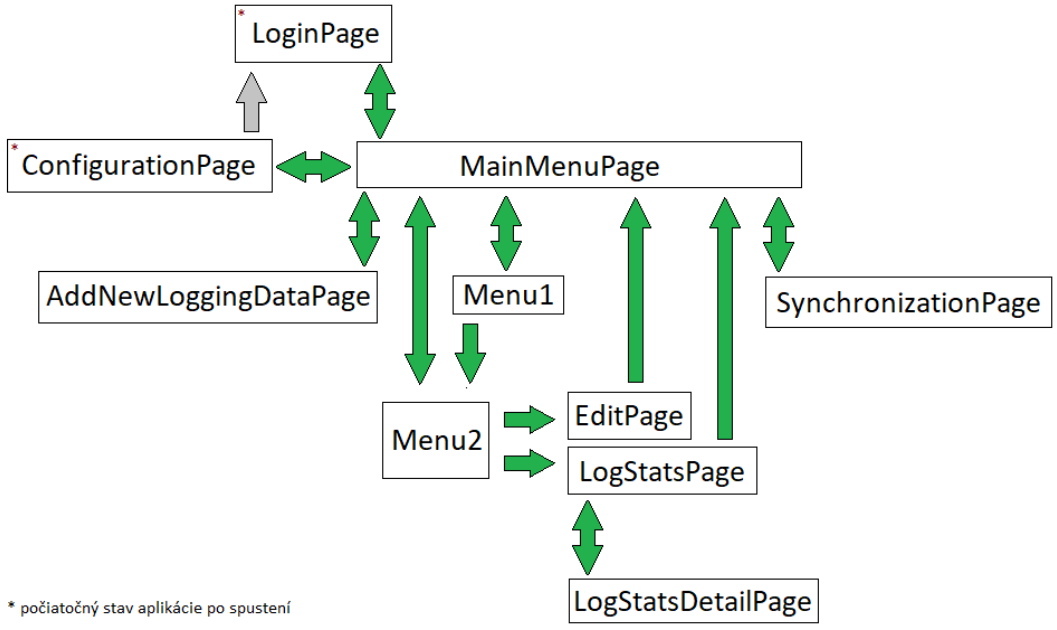
V prípade, že by sme pri presmerovaní na dané stránky nedefinovali hodnoty dodatočných parametrov, aplikácia by nepracovala korektne, poprípade by prestala pracovať úplne.

3.3 Užívateľské prostredie mobilnej aplikácie

Užívateľské prostredie mobilnej aplikácie má responzívny charakter, ktorý je zabezpečený použitím stavebného prvku *guideline*. *Guideline* umožňuje definovať percentuálne rozloženie stránky na jednotlivé sekcie, ktoré následne slúžia na určovanie relatívnej polohy jednotlivých prvkov [13]. Použitím prvku *guideline* aplikácia využíva vždy celý potenciál obrazovky zariadenia. To je v množstve prípadov výhodou, avšak môžu nastať situácie, kedy pri relatívne malých rozmeroch obrazovky mobilného zariadenia nebude zobrazený celý obsah, z dôvodu, že percentuálny výpočet veľkosti komponenty môže byť menší ako požadovaná veľkosť potrebná pre zobrazenie obsahu komponenty.

3.3.1 Návrh užívateľského prostredia

Celá aplikácia sa skladá z niekoľkých stránok, ktoré sa starajú o poskytovanie funkcií užívateľovi. Každá stránka vykonáva práve jednu z hlavných úloh aplikácie. Na nasledujúcom obrázku je zobrazené grafické rozloženie jednotlivých stránok aplikácie spolu s informáciou o prístupe.



Obrázok 3.3: Grafické znázornenie jednotlivých stránok mobilnej aplikácie, spolu s vyznačeným smerom prechodu medzi stránkami

Spoločnými vlastnosťami väčšiny stránok je možnosť radenia dát na základe vybraných atribútov. Táto vlastnosť umožní akumulovanie zobrazených dát, ktoré vo vybraných atribútoch obsahujú rovnakú hodnotu, čím sa zlepší prehľadnosť

zobrazených dát. Výberom jednotlivých radiacich atribútov, a následným kliknutím na atribút filtru je radenie dát uplatnené na zobrazené dáta.

3.3.2 Definovanie užívateľského prostredia počas behu aplikácie

Zobrazované stránky v množstve prípadov obsahujú komponentu, ktorej obsah má byť špecifikovaný až za behu programu. Vzhľadom na fakt, že obsah komponenty môže nadobúdať veľký objem dát, nie je definovanie komponenty za behu programu triviálnou záležitosťou v zmysle časovej zložitosti. Z tohto dôvodu je definovanie obsahu komponenty vykonávané v separátnom vlákne, čo zabezpečuje, že hlavné vlákno aplikácie nie je blokované. Po úspešnom vytvorení obsahu je pomocou metódy *runOnUiThread* obsah pripojený ku komponente v hlavnom vlákne aplikácie.

3.4 Podpora radenia dát

Mobilná aplikácia umožňuje pri využití funkcií editácie a zobrazenia štatistických údajov odovzdávkových dát definovať radiace podmienky, pomocou ktorých sú zobrazené dáta radené vzostupne alebo zostupne. Radiace podmienky sú reprezentované triedou *Order*, ktorá uchováva informáciu o názve atribútu spolu s usporiadaním. Každá inštancia triedy *Order* reprezentuje práve jednu radiacu podmienku. Vzhľadom na to, že aplikácia umožňuje dáta radiť na základe viacerých podmienok, je potrebné, aby bolo možné radiace podmienky vzájomne kombinovať. Túto funkciu poskytuje metóda *addOrderIntoMultipleCondition*, ktorá zabezpečuje pridanie, modifikáciu, prípadne odstránenie atribútu z kombinovanej radiacej podmienky. Kombinovaná radiaca podmienka je v aplikácií reprezentovaná ako zoznam jednotlivých radiacich podmienok, teda ako zoznam objektov triedy *Order*. V prípade, že príde od užívateľa požiadavka na uplatnenie radiacich podmienok je pomocou metódy *orderByConverter* výsledná kombinovaná podmienka na radenie dát transformovaná do textovej reprezentácie, ktorá má formát SQL atribútu *Order By*.

3.5 Udržiavanie dát aktivity

Každá trieda reprezentujúca aktivitu mobilnej aplikácie sa pri zmene stavu mobilného zariadenia znovu vytvára. Na základe tejto vlastnosti je aplikáciou vyžadované spravovať objekt, ktorý bude uchovávať všetky potrebné údaje a aktuálny stav aktivity, aby sme pri zmene stavu mobilného zariadenia nestratili užívateľom zadané hodnoty. Pre vytvorenie takéhoto objektu je umožnené použiť triedu Fragment,

ktorá pri zavolaní funkcie *setRetainInstance* s parametrom *true* umožní udržanie inštancie fragmentu počas procesu opätovného vytvorenia aktivity z dôvodu zmeny stavu zariadenia. Využitím danej funkcie je teda umožnené zachovať existujúcu inštanciu fragmentu, a teda udržiavať dáta aktivity v konzistentnom stave aj po zmene stavu mobilného zariadenia.

3.5.1 Komunikácia s fragmentom

Inštanciu fragmentu k aktivite pripájame v metóde *onCreate*. V prípade že inštancia fragmentu neexistuje, je vytvorená nová. V opačnom prípade je existujúca inštancia pripojená k aktivite a následne v metóde *onRestoreInstanceState* vykonaná synchronizácia aktivity s fragmentom.

V závislosti na ukladaní a získavaní dát fragmentu je nutné definovať vzájomnú interoperabilitu medzi aktivitou a pridruženým fragmentom. Každý fragment definuje interface, ktorý popisuje sadu metód, pomocou ktorých je umožnené informovať aktivitu o zmene stavu dát. Aktivita, ktorá využíva na ukladanie dát fragment, implementuje interface definovaný v danom fragmente, a týmto spôsobom si zabezpečuje získanie adekvátnej informácie v prípade, že nastane aktualizácia dát vo fragmente. Implementovanie definovaného rozhrania v aktivite umožňuje priradiť kontext získaný z aktivity do objektu, ktorý zodpovedá definovanému rozhraniu. Tento proces prebieha prostredníctvom metódy fragmentu *onAttach.* V prípade, že aktivita neimplementuje definované rozhranie, pripojenie fragmentu k aktivite prebehne úspešne, avšak pri zmene hodnôt dát nie je aktivita o tomto stave informovaná, a tak konzistencia dát aktivity s fragmentom nie je zabezpečená.

3.6 Práca s dátami

V tejto podkapitole popíšeme časti aplikácie, ktoré slúžia na komunikáciu s databázou, definujeme schému databázy spolu s popisom tabuliek, ktoré sú v databáze definované.

3.6.1 Vytvorenie spojenia s databázou

Spojenie s databázou je zabezpečené použitím *Room Persistence Library*, ktorá umožňuje priame mapovanie objektov na riadky SQLite relačného databázového systému. Na vytvorenie spojenia s databázou je potrebné definovať hlavné komponenty *Room Databases* štruktúry, ktoré popisujú databázu.

Medzi hlavné komponenty podieľajúce sa na definovaní spojenia s databázou patrí trieda označená anotáciou *@Dao*, ktorá obsahuje metódy pre prístup k databáze, trieda označená *@Database*, ktorá slúži ako hlavný prístupový bod komunikácie, a entita, ktorá reprezentuje jednotlivé tabuľky databázy. Pre úspešné definovanie triedy označenej anotáciou *@Database* je nevyhnutné aby implementácia splňovala nasledujúce podmienky [14]:

- ide o abstraktnú triedu, ktorá rozširuje *RoomDatabase*,
- obsahuje abstraktnú metódu bez argumentov, ktorá vracia triedu označenú anotáciou *@Dao*,
- v anotácií triedy sa nachádza zoznam entít, ktoré sú s databázou asociované.

Špecifickou vlastnosťou implementácie spomínanej triedy je udržovanie súkromného statického objektu, ktorý reprezentuje objekt danej triedy a pomocou ktorého môžeme pristupovať k nami popisovanej databáze.

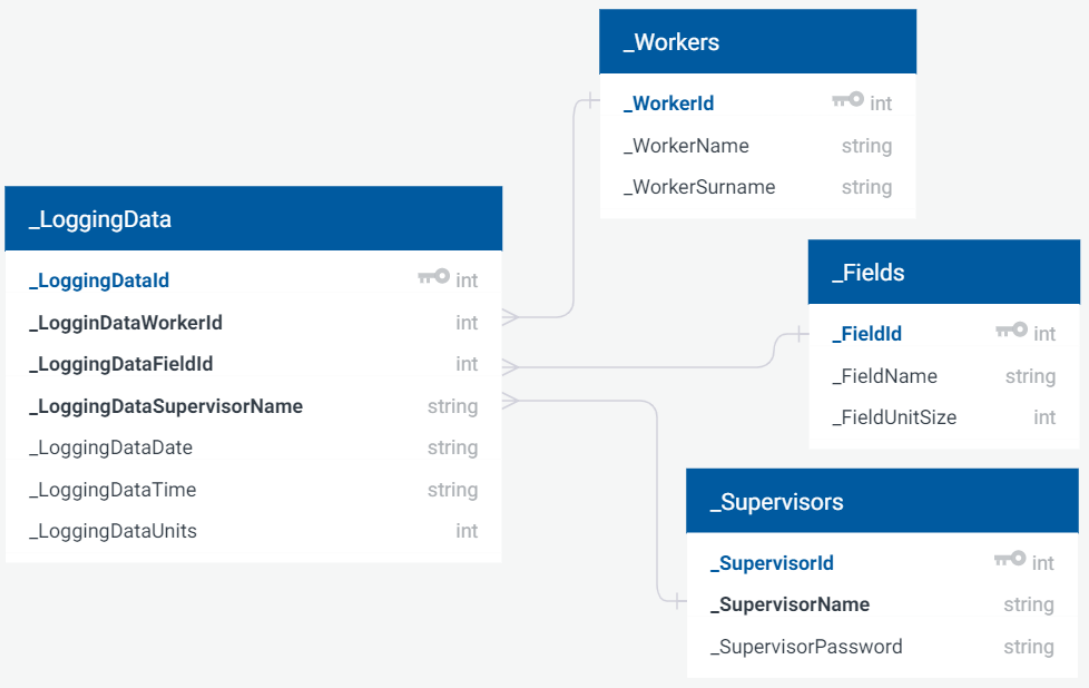
3.6.2 Komunikácia s databázou

Komunikácia s databázou je zabezpečená prostredníctvom metód, ktoré sú súčasťou objektov označených anotáciou *@Dao*. Tieto metódy môžu reprezentovať SQL dotazy rôznych typov, ako napríklad pridanie, zmazanie, výber záznamov a ďalšie. Taktiež umožňujú definovanie tzv. *rawQuery* metód, ktoré sú použité na vykonávanie SQL dotazov vytvorených za behu aplikácie.

Samotný proces získavania dát z databázy je vykonávaný využitím triedy typu *AsyncTask*, ktorá zabezpečuje potrebné získavanie dát pre aplikáciu v separátnom vlákne, a tak nedochádza k blokovaniu hlavného vlákna aplikácie.

3.6.3 Formát tabuliek

Dáta, s ktorými aplikácia pracuje sa delia do štyroch oblastí, podľa informácií, ktoré poskytujú. Nasledujúci obrázok znázorňuje grafické rozloženie jednotlivých tabuliek, s ktorými mobilná aplikácia pracuje, a ktoré zoskupujú záznamy, nesúce rovnakú výpovednú informáciu.



Obrázok 3.4: Schéma použitej lokálnej databázy mobilnej aplikácie

- **_Supervisors** tabuľka obsahuje záznamy reprezentujúce jednotlivých užívateľov, ktorý majú umožnený prístup do mobilnej aplikácie
- **_Workers** tabuľka obsahuje záznamy reprezentujúce jednotlivých pracovníkov, ktorí sú vedení ako zberači
- **_Fields** tabuľka obsahuje záznamy reprezentujúce jednotlivé polia, na ktorých môže byť vykonávaný zber
- **_LoggingData** tabuľka obsahuje záznamy, ktoré zodpovedajú odovzdávkovým dátam jednotlivých pracovníkov

Predpokladané formáty atribútov jednotlivých tabuliek:

- ***_LoggingDataDate*** má formát *yyyy/mm/dd*
- ***_LoggingDataTime*** má formát *hh:mm:ss*

Ako je z obrázku 3.4 zrejmé, niektoré tabuľky databázy majú medzi sebou definované prepojenie. Toto prepojenie je určené iba na teoretickej úrovni implementácie. Dané prepojenie nie je špecifikované nikde v objektoch reprezentujúcich štruktúru databázy, s tým zámerom, aby dáta tabuľky boli vždy konzistentné. V závislosti na synchronizačnom procese aplikácie platí, že pri prenose dát medzi mobilnou aplikáciou a hlavnou databázou sú najskôr prenesené záznamy tabuliek *Supervisor*,

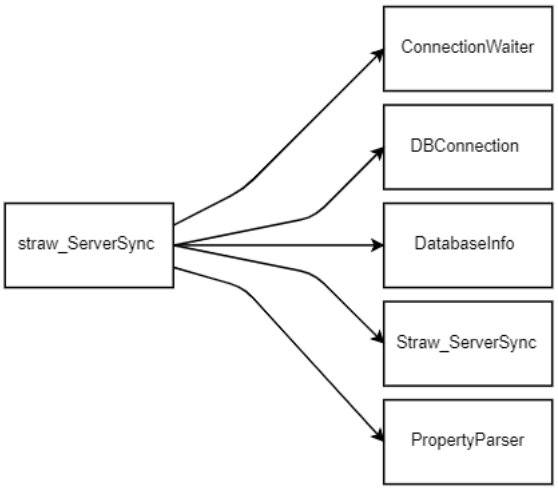
Field a *Worker*. V zmysle tohto postupu by sme sa mohli dostať do situácie, kedy by záznamy definované v tabuľke *LoggingData* vyžadovali svoje vymazanie, prípadne zmenu z dôvodu neexistencie relačne spojeného záznamu, na ktorý sa odovzdávkový záznam odkazoval. Stav, kedy by relačne spojené záznamy prestali existovať, nemá v aplikácií spôsobiť odstránenie záznamu, ktorý sa naň odkazuje. Preto pri definovaní štruktúry našej databázy nevyužívame funkciu definovania *ForeignKeys* a prepojenie jednotlivých tabuliek definujeme iba na teoretickej úrovni.

4 Popis riešenia synchronizačnej jednotky strawSync

V tejto kapitole sa budeme zaoberať jednotlivými časťami synchronizačnej jednotky, návrhom jej riešenia a rozložením tried zabezpečujúcich funkčnosť aplikácie. Popíšeme priebeh synchronizácie spolu s riadiacimi správami, ktoré reprezentujú riadenie toku dát prenášaných medzi aplikáciami.

4.1 Štruktúra synchronizačnej jednotky

Celá funkčnosť aplikácie je tvorená v balíčku *straw_ServerSync*, ktorý obsahuje všetky triedy podieľajúce sa na zabezpečení poskytovaných funkcií aplikácie.



Obrázok 4.1: Schéma štruktúry aplikácie

- **ConnectionWaiter** je trieda, ktorá zabezpečuje vykonávanie procesu synchronizácie
- **DBConnection** je trieda, ktorá sa stará o vytvorenie spojenia s databázou a následnou komunikáciou s ňou
- **DatabaseInfo** je trieda, ktorá reprezentuje schému použitej databázy pomocou statických parametrov slúžiacich na udržiavanie názvov tabuliek a ich stĺpcov
- **Straw_ServerSync** je trieda obsahujúca statickú metódu *main*, ktorá je vykonaná po spustení aplikácie a ktorá riadi jej chod
- **PropertyParser** je trieda, ktorá slúži na načítanie a reprezentovanie konfiguračných parametrov naprieč aplikáciou

4.2 Návrh riešenia synchronizačnej jednotky

Synchronizačná jednotka je vytvorená ako konzolová aplikácia, ktorá inicializuje hlavnú databázu, a umožňuje synchronizovať dáta mobilnej aplikácie s dátami hlavnej databázy. Pri spustení synchronizačnej jednotky je potrebné zadať povinný parameter *-conf* spolu s cestou k súboru, v ktorom sú definované konfiguračné parametre, ktoré musia byť pri spustení aplikácie inicializované. Po úspešnej inicializácii konfiguračných parametrov sa aplikácia pokúša vytvoriť spojenie s hlavnou databázou. Toto spojenie je definované hodnotami získanými z konfiguračných parametrov. Pri vytvorení spojenia prebieha kontrola schémy databázy. Pokiaľ je aktuálna schéma databázy odlišná od popisu schémy v nasledujúcej podkapitole, je obsah celej databázy odstránený a prebieha pokus o vytvorenie tabuliek, ktoré zodpovedajú nami požadovanej schéme. Ak je aktuálna schéma zhodná s požadovanou, aplikácia čaká v cykle na spojenie s mobilnou aplikáciou. Po prijatí požiadavky na nadviazanie spojenia je v separátnom vlákne spustený synchronizačný proces, ktorý je riadený pomocou riadiacich správ. V prípade že počas synchronizácie nastane chyba aplikácie, z akéhokoľvek dôvodu (ako napríklad strata spojenia s databázou), je dané spojenie zrušené a synchronizácia končí chybovým stavom.

4.3 Načítavanie a formát konfiguračných dát

Načítavanie konfiguračných dát aplikácie je zabezpečené prostredníctvom triedy *Properties*. Zavolaním metódy *load* na inštancii triedy *Properties* s parametrom určujúcim súbor, ktorý obsahuje konfiguračné parametre, je vytvorený zoznam vlastností reprezentovaný danou inštanciou. Súbor, z ktorého sa konfiguračné parametre načítavajú musí obsahovať všetky povinné atribúty spolu s ich hodnotami. V opačnom prípade nebude aplikácia správne spustená a užívateľ bude o tomto stave informovaný výpisom správy.

Podporované atribúty konfiguračného súboru sú využívané k definovaniu spojenia s databázou a k zabezpečeniu spojenia s mobilnou aplikáciou. Atribúty, ktoré sú v zmysle získania úspešného spojenia s databázou potrebné sú *hostname*, *port* a *properties*, pomocou ktorých sa odkazujeme na požadovanú databázu, a *username* spolu s atribútom *password*, pomocou ktorých je získaný prístup k databáze. Atribúty podieľajúce sa na definovaní spojenia s mobilnou aplikáciu sú *port* reprezentujúci číslo portu pre komunikáciu s mobilnou aplikáciou a *logger* definujúci cestu k súboru,

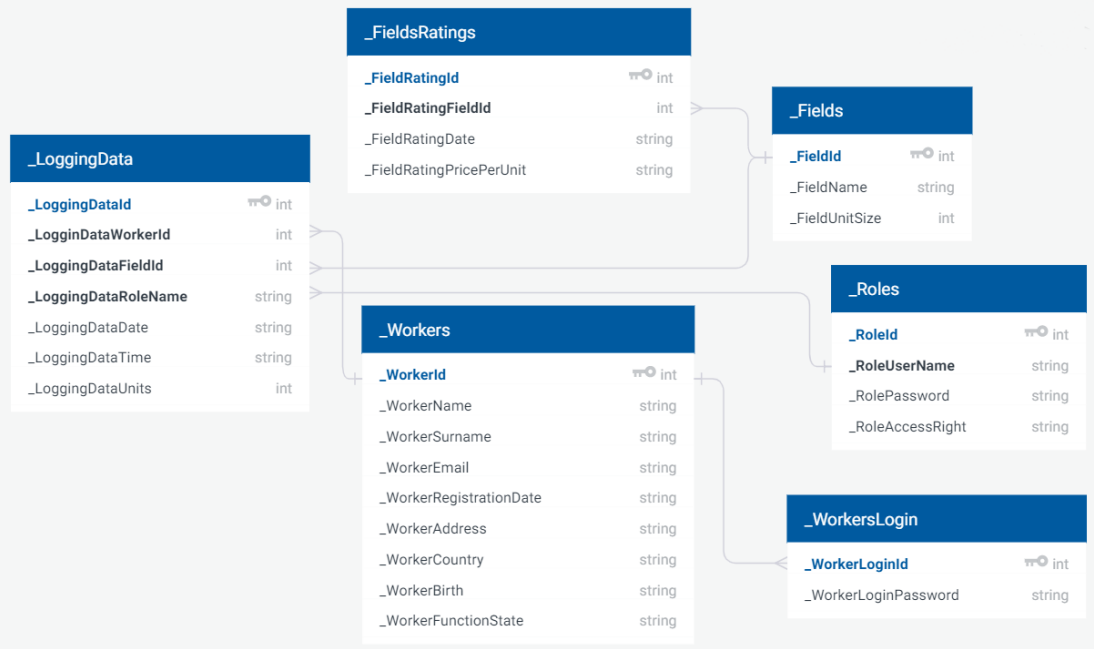
ktorý slúži na ukladanie riadiacich správ z procesu synchronizácie. Na nasledujúcom obrázku môžeme vidieť príklad konfiguračného súboru, ktorý definuje konfiguračné parametre spolu s priradenými hodnotami.

```
# database configuration attributes
database.hostname=localhost
database.port=3306
database.username=iStraw_User
database.password=Chl4dn1ck4
# sync configuration attributes
synchronization.port=7529
synchronization.logger=outputNew.txt
```

Obrázok 4.2: Výzor konfiguračného súboru (vzorový príklad)

4.4 Schéma databázy

Databáza, s ktorou pracuje synchronizačná jednotka je totožná s databázou webovej aplikácie. Schéma použitej databázy je nadstavbou lokálnej databázy mobilnej aplikácie. Na nasledujúcom obrázku môžeme vidieť formát jednotlivých tabuliek, ktoré sú v hlavnej databáze udržiavané.



Obrázok 4.3: Schéma tabuliek databázy spolu s typom dát, ktoré združujú

- **_LoggingData** tabuľka obsahuje záznamy, ktoré zodpovedajú odovzdávkovým dátam jednotlivých pracovníkov
- **_Workers** tabuľka obsahuje záznamy, ktoré reprezentujú jednotlivých pracovníkov

- **_WorkersLogin** tabuľka obsahuje záznamy, ktoré slúžia ako prihlasovacie údaje jednotlivým pracovníkom, ktorí sú v aplikácií vedení ako zberači
- **_Roles** tabuľka obsahuje záznamy, ktoré slúžia ako prihlasovacie údaje spolu s prístupovými právami pracovníkov, ktorí majú právo prístupu do mobilnej aplikácie strawPick a webovej aplikácie strawManager
- **_Fields** tabuľka obsahuje záznamy, ktoré reprezentujú polia, na ktorých je umožnený zber
- **_FieldsRatings** tabuľka ukladá dáta, ktoré reprezentujú cenu za jednotku na jednotlivých poliach za určitý deň zberu

Predpokladané formáty atribútov jednotlivých tabuliek:

- ***_LoggingDataDate, _FieldsRatingDate*** má formát *yyyy/mm/dd*
- ***_LoggingDataTime*** má formát *hh:mm:ss*
- ***_LoggingDataRegistrationDate*** má formát *yyyy-mm-dd*
- ***_LoggingDataFunctionState*** môže nadobúdať hodnotu *picker* a *role*
- ***_RoleAccessRight*** môže nadobúdať hodnotu *admin* a *supervisor*

4.5 Komunikácia s databázou

Komunikácia s databázou je riadená triedou *DBConnection*, ktorej inštancia nám sprístupňuje použitie metód na získanie dát z databázy. Vytvorením inštancie danej triedy vznikne objekt, ktorý má nastavené atribúty, potrebné pre nadviazanie spojenia s databázou. Na vytvorenej inštancii je následne nutné zavolať metódu *tryToConnect*, ktorá vytvára spojenie s databázou. V prípade, že spojenie je úspešne vytvorené, je užívateľovi umožnené pracovať s databázou pomocou daného objektu. Ak by po vytvorení inštancie triedy *DBConnection* nebola zavolaná metóda *tryToConnect*, spojenie s databázou by nebolo nadviazané a výsledkom každej metódy by bol chybový stav dotazu.

Na získavanie, modifikovanie aj odstraňovanie záznamov z tabuliek databázy sú použité objekty vytvoreného spojenia, *PreparedStatement*. Tieto objekty umožňujú viacnásobné vykonávanie použitého SQL dotazu uplatnením rôznych argumentov dotazu, a teda umožňujú efektívnejšie vykonávanie dotazov nad SQL tabuľkami.

4.6 Proces synchronizácie

Hlavnou úlohou synchronizačnej jednotky je prenos dát medzi hlavnou databázou a lokálnou databázou mobilnej aplikácie strawPick. Celý tento proces prenosu dát je riadený triedou *Synchronization* v separátnom vlákne, kde je pre každú požiadavku na synchronizáciu vytvorená nová inštancia danej triedy. Komunikácia medzi aplikáciami prebieha pomocou objektov získaných z metód *getInputStream* a *getOutputStream* z inštancie triedy *Socket*, ktorá reprezentuje vytvorené spojenie.

4.6.1 Riadenie synchronizácie

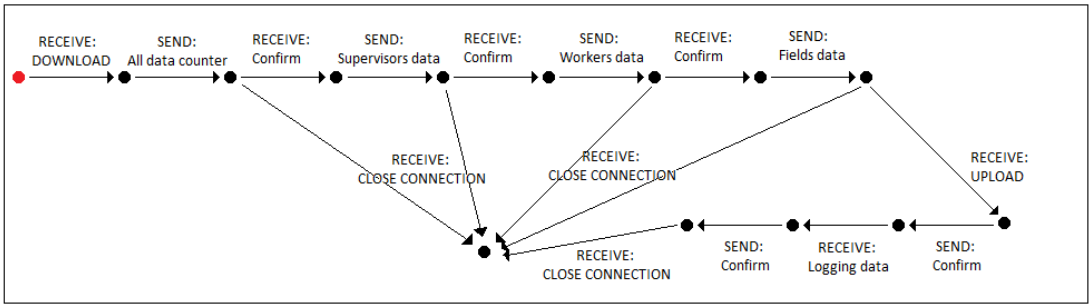
Synchronizačná jednotka podporuje dva druhy prenosu dát a to typ *download* a *upload*. Typ *download* slúži k prenosu dát smerom z hlavnej databázy do mobilnej aplikácie strawPick. Tento prenos dát zabezpečuje konzistenciu záznamov o poliach, zberačoch a iných pracovníkoch, ktorí majú prístupové práva do mobilnej aplikácie. Pri prenose dát typu *download* synchronizačná jednotka prechádza nasledujúcimi stavmi:

1. Synchronizačná jednotka zašle mobilnej aplikácii informáciu o počte všetkých dát, ktoré budú zaslané do mobilnej aplikácie.
2. Synchronizačná jednotka príjme kontrolný kód.
3. Synchronizačná jednotka zašle mobilnej aplikácii informáciu o počte pracovníkov s prístupovými právami do aplikácie a následne odošle jednotlivé záznamy.
4. Synchronizačná jednotka príjme kontrolný kód.
5. Synchronizačná jednotka zašle mobilnej aplikácii informáciu o počte pracovníkov, ktorí sú vedení ako zberači a následne odošle jednotlivé záznamy.
6. Synchronizačná jednotka príjme kontrolný kód.
7. Synchronizačná jednotka zašle mobilnej aplikácii informáciu o počte polí, na ktorých je možné vykonávať zber a následne odošle jednotlivé záznamy.

Prenos typu *upload* slúži k prenosu dát smerom z mobilnej aplikácie strawPick do hlavnej databázy, a teda ide o prenos opačný k typu *download*. Tento prenos slúži k modifikácii tabuľky hlavnej databázy, ktorá udržiava odovzdávkové dáta, o záznamy získané z mobilnej aplikácie strawPick. Pri vykonávaní prenosu dát typu *upload* synchronizačná jednotka prechádza nasledujúcimi stavmi:

1. Synchronizačná jednotka príjme informáciu o počte odovzdávkových dát, ktoré budú mobilnou aplikáciou prenesené a následne príjme požadované dáta.
2. Synchronizačná jednotka zašle mobilnej aplikácii kontrolný kód.

Po dokončení prenosu *download* alebo *upload*, mobilná aplikácia zašle synchronizačnej jednotke tag *closeConnection*, ktorým naznačí ukončenie spojenia. Tento tag môže byť použitý aj ako kontrolný kód v prípade degradácie získaných dát a oboznamuje tým synchronizačnú jednotku, že chce proces prenosu dát ukončiť. Druhým typom kontrolného kódu je tag *success*, ktorým si aplikácie medzi sebou potvrdzujú prijatie zaslaných dát.



Obrázok 4.4: Schéma riadenia synchronizácie z pohľadu synchronizačnej jednotky

4.7 Informačné výpisy aplikácie

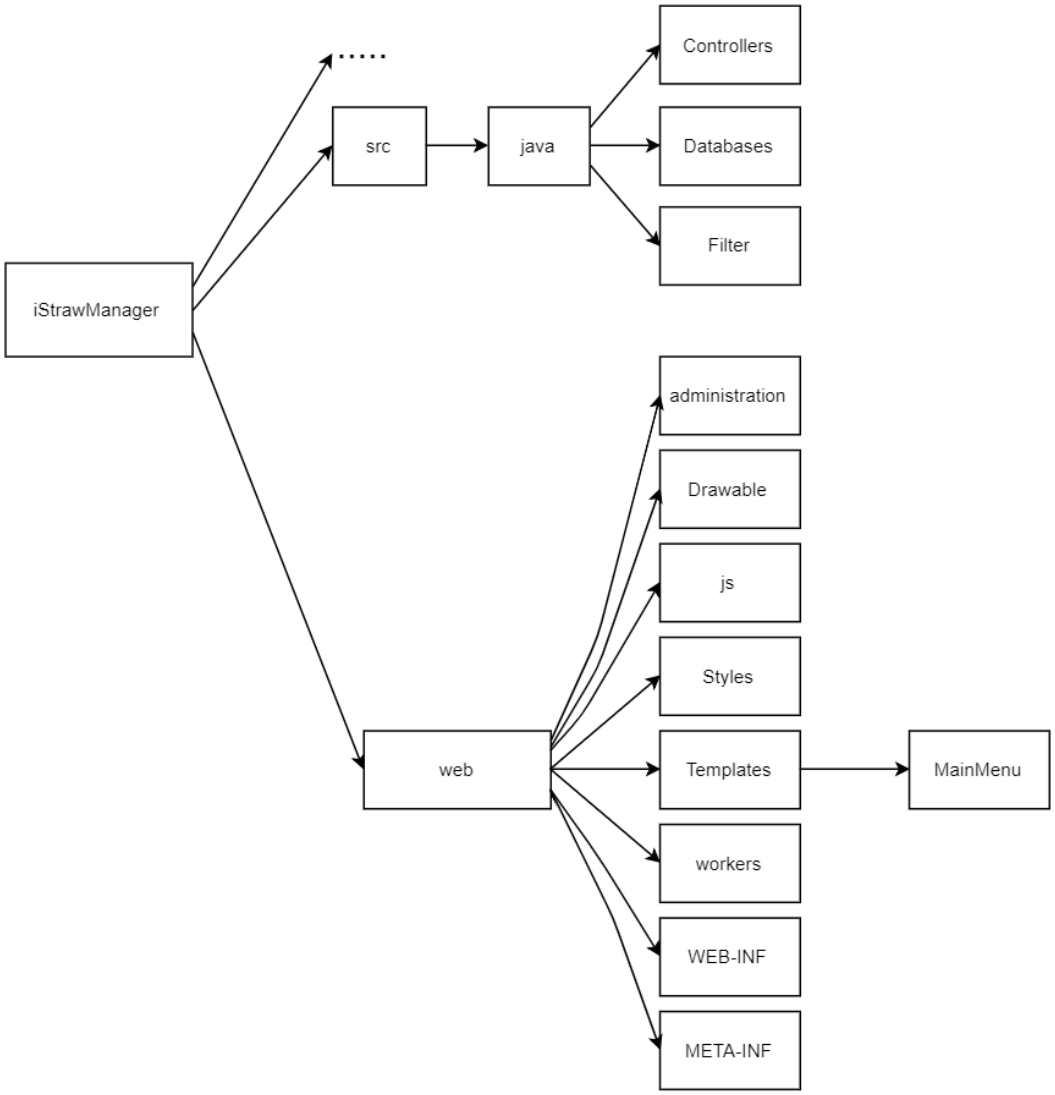
V prípade, že v konfiguračnom súbore bol definovaný parameter *logger*, je využitá funkcia výpisu informačných správ aplikácie z procesu synchronizácie. Výpisy, ktoré sú pomocou súboru uchovávané, reprezentujú zmeny stavu aplikácie v procese synchronizácie, akou je nadviazanie spojenia s mobilnou aplikáciou, prijatie a odoslanie dát mobilnej aplikácie a ďalšie, ktoré umožňujú v prípade výskytu chyby, dohľadať miesto v aplikácií, kde a prečo chyba nastala. Keďže synchronizačnej jednotke je umožnené komunikovať zároveň s viacerými mobilnými zariadeniami, je nutné implementovať metódu zapisovania do súboru ako *Thread Safe* metódu. Na základe tejto skutočnosti je zapisovanie do súboru riešené pomocou metódy *addIntoQueue*. Ide o synchronizovanú metódu, ktorá ukladá správy do dátovej štruktúry typu FIFO a následne, ak je veľkosť dátovej štruktúry väčšia ako povolené maximum, prípadne ak je metóda zavolaná s parametrom okamžitého zápisu nastaveného na *true*, je vykonaný zápis správ z dátovej štruktúry do súboru. Definovanie danej metódy pod označením *synchronized* nám umožní, aby k nej mohlo pristupovať v rovnakom čase vždy iba jedno vlákno.

5 Popis riešenia webovej aplikácie strawManager

V tejto kapitole popíšeme jednotlivé časti, ktoré sa podieľajú na tvorbe webovej aplikácie. Popíšeme taktiež návrh riešenia vizuálnej a funkčnej stránky aplikácie. Definujeme použité prostriedky a rozoberieme priebeh komunikácie s MySQL relačným databázovým systémom.

5.1 Štruktúra webovej aplikácie

Webová aplikácia pozostáva z niekoľkých častí, ktoré sa podieľajú na jej definovaní. Najpodstatnejšími časťami aplikácie sú balíčky *src* a *web*, ktoré udržiavajú všetky implementačné časti funkcií aplikácie.



Obrázok 5.1: Štruktúra a členenie kódu webovej aplikácie

- **Web** balíček obsahuje JSP stránky, ktoré definujú výzor prihlasovacej a úvodnej stránky a ďalšie vnorené balíčky

- **Controllers** balíček obsahuje všetky triedy, ktoré plnia funkciu Servletov, a teda zabezpečujú vykonávanie funkcií požadovaných JSP stránkami
- **Databases** balíček obsahuje všetky triedy, ktoré sa podieľajú na definovaní a vytvorení spojenia a následnej komunikácii s databázou
- **Filter** balíček obsahuje triedy slúžiace ako filtračná jednotka, ktorá zabezpečuje správne pristupovanie k stránkam na základe prístupových práv užívateľa
- **Drawable** balíček obsahuje všetky ikony, obrázky a ďalšie grafické symboly, ktoré sú aplikáciou používané
- **Js** balíček obsahuje všetky súbory typu javascript definujúce funkcie, ktoré je umožnené v aplikácií používať
- **Styles** balíček obsahuje všetky kaskádové štýly, ktoré sú aplikáciou využívané
- **WEB-INF** balíček obsahuje tzv. *deployment descriptors* webovej aplikácie
- **Templates** balíček obsahuje časti webových stránok, ktoré slúžia ako šablóny
- **Workers** balíček obsahuje JSP stránky, ktoré definujú výzor stránok sekcie 'Správca pracovníkov'
- **Administration** balíček obsahuje JSP stránky, ktoré definujú výzor stránok sekcie 'Administrácia'

5.2 Typy účtov

Webová aplikácia podporuje tri typy užívateľských rolí s prístupovými právami, pričom každá z nich disponuje rôznou škálou oprávnení. Jednotlivé užívateľské role tak umožňujú pracovať s webovou aplikáciou v rôznych režimoch spravovania dát. Minimálne práva na prácu s dátami sú definované užívateľskou rolou *picker*. Pracovník s priradenou užívateľskou rolou *picker* smie s webovou aplikáciou pracovať výlučne v read-only režime, ktorý umožňuje zobrazenie dát prihláseného pracovníka, pričom nie je oprávnený pridávať ani editovať žiadne záznamy, a teda sekcie aplikácie, ktoré tieto funkcie poskytujú sú pred daným užívateľom skryté. Druhou užívateľskou rolou je *supervisor*, ktorá disponuje širším rozpätím oprávnení, umožňuje vykonávanie funkcií pridávania, editácie či odstraňovania odovzdávkových dát jednotlivých pracovníkov. Treťou užívateľskou rolou, ktorá môže byť v aplikácií využitá je rola *admin*. Ide o užívateľskú rolu s neobmedzenými právami pre prácu s dátami. Umožňuje využívať všetky funkcie ponúkané webovou aplikáciou.

Pracovník, ktorý má priradenú užívateľskú rolu *supervisor* alebo *admin* má okrem prístupu do webovej aplikácie umožnený prístup aj do mobilnej aplikácie.

5.3 Riadenie prístupu k stránkam webovej aplikácie

Vzhľadom na obmedzenia prístupu k funkciám webovej aplikácie definovaných pomocou užívateľských rolí, je nutné, aby mal prihlásený užívateľ prístup výhradne k tým častiam aplikácie, ktoré zodpovedajú jeho právomociam. Pre zaistenie danej funkčnosti je využitá implementácia triedy *Filter*. Pri každom pokuse o získanie prístupu k stránke webovej aplikácie sú filtrom overené právomoci užívateľa, ktorý k stránke pristupuje. V prípade, že užívateľ disponuje právami prístupu k danej stránke, filter vyhovie danej požiadavke a stránku užívateľovi sprístupní. Naopak, stránka nie je užívateľovi sprístupnená, ak nevlastní prístupové práva. Existujú dva dôvody, ktoré môžu zapríčiňovať zamietnutie prístupu k stránke webovej aplikácie. Jedným z dôvodov je nedostatočné definovanie práv užívateľskej role, ďalším je fakt, že užívateľ ešte neposkytol prihlasovacie údaje a teda nie je vo webovej aplikácii dosiaľ prihlásený. V týchto prípadoch je užívateľ presmerovaný na chybovú stránku, prípadne na prihlasovaciu stránku webovej aplikácie.

5.4 Užívateľské rozhranie webovej aplikácie

Užívateľské prostredie webovej aplikácie je tvorené pomocou JSP stránok, ktoré definujú dynamické webové stránky využitím HTML značiek a jazyka Java. To nám umožňuje vytvoriť pri inicializácii stránky konštantné parametre získané z definovaných Java tried. Následne môžu byť definované konštantné parametre použité ako hodnoty jednotlivých elementov stránky. HTML značky slúžia na definovanie obsahu jednotlivých stránok, pričom neumožňujú nastavovať rozloženie komponent a ich vizuálne vlastnosti. Definovanie výzoru webovej aplikácie výhradne pomocou HTML má za následok veľmi jednoduchý vzhľad bez možnosti prispôsobenia na konkrétny typ obrazovky. Na dodatočné definovanie vizuálnych vlastností jednotlivých elementov využívame CSS atribúty, ktoré slúžia na špecifikovanie dizajnu výslednej stránky webovej aplikácie. Stránky webovej aplikácie majú responzívny charakter, a teda umožňujú prispôsobenie výzoru stránky aktuálne používanej veľkosti obrazovky. Rozpoloženie jednotlivých komponent môže byť v závislosti na veľkosti obrazovky pozmenené, v prípade že je použité CSS pravidlo *@media*.

5.5 Práca s grafmi

Webová aplikácia využíva grafickú reprezentáciu zobrazenia štatistických údajov získaných z odovzdávkových dát. Vykreslenie grafu je zabezpečené použitím knižnice *Chart.js*, ktorá poskytuje rôzne typy grafov na použitie. Definovať graf je umožnené výlučne na HTML elemente *canvas*, ktorý slúži ako kontajner pre grafické komponenty stránky, definované za behu aplikácie. Pri vytváraní grafu je umožnené špecifikovať potrebné vlastnosti, ktorými má graf disponovať a zároveň metódy, ktoré majú slúžiť na spracovávanie požiadaviek.

5.6 Práca s dátami

V tejto podkapitole sa budeme bližšie venovať jednotlivým častiam webovej aplikácie, ktoré akýmkoľvek spôsobom pracujú s dátami uchovávanými v databáze. Popisujeme ako aplikácia získava a nadväzuje spojenie s databázou a následne popisujeme proces komunikácie. Schéma použitej databázy nebude popísaná, keďže webová aplikácia pracuje s tou istou databázou ako synchronizačná jednotka strawSync.

5.6.1 Vytvorenie spojenia a komunikácia s databázou

Webová aplikácia definuje balíček *Databases*, ktorý obsahuje triedy vykonávajúce funkciu komunikačných prvkov s databázou. Trieda *DB_BasicInfo* poskytuje opis použitej databázy definovaním verejných statických premenných, ktoré reprezentujú názvy jednotlivých tabuliek, spolu s názvami ich stĺpcov. Abstraktná trieda *DB_iStrawManager* slúži k vytvoreniu spojenia medzi webovou aplikáciou a databázou. Spojenie vytvára využitím definovaného rozhrania JDBC, pomocou ktorého je taktiež vykonávaná komunikácia s databázou. Triedy, ktoré sa podieľajú na samotnej komunikácií s databázou sú rozšírením triedy *DB_iStrawManager* a zastrešujú prácu s dátami rovnakého informačného charakteru. Jednotlivé metódy týchto tried teda slúžia na získavanie, modifikovanie, prípadne odstránenie dát z databázy. Pri obdržaní požiadavky na komunikáciu s databázou je vždy vytvorená nová inštancia objektu reprezentujúceho komunikačné spojenie. Tým je zabezpečená komunikácia s databázou ako *Thread Safe* a teda v prípade viacerých požiadaviek na komunikáciu v rovnaký čas je zaručené bezpečné fungovanie. Keďže vykonávanie dotazu nad databázou nemusí byť v každom prípade triviálnou záležitosťou, aplikácia využíva možnosť zobrazenia načítavacieho dialógu počas komunikácie s databázou,

aby tak zamedzila vzniku nekorektného stavu aplikácie z dôvodu veľkého časového zaťaženia SQL dotazu.

5.6.2 Vykonávanie SQL dotazov

Pre vykonávanie funkcií, ktoré definujú komunikačné triedy nám rozhranie JDBC poskytuje objekty *Statement* a *PreparedStatement*, pomocou ktorých sú realizované SQL dotazy. Podľa oficiálnej dokumentácie je *Statement* definovaný ako objekt používaný na vykonávanie statického SQL dotazu, ktorý vracia výsledok [15]. *PreparedStatement* je definovaný ako objekt, ktorý reprezentuje prekompilovaný SQL dotaz, ktorý môže byť efektívne použitý pri viacnásobnej aplikácii dotazu [16]. Objekt *PreparedStatement* sa javí ako výhodnejšia voľba, avšak treba uvážiť spôsob, akým sú požiadavky na prácu s databázou vykonávané. Keďže pri každej požiadavke na prácu s dátami databázy webová aplikácia vytvára novú inštanciu komunikačnej triedy, je použitie objektu *PreparedStatement* zbytočné, pretože sa nevyužívajú možnosti viacnásobného aplikovania prekompilovaného SQL dotazu. Tým by nám vznikol zbytočný režijný náklad na prekompilovanie SQL dotazu a jeho uchovávanie v pamäti. Z tohto hľadiska je výhodnejšie použiť objekt *Statement*. Na druhej strane značnú nevýhodu objektu *Statement* vidíme v *SQL injection*, teda v dosadzovaní hodnôt do SQL dotazu. V prípade, že definujeme hodnoty SQL dotazu za behu aplikácie, môžu nastať prípady, kedy poskytnutá hodnota povolí SQL dotazu sprístupnenie neoprávnených dát [17].

Vzhľadom na vyššie spomenuté klady a zápory jednotlivých objektov, ktoré umožňujú vykonávanie SQL dotazov, sme použili objekt *PreparedStatement* v metódach, ktoré testujú poskytnuté prihlasovacie údaje, a teda vykonávajú proces autentifikácie užívateľa. Na vykonávanie ostatných SQL dotazov sme vybrali objekt *Statement*, aby sme zabránili zbytočným režijným nákladom ako v prípade objektu *PreparedStatement* pri dotazoch, ktoré sú použité jednorazovo.

5.7 Načítanie konfiguračných parametrov

Na uchovávanie konfiguračných parametrov webovej aplikácie strawManager používame súbor *configurationFile.conf* umiestnený v adresári WEB-INF. Konfiguračný súbor udržiava dáta potrebné na definovanie spojenia s databázou a pre správne nastavenie emailového poskytovateľa. Konfiguračné parametre sú z konfiguračného súboru načítavané pomocou metód triedy *Properties*.

Java Servlet *ConfigurationParameters* obsahuje statické parametre, ktoré zodpovedajú podporovaným konfiguračným parametrom. Pomocou metódy *init* vykonáva mapovanie konfiguračných parametrov zo súboru na premenné definované danou triedou. Trieda *ConfigurationParameters* je inicializovaná pri spustení aplikácie použitím atribútu <*load-on-startup*> daného Java Servletu v súbore *web.xml*. To zabezpečuje, že konfiguračné parametre sú správne načítané pri spustení aplikácie a počas celej jej životnosti sú k dispozícií pomocou statických metód, ktoré sprístupňujú namapované parametre.

5.8 Vykonávanie POST a GET metód

Webová aplikácia využíva Java Servlety na obsluhovanie POST a GET požiadaviek HTML stránky. Jednotlivé požiadavky môžeme zo strany serveru rozdeliť na dva typy, podľa toho, ako na požiadavku reagujú, a to na výsledkové a bezvýsledkové. Ako je z názvov zrejmé, delenie je vykonávané na základe skutočnosti, či server vracia/nevracia výsledok požiadavky JSP stránke, ktorá ju vytvorila. V prípade, že server neposkytuje spätnú väzbu JSP stránke je nutné, aby server vytvoril riadiaci príkaz, ktorý nesie informáciu o tom, do akého stavu sa má webová aplikácia presunúť. Príkladom takéhoto riadiaceho príkazu je presmerovanie. Na vytvorenie daného typu požiadavky JSP stránka využíva možnosť definovania formulára, ktorý po potvrdení vytvorí požiadavku na server. V opačnom prípade, keď server poskytuje výsledok na požiadavku, je JSP stránkou vyžiadané čakať na výsledok, ktorý je po realizácii aplikovaný na výzor JSP stránky. Príkladom využitia výsledkovej požiadavky je, keď JSP stránka žiada o dáta získané z databázy použitím GET metódy a následne výsledok priradí ako obsah jednotlivým HTML elementom. Všetky Java Servlety, ktoré sú webovou aplikáciou využívané sa nachádzajú v balíčku *Controllers*.

5.8.1 Komunikácia medzi JSP stránkou a Java Servletom

V prípade, že sa jedná o výsledkovú požiadavku je nutné, aby JSP stránka reagovala na výsledok získaný zo serveru. Na vytvorenie požiadavky je využitá webová technológia AJAX, ktorá umožňuje meniť jednotlivé obsahové časti HTML stránky bez nutnosti znovunačítania stránky webovej aplikácie.

Obrázok 5.2: Príklad definovania požiadavky na vykonanie POST metódy.

V niektorých prípadoch je pri vytváraní požiadavky nevyhnutné špecifikovať parameter *action*, ktorý bližšie určuje požadovaný úkon. Každý Java Servlet, ktorý umožňuje použitie parametru *action* definuje typ Enum, ktorý obsahuje jednotlivé hodnoty *action* parametru (použitie parametru môžeme vidieť aj na Obrázku 5.2). Z pohľadu servera je potrebné definovať formát výsledku, ktorý ponúka JSP stránke. Každý Java Servlet, ktorý vracia výsledok na nejakú požiadavku definuje verejné statické atribúty, ktorých hodnota reprezentuje stav výsledku. V prípade ak server poskytol dáta ako výsledok, môže definovať oddeľovač dát, ktorý zabezpečuje korektnú definíciu poskytnutých údajov.

Využitie technológie AJAX je pre našu webovú aplikáciu prínosné, pretože pri každej požiadavke na vytvorenie, editáciu, či zmazanie dát nemusí byť celý obsah HTML stránky znovu načítaný, a teda nie je nutné získavanie potrebných dát nanovo. Pozmenený je iba úsek, ktorý obsahoval dané dáta. Ostatné časti ostávajú nezmenené.

5.9 Zasielanie emailov

Pri vytvorení nového pracovníka typu *picker* a taktiež pri aktualizácii metadát existujúceho pracovníka je webovou aplikáciou vykonaný proces odoslania emailu danému pracovníkovi. O odoslanie emailov sa stará trieda *EmailManager*. Tento proces je vykonávaný v metóde *sendEmail*, kde využitím objektu *Properties* definujeme konfiguračné parametre emailového poskytovateľa. Následne je na základe predošlého objektu vytvorená inštancia triedy *Session*, ktorá je poskytnutá ako parameter konštruktoru triedy *MimeMessage*. Po získaní inštancie triedy *MimeMessage* nám tento objekt slúži k definovaniu všetkých vlastností emailovej správy, ako sú nastavenie odosielateľa, príjemcu, predmetu správy, textu tela správy a mnoho ďalších. Zároveň nám umožňuje získanie implementácie transportného protokolu použitím metódy *getTransport*. Posledným krokom je nastavenie spojenia potrebného pre autentifikáciu odosielateľa. Po jej ukončení je vykonané samotné odoslanie emailovej správy pomocou objektu triedy *Transport*.

5.10 Export odovzdávkových dát

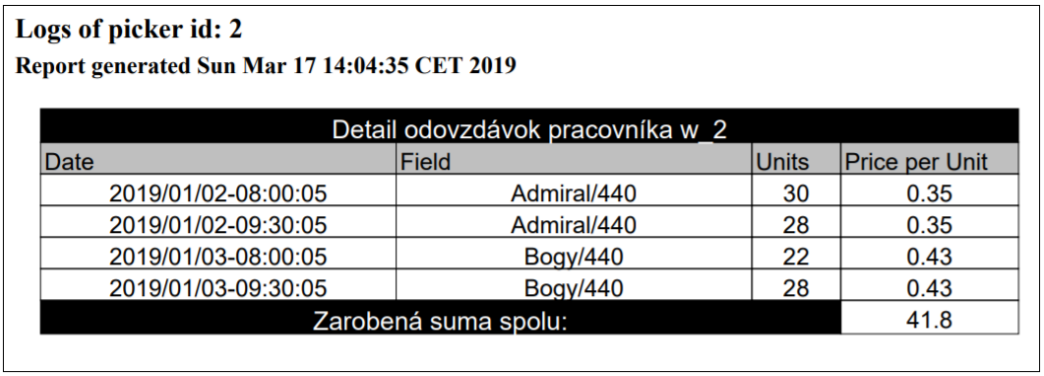
Webová aplikácia poskytuje funkciu exportu odovzdávkových dát jednotlivých pracovníkov. Formát výstupného súboru exportovaných dát je PDF, ktorý zabezpečuje konzistenciu obsahu naprieč zariadeniami. Na vytvorenie PDF dokumentu v aplikácií je využitá knižnica *iText*, pomocou ktorej je vytvorená PDF reprezentácia odovzdávkových dát. Následne je spustený proces sťahovania PDF súboru.

5.10.1 Tvorba PDF dokumentu

Vytvorenie PDF dokumentu je riadené triedou *ExportToPDF*, ktorá zabezpečuje získanie dát pre export a zápis daných dát do PDF súboru. Základnými jednotkami, ktoré sa podieľajú na vytvorení PDF súboru sú inštancia triedy *Document,* nesúca obsahové informácie dokumentu a objekt reprezentujúci *OutputStream,* do ktorého sa vykonáva zápis. Po korektnom vytvorení sú tieto objekty poskytnuté ako parametre statickej metóde *getInstance* triedy *PDFWriter*, ktorá zabezpečuje prepojenie medzi danými objektami. Hlavnou charakteristikou zmieneného prepojenia je, že každá zmena vykonaná na objekte inštancie triedy *Document* je prenesená na objekt reprezentujúci *OutputStream*. K definovaniu výzoru PDF dokumentu následne slúži vytvorená inštancia triedy *Document*, na ktorej definujeme jednotlivé komponenty, ktoré majú byť v PDF dokumente obsiahnuté.

5.10.2 Výzor PDF dokumentu

PDF dokument obsahuje informácie o identifikačnom čísle pracovníka a odovzdávkových dátach, ktoré sú reprezentované dátumom vytvorenia, polom, a počtom odovzdaných jednotiek na konkrétneho pracovníka. Ďalšou informáciou, ktorú môže PDF dokument poskytovať je cena za jednotku na poliach, na ktorých pracovník vykonával zber spolu s výsledným zárobkom pracovníka, ktorý je počítaný zo všetkých odovzdávkových dát.



Logs of picker id: 2

Report generated Sun Mar 17 14:04:35 CET 2019

Detail odovzdávok pracovníka w_2			
Date	Field	Units	Price per Unit
2019/01/02-08:00:05	Admiral/440	30	0.35
2019/01/02-09:30:05	Admiral/440	28	0.35
2019/01/03-08:00:05	Bogy/440	22	0.43
2019/01/03-09:30:05	Bogy/440	28	0.43
Zarobená suma spolu:			41.8

Obrázok 5.3:Ukážka výzoru PDF dokumentu

Pri definovaní tabuľky PDF dokumentu pracujeme s inštanciou triedy *PdfPTable*. Pri vytvorení poskytujeme inštancii informáciu o počte stĺpcov, ktoré tabuľka obsahuje. Následne použitím metódy *addCell* na objekte reprezentujúcom tabuľku je umožnené pridať novú bunku, ktorá je reprezentovaná inštanciou triedy *PdfPCell*.

5.10.3 Stiahnutie PDF dokumentu

Proces získania PDF dokumentu zo serveru je zabezpečený korektným nastavením troch metód na objekte typu *HttpServletResponse*. Jednou z metód, ktorú je nutné nastaviť je *setHeader*, ktorá ako argument požaduje názov hlavičky spolu s jej hodnotou. Názov hlavičky je v našom prípade textový reťazec *Content-Disposition*, spolu s hodnotou *attachment; filename="fileName.pdf"*, ktorá nastavuje informáciu o pripojenom súbore. Touto definíciou hodnoty hlavičky nastavujeme požiadavku na webový prehliadač o stiahnutie požadovaného súboru [18]. Ďalej je nutné definovať typ súboru *application/pdf* použitím metódy *setContentType.* Tento typ súboru oboznamuje webový prehliadač o reprezentovaní pripojenej jednotky pomocou PDF dokumentu. Poslednou metódou je *setContentLength*, ktorej ako parameter zadávame veľkosť súboru. Po korektnom nastavení spomenutých metód sú jednotlivé binárne dáta reprezentujúce PDF dokument zaslané do webového prehliadača pomocou objektu získaného metódou *getWriter* z objektu typu *HttpServletResponse*.

6 Záver

Vytvorili sme sadu aplikácií, ktoré sa podieľajú na udržiavaní a správe dát. Analyzovali sme výber jednotlivých technológií, ktoré sú v jednotlivých aplikáciách použité, a ktoré zabezpečujú poskytovanú funkčnosť a konzistenciu dát naprieč aplikáciami. Špecifikovali sme zvolenú štruktúru jednotlivých aplikácií, pomocou ktorej sme docielili, že mobilná aplikácia umožňuje prácu s dátami v režime off-line, a teda nie je nutné, aby počas práce s aplikáciou malo mobilné zariadenie neobmedzený prístup k internetu. Internetové pripojenie je potrebné iba pre synchronizáciu dát mobilnej aplikácie s hlavnou databázou. Definovaním jednotlivých funkcií poskytovaných výslednými aplikáciami sme ponúkli užívateľom širokú škálu možností ako s uloženými dátami pracovať pomocou jednoduchého užívateľského rozhrania. Definovali sme právo prístupu do webovej aplikácie v závislosti na pridelenej užívateľskej role, a tak sme umožnili, aby s webovou aplikáciou mohli narábať viacerí užívatelia, každý v zmysle pridelených práv.

Testovanie

Všetky aplikácie prešli viacerými testovacími scenármi, ktoré testovali ich funkčnosť. Mobilná aplikácia bola testovaná pomocou emulátoru na viacerých typoch zariadení, a zároveň bola testovaná na fyzických zariadeniach Samsung Galaxy J3 a Samsung Galaxy A3. Synchronizačná jednotka spolu s webovou aplikáciou boli testované na počítačoch s operačným systémom Windows.

Zoznam použitých zdrojov

[1] Statista. Global market share held by smartphone operating systems 2009-2018. *https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems*, 2018. [Online; cit. 2019-02-03].

[2] LACKO, Ľuboslav. *Mistrovství – Android*. Preložil Martin HERODEK. Brno: Computer Press, 2017. Mistrovství. ISBN 978-80-251-4875-4

[3] Android Open Source Project. Dashboards. *https://developer.android.com/about/dashboards*, 2018. [Online; cit. 2019-02-05]

[4] SQLite. 35% Faster Than The Filesystem. *https://www.sqlite.org/fasterthanfs.html*, 2017. [Online; cit. 2019-02-09]

[5] Android Open Source Project. Build a Responsive UI with ConstraintLayout. *https://developer.android.com/training/constraint-layout*, 2016. [Online; cit. 2019-02-07]

[6] Android Open Source Project. Optimizing Layout Hierarchies. *https://developer.android.com/training/improving-layouts/optimizing-layout*, 2011. [Online; cit. 2019-02-07]

[7] Statista. Global market share held by operating systems for desktop PCs 2013-2019. *https://www.statista.com/statistics/218089/global-market-share-of-windows-7*, 2019. [Online; cit. 2019-02-11]

[8] Mono project. Compatibility. *https://www.mono-project.com/docs/about-mono/compatibility*, 2019. [Online; cit. 2019-02-10]

[9] Tiobe. Tiobe Index for February 2019. *https://www.tiobe.com/tiobe-index*, 2019. [Online; cit. 2019-02-18]

[10] Java™ Platform, Standard Edition 7 API Specification. Class java.util.Properties. *https://docs.oracle.com/javase/7/docs/api/java/util/Properties.html*, 2010. [Online; cit. 2019-01-05]

[11] Statista. Global marker share held by internet browsers 2012-2018. *https://www.statista.com/statistics/268254/market-share-of-internet-browsers-worldwide-since-2009*, 2019. [Online; cit. 2019-02-12]

[12] Ing. František Kučera. Java na serveru: porovnání Javy a PHP. *https://blog.frantovo.cz/c/333/Java%20na%20serveru%3A%20porovn%C3%A1n%C3%AD%20Javy%20a%C2%A0PHP*, 2015. [Online; cit. 2019-02-20]

[13] Android Open Source Project. Guideline. *https://developer.android.com/reference/android/support/constraint/Guideline*, 2016. [Online; cit. 2019-02-07]

[14] Android Open Source Project. Save data in a local database using Room. *https://developer.android.com/training/data-storage/room*, 2012. [Online; cit. 2019-02-25]

[15] Java™ Platform, Standard Edition 7 API Specification. Interface java.sql.Statement. *https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html*, 2018. [Online; cit. 2019-01-05]

[16] Java™ Platform, Standard Edition 7 API Specification. Interface java.sql.PreparedStatement. *https://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html*, 2018. [Online; cit. 2019-01-05]

[17] W3Schools. SQL Injection. *https://www.w3schools.com/sql/sql_injection.asp*, n.d. [Online; cit. 2019-02-20]

[18] MDN Web Docs. Content-Disposition. *https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Disposition*, 2019. [Online; cit. 2019-03-01]

[19] Wikipedia, The Free Encyclopedia. WAR (file format). *https://en.wikipedia.org/wiki/WAR_(file_format)*, 2018. [Online; cit. 2019-03-10]

Zoznam použitých skratiek

UI	User Interface
API	Application Programming Interface
HTML	Hypertext Markup Language
XHTML	Extensible Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
JRE	Java Runtime Environment
XML	Extensible Markup Language
JDBC	Java Database Connectivity
JSP	JavaServer Pages
SQL	Structured Query Language
PDF	Portable Document Format
APK	Android application package
URL	Uniform Resource Locator
WAR	Web Application Resource

Prílohy

Súčasťou tejto práce je elektronická príloha, ktoré obsahuje spustiteľné súbory výsledných aplikácií spolu so zdrojovými kódmi a vygenerovanou dokumentáciou. Elektronická príloha taktiež obsahuje SQL súbory obsahujúce SQL príkazy na vytvorenie záznamov tabuliek, ktoré slúžia ako testovacie dáta.

- **/strawPick** priečinok obsahuje spustiteľný súbor, zdrojové kódy a vygenerovanú dokumentáciu mobilnej aplikácie strawPick
- **/strawSync** priečinok obsahuje spustiteľný súbor, zdrojové kódy a vygenerovanú dokumentáciu synchronizačnej jednotky strawSync
- **/strawManager** priečinok obsahuje spustiteľný súbor, zdrojové kódy a vygenerovanú dokumentáciu webovej aplikácie strawManager
- **/testsData** priečinok obsahuje SQL súbory, ktoré vytvárajú testovacie záznamy do tabuliek databázy
- **README** súbor obsahuje postup pre spustenie jednotlivých aplikácií ako celok

A. Užívateľská dokumentácia mobilnej aplikácie

Mobilná aplikácia strawPick slúži k zaznamenávaniu odovzdávkových dát jednotlivých pracovníkov a k synchronizácií získaných dát s dátami hlavnej databázy.

Odporúčanie

Pre správny a efektívny chod aplikácie sa požaduje, aby bola synchronizácia mobilnej aplikácie s hlavnou databázou vykonávaná dvakrát denne a to pred začatím práce na poli a po ukončení práce na danom poli. Tým nevzniknú veľké pamäťové nároky v mobilnom zariadení na uchovávanie dát, ktoré už viac nie sú v mobilnej aplikácií potrebné. Zároveň nevznikne situácia, kedy by boli funkcie mobilnej aplikácie degradované z dôvodu neexistencie všetkých potrebných dát, ktoré sa podieľajú na správnosti poskytovaných funkcií.

A.1 Sprievodca inštaláciou

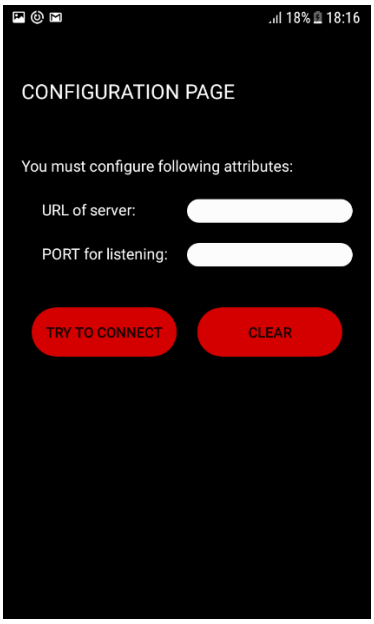
Mobilná aplikácia môže byť spustená na zariadeniach s operačným systémom Android 4.0.3 a vyšším. Aplikácia vyžaduje ručnú inštaláciu na mobilnom zariadení využitím inštalačného súboru formátu APK. Postup pre korektnú inštaláciu aplikácie je nasledovný:

- povoliť inštaláciu aplikácií získaných z neznámych zdrojov na mobilnom zariadení (Nastavenie → Zamknutie a zabezpečenie → Neznáme zdroje),
- presunúť inštalačný súbor .APK do mobilného zariadenia,
- v mobilnom zariadení spustiť inštalačný súbor .APK a potvrdiť inštaláciu zvolenej aplikácie.

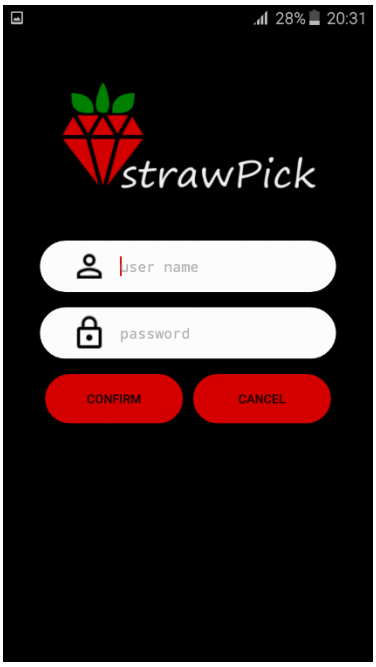
Po vykonaní popisovaných krokov je aplikácia úspešne nainštalovaná do mobilného zariadenia a pripravená k použitiu.

A.2 Úvodná stránka aplikácie

Pri prvom spustení je zobrazená konfiguračná stránka, ktorá umožňuje užívateľovi definovať konfiguračné parametre potrebné pre úspešné nadviazanie spojenia so synchronizačnou jednotkou. Pomocou tohto nastavenia je mobilná aplikácia inicializovaná a umožní tak prístup k aplikácií definovaným užívateľom. Vyplnením potrebných parametrov, ktoré špecifikujú URL adresu synchronizačnej jednotky a port, na ktorom má aplikácia so synchronizačnou jednotkou komunikovať, je po stlačení tlačidla 'Try To Connect' vykonaný pokus o nadviazanie spojenia. V prípade, že pomocou poskytnutých hodnôt nadviazala mobilná aplikácia spojenie so synchronizačnou jednotkou, je užívateľ presmerovaný na prihlasovaciu stránku aplikácie. V opačnom prípade je zobrazená informácia o tom, že sa nepodarilo nadviazať spojenie, a konfiguračná stránka zostáva naďalej zobrazená.



Obrázok A.1: Konfiguračná stránka mobilnej aplikácie

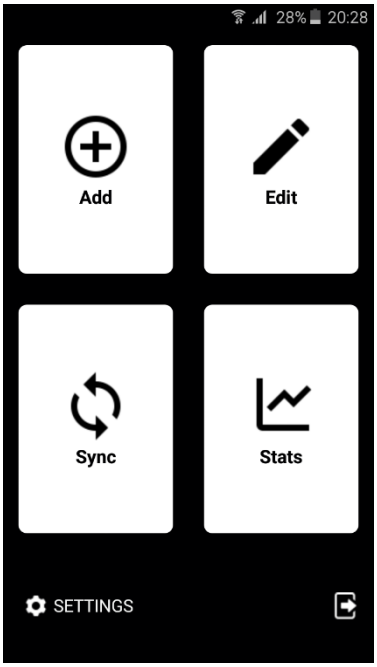


Obrázok A.2: Prihlasovacia stránka mobilnej aplikácie

Po úspešnom nadviazaní spojenia so synchronizačnou jednotkou je pri každom opätovnom otvorení mobilnej aplikácie zobrazená prihlasovacia stránka (v prípade, že užívateľ nie je prihlásený). Tá slúži ako miesto pre definovanie prihlasovacích údajov užívateľa, žiadajúceho o sprístupnenie funkcií mobilnej aplikácie. Po poskytnutí prihlasovacích údajov je kliknutím na tlačidlo 'Confirm' spustený proces autentifikovania užívateľa. V prípade korektných prihlasovacích údajov je užívateľ presmerovaný na navigačný panel aplikácie. V opačnom prípade je užívateľ oboznámený, že poskytnuté prihlasovacie údaje sú chybné, a prihlasovacia stránka zostane zobrazená.

A.3 Navigačný panel aplikácie

Úvodná stránka po prihlásení slúži ako navigačný panel pre užívateľa, ktorý ma možnosť kliknutím na jednotlivé ikony zobraziť stránku poskytujúcu jednu z funkcií mobilnej aplikácie. Tento navigačný panel taktiež umožňuje prekliknutie sa do sekcie nastavení, kde je užívateľovi zobrazená konfiguračná stránka mobilnej aplikácie. V prípade, že užívateľ danú stránku opustí, alebo vykoná pokus o nadviazanie spojenia, ktoré je neúspešné, konfiguračné parametre zostanú nezmenené. Kliknutím na ikonu opustenia aplikácie je prihlásený užívateľ odhlásený z aplikácie a je presmerovaný na prihlasovaciu stránku.

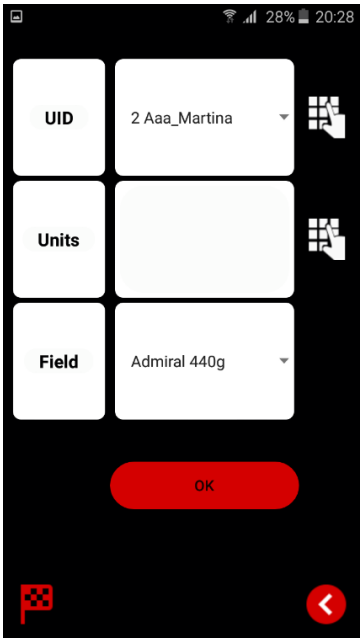


Obrázok A.3: Navigačný panel mobilnej aplikácie

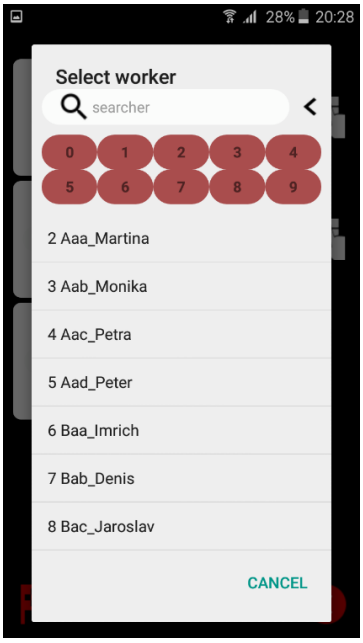
A.4 Vytvorenie odovzdávkových dát

Mobilná aplikácia poskytuje možnosť vytvárania odovzdávkových dát jednotlivých pracovníkov. Pre zobrazenie stránky, ktorá túto funkciu sprístupňuje je potrebné kliknúť na ikonu ‘Add’ v navigačnom paneli mobilnej aplikácie. Stránka pre vytvorenie odovzdávkových dát obsahuje: zoznam všetkých pracovníkov, ktorí sú vedení ako zberači; zoznam polí, na ktorých sa zber môže uskutočňovať; počet jednotiek, ktoré sú odovzdané. Definovaním všetkých spomenutých atribútov a nastavením na korektnú hodnotu je pomocou tlačidla ‘Ok’ vytvorený odovzdávkový údaj pracovníka. V prípade veľkého počtu pracovníkov je užívateľovi umožnené rýchle vyhľadávanie pracovníka použitím filtračného dialógového okna (viz Obrázok A.5). Užívateľ môže taktiež využiť možnosť zadávania počtu jednotiek bez použitia klávesnice, pomocou kruhového výberu (viz Obrázok A.6).

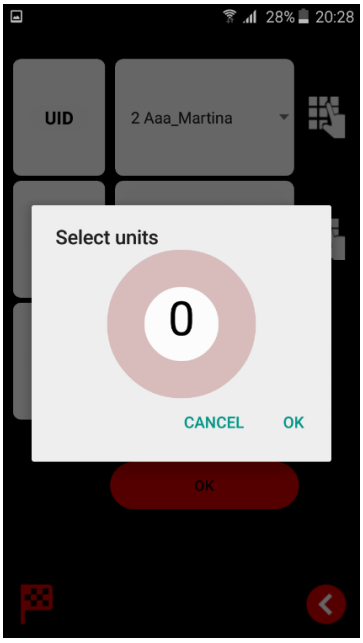
Stránka pre definovanie odovzdávkových dát, po kliknutím na ikonu vlajky, poskytuje možnosť vytvorenia štartovných dát jednotlivých pracovníkov, ktoré nesú informáciu o tom, kedy jednotliví pracovníci zahájili zber na danom poli (nevyhnutné pre využitie funkcie automatickej generácie ceny polí).



Obrázok A.4: Stránka pre definovanie odovzdávkových dát

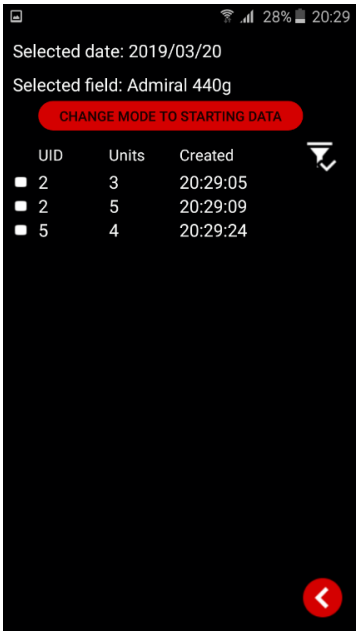


Obrázok A.5: Filtračný dialóg pre výber pracovníka



Obrázok A.6: Možnosť kruhového výberu počtu jednotiek

A.5 Editačná stránka mobilnej aplikácie



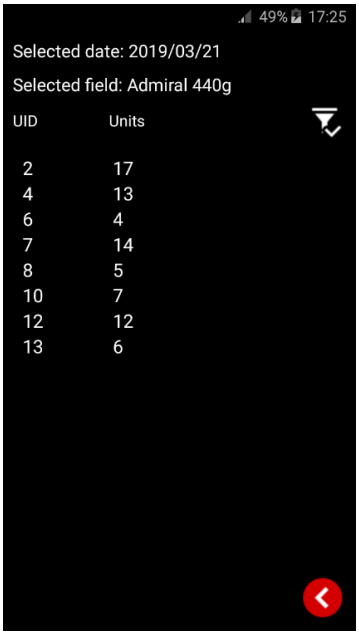
Obrázok A.7: Editačná stránka mobilnej aplikácie

Ďalšia funkcia, ktorá je poskytovaná mobilnou aplikáciou je editácia vytvorených odovzdávkových dát. Táto funkcia je užívateľovi sprístupnená po kliknutí na ikonu ‘Edit’ v navigačnom paneli. V prípade, že aplikácia obsahuje odovzdávkové dáta, ktoré nemajú spoločný dátum vytvorenia, prípadne spoločné pole, je užívateľovi najprv zobrazené menu pre špecifikovanie výberu požadovaného dňa a poľa, z ktorých chce odovzdávkové dáta upravovať. Editačná stránka mobilnej aplikácie poskytuje možnosť zmazania jednotlivých odovzdávkových dát, prípadne zmazania štartovných dát, ktoré sú zobrazené kliknutím na tlačidlo ‘Zmena módu’.

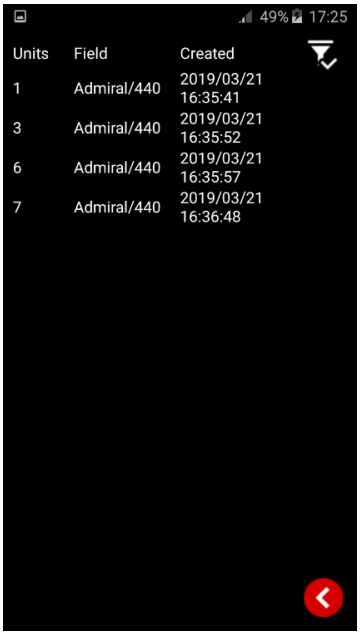
A.6 Štatistická stránka mobilnej aplikácie

Štatistická stránka mobilnej aplikácie umožňuje zobrazovať štatistické údaje získané z odovzdávkových dát jednotlivých pracovníkov spôsobom zobrazovania sumy všetkých odovzdaných jednotiek jednotlivých pracovníkov. Umožňuje tak

získanie informácií o pracovnom výkone zberačov (pred zobrazením štatistickej stránky je potrebné definovať výber dňa a poľa, z ktorých chceme štatistické dáta zobraziť). Klikom na jednotlivé riadky, ktoré reprezentujú štatistický údaj pracovníka sa zobrazí stránka detailu, ktorá obsahuje zoznam všetkých odovzdávkových dát, z ktorého bol štatistický údaj vytvorený. Štatistická stránka mobilnej aplikácie je zobrazená po kliknutí na ikonu ‘Stats’ v navigačnom paneli.



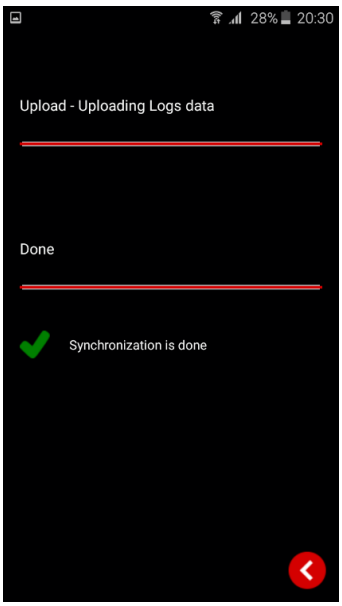
Obrázok A.8: Štatistická stránka mobilnej aplikácie



Obrázok A.9: Detail štatistického údaju

A.7 Synchronizačná stránka aplikácie

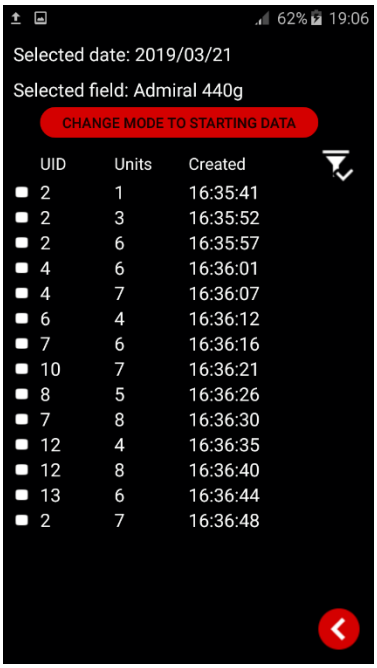
Mobilná aplikácia poskytuje možnosť synchronizácie dát mobilnej aplikácie s dátami hlavnej databázy. Synchronizačná stránka je zobrazená po kliknutí na ikonu ‘Sync’ v navigačnom paneli. Následne pomocou tlačidla ‘Synchronizovať’ začne proces synchronizácie. Aby bola synchronizácia úspešná, je potrebné, aby malo mobilné zariadenie prístup k internetu, a aby konfiguračné parametre, ktoré slúžia k nadviazaniu spojenia so synchronizačnou jednotkou boli správne nastavené. O výsledku synchronizačného procesu je užívateľ po skončení informovaný.



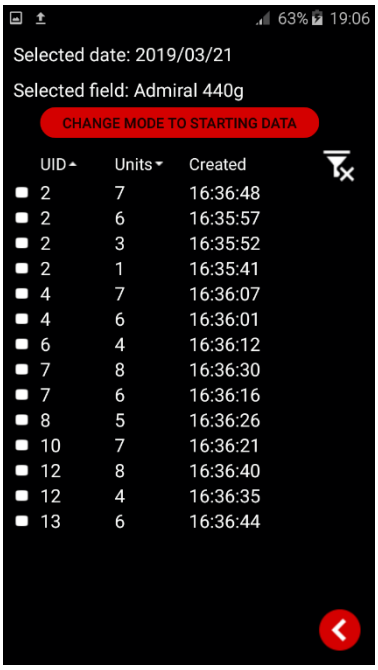
Obrázok A.10: Synchronizačná stránka mobilnej aplikácie

A.8 Radenie dát mobilnej aplikácie

Jednotlivé sekcie mobilnej aplikácie umožňujú radiť zobrazené dáta na základe vybraných atribútov. Atribúty, na základe ktorých je umožnené zobrazené dáta radiť zodpovedajú jednotlivým názvom stĺpcov. V prípade, že chceme atribút zaradiť do podmienky na radenie zobrazených dát, je nutné na atribút kliknúť. Tým sa označí radenie daného atribútu a nastaví sa zostupné poradie zobrazených dát podľa daného atribútu. V prípade, že chceme zmeniť poradie zobrazených dát podľa vybraného radiaceho atribútu, je nutné na atribút opätovne kliknúť, čím sa zmení poradie na vzostupné. Po kliknutí na daný atribút po tretí krát, je radenie na základe daného atribútu zrušené. Po korektnom nastavení radiacich atribútov spolu s ich definovaným poradím na zobrazené dáta, je nutné kliknúť na ikonu aplikovania radiacich atribútov. Následne sa poradie dát zmení na základe vybraných radiacich podmienok. V režime, keď sú radiace atribúty aplikované ich nie je možné zmeniť. Zmena je umožnená až po tom, čo je zrušené radenie dát kliknutím na ikonu zrušenia radiacich atribútov.



Obrázok A.11: Editačná stránka mobilnej aplikácie pred definovaním radiacich atribútov



Obrázok A.12: Editačná stránka mobilnej aplikácie po definovaní a aplikácií radiacich atribútov

B. Užívateľská dokumentácia synchronizačnej jednotky

Synchronizačná jednotka strawSync slúži k inicializácii hlavnej databázy a na prenos dát medzi mobilnou aplikáciou a hlavnou databázou.

Predpokladom pre spustenie aplikácie je spojenie s MySQL. V prípade, že požadovaný databázový systém nie je doposiaľ inštalovaný, je nutné vykonať jeho inštaláciu[2]. Následne, po korektnej inštalácii a spustení MySQL je pre aplikáciu zabezpečená komunikácia s databázou.

B.1 Formát konfiguračného súboru

Konfiguračný súbor musí obsahovať všetky povinné atribúty, ktoré synchronizačná jednotka definuje spolu s ich hodnotami.

Zoznam všetkých povinných konfiguračných atribútov:

- **database.hostname** atribút nastavujúci názov databázového servera,
- **database.port** atribút nastavujúci číslo portu, na ktorom bude prebiehať komunikácia s databázou (podporované rozmedzie hodnoty portu je 0 až 65535),
- **database.username** atribút nastavujúci prístupové meno používateľa databázy,
- **database.password** atribút nastavujúci prístupové heslo používateľa databázy,
- **synchronization.port** atribút nastavujúci číslo portu, na ktorom bude prebiehať komunikácia s mobilnou aplikáciou (podporované rozmedzie hodnoty portu je 0 až 65535).

Mimo povinné konfiguračné parametre je umožnené definovať aj voliteľné.

Podporované voliteľné konfiguračné atribúty:

- **database.properties** atribút nastavujúci vlastnosti spojenia s MySQL relačným databázovým systémom,

[2] Postup pre korektnú inštaláciu MySQL databázového relačného systému viz externý odkaz: https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_HowTo.html

- **synchronization.logger** atribút nastavujúci cestu k súboru, do ktorého bude synchronizačná jednotka zapisovať riadiace správy komunikácie s mobilnou aplikáciou. V prípade že daný atribút nie je definovaný, funkcia vypisovania riadiacich výpisov aplikácie ostane nevyužitá.

```
# database configuration attributes
database.hostname=localhost
database.port=3306
database.username=iStraw_User
database.password=Chl4dn1ck4
# sync configuration attributes
synchronization.port=7529
synchronization.logger=outputNew.txt
```

Obrázok B.1: Príklad konfiguračného súboru synchronizačnej jednotky, v ktorom pomocou atribútu 'database.properties' nastavujeme časové pásmo

B.2 Spustenie aplikácie

Synchronizačná jednotka je konzolová aplikácia, a teda nemá žiadne definované užívateľské rozhranie, pomocou ktorého je umožnené s aplikáciou pracovať. Jediný krok potrebný pre fungovanie aplikácie je jej korektné spustenie z príkazového riadku. Pri spustení aplikácia vyžaduje zadanie jedného parametru '-conf' spolu s jeho hodnotou. Hodnota tohto parametru značí cestu k súboru, v ktorom sú špecifikované konfiguračné parametre synchronizačnej jednotky. V prípade, že pri inicializácii komponent programu, ktoré sa podieľajú na synchronizačnom procese nastane chyba, je užívateľ o tomto stave informovaný. V prípade že komponenty sú správne inicializované, je aplikácia korektne spustená a čaká na prichádzajúce spojenia s mobilnými aplikáciami.

V prípade že zadáme požiadavku na ukončenie aplikácie, pre korektné ukončenie je potrebné zadať reťazec 'quit' na príkazovom riadku spustenej aplikácie, ktorý zabezpečí ukončenie všetkých prebiehajúcich synchronizačných procesov a následne ukončí chod aplikácie.

B.3 Riadiace výpisy aplikácie

Funkcia riadiacich výpisov aplikácie je aplikovaná v prípade, že v konfiguračnom súbore bol špecifikovaný atribút *synchronization.logger*. Daná funkcia je následne využitá pri synchronizačných procesoch s mobilnou aplikáciou a zaznamenáva danú komunikáciu. Využitie danej funkcie je voliteľné, keďže nemá priamy dopad na chod

aplikácie, a teda má iba informatívny charakter. Každá riadiaca správa obsahuje informáciu o dátume a čase vloženia daného záznamu vo formáte *yyyymmdd_hhmmss*. Zoznam informácií, ktoré aplikácia zapisuje do súboru:

- nadviazanie spojenia s mobilnou aplikáciou,
- prijatie smerovacieho, potvrdzovacieho, prípadne ukončovacieho symbolu,
- odoslanie potvrdzovacieho, prípadne ukončovacieho symbolu,
- prijatie, odoslanie dát.

```
20190321_230137 Connection accepted: user iStraw_User
20190321_230137 Communicate with user iStraw_User: Received flag DOWNLOADDATA
20190321_230137 Communicate with user iStraw_User: Send information about count of all data to synchronize -> count=5
20190321_230137 Communicate with user iStraw_User: Received confirmation flag: Success
20190321_230137 Communicate with user iStraw_User: Send information about size of supervisors data to synchronize -> count=1
20190321_230137 Communicate with user iStraw_User: Send supervisor login data -> ID/Login/Pasword=1/iStraw_User/Chl4dn1ck4
20190321_230137 Communicate with user iStraw_User: Received confirmation flag: Success
20190321_230137 Communicate with user iStraw_User: Send information about size of workers data to synchronize -> count=2
20190321_230137 Communicate with user iStraw_User: Send worker data -> id/name/surname=2/Aaa/Martina
20190321_230137 Communicate with user iStraw_User: Send worker data -> id/name/surname=3/Aab/Monika
20190321_230137 Communicate with user iStraw_User: Received confirmation flag: Success
20190321_230137 Communicate with user iStraw_User: Send information about size of fields data to synchronize -> count=2
20190321_230137 Communicate with user iStraw_User: Send field data -> id/name/quoty=1/Admiral/440
20190321_230137 Communicate with user iStraw_User: Send field data -> id/name/quoty=2/Admiral/510
20190321_230137 Communicate with user iStraw_User: Received confirmation flag: Success
20190321_230137 Communicate with user iStraw_User: Received flag CLOSECONNECTION -> Connection successfully closed
```

Obrázok B.2: Príklad výstupného súboru vygenerovaného funkciou riadiacich výpisov synchronizačného procesu

C. Užívateľská dokumentácia webovej aplikácie

Webová aplikácia strawManager slúži k správe dát pracovníkov a poskytuje jednoduché užívateľské rozhranie pre prácu. Webová aplikácia disponuje užívateľskými rolami *admin*, *supervisor* a *picker*, ktoré definujú prístupové práva, čím umožňujú jednotlivým pracovníkom použitie poskytovaných funkcií aplikácie.

C.1 Spustenie aplikácie

Na spustenie webovej aplikácie slúži súbor formátu WAR používaný na distribúciu kolekcií súborov tvoriacich webovú aplikáciu, a ktorý je poskytnutý webovému serveru pri nasadení aplikácie na server. V adresári WEB-INF sa nachádza konfiguračný súbor, ktorého obsah je nutné definovať hodnotami parametrov.

Formát konfiguračného súboru

Konfiguračný súbor musí obsahovať všetky povinné atribúty, ktoré webová aplikácia definuje spolu s ich hodnotami. Korektné nastavenie daných atribútov je kľúčové, vzhľadom na funkčnosť webovej aplikácie.

```
# smtp configuration attributes
smtp.host=smtp.gmail.com
smtp.username=iStrawManager@gmail.com
smtp.password=iStr4wM4n4g3r
smtp.port=587
# database configuration attributes
jdbc.url=jdbc:mysql://localhost:3306/
jdbc.username=iStraw_User
jdbc.password=Ch14dn1ck4
```

Obrázok C.1: Príklad konfiguračného súboru webovej aplikácie

Zoznam všetkých povinných konfiguračných atribútov:

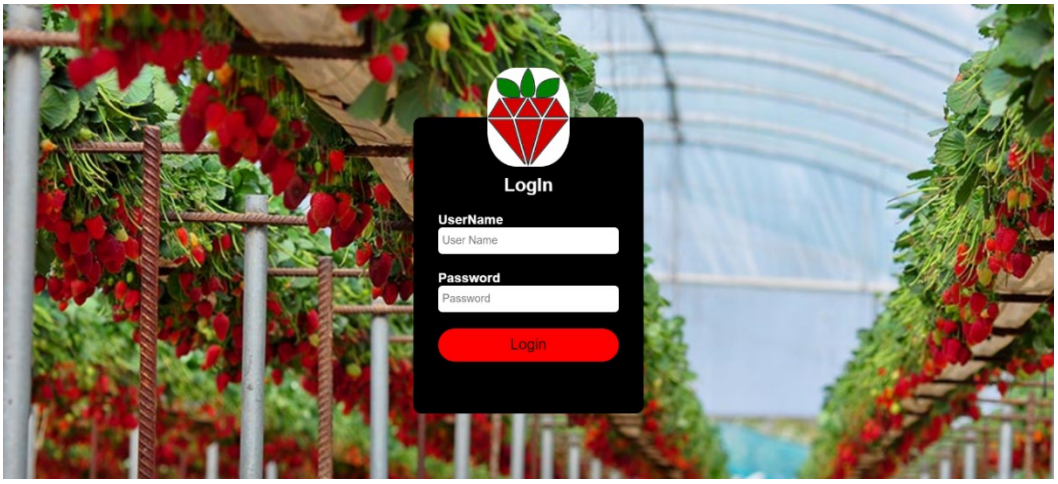
- **jdbc.url** atribút nastavujúci URL databázového servera,
- **jdbc.username** atribút nastavujúci prístupové meno používateľa databázy,
- **jdbc.password** atribút nastavujúci prístupové heslo používateľa databázy,
- **smtp.host** atribút nastavujúci meno emailového poskytovateľa,
- **smtp.port** atribút nastavujúci číslo portu pre komunikáciu s emailovým poskytovateľom (podporované rozmedzie hodnoty portu je 0 až 65535),
- **smtp.username** atribút nastavujúci prístupové meno emailového účtu,
- **smtp.password** atribút nastavujúci prístupové heslo emailového účtu.

Zoznam všetkých voliteľných konfiguračných atribútov:

- **jdbc.properties** atribút nastavujúci vlastnosti spojenia s databázou.

C.2 Prihlasovacia stránka webovej aplikácie

Na sprístupnenie jednotlivých funkcií, ktoré webová aplikácia ponúka je nutné poskytnúť prihlasovacie údaje, pomocou ktorých je užívateľ autentifikovaný. V prípade správnych prihlasovacích údajov je užívateľ presmerovaný na úvodnú stránku aplikácie.



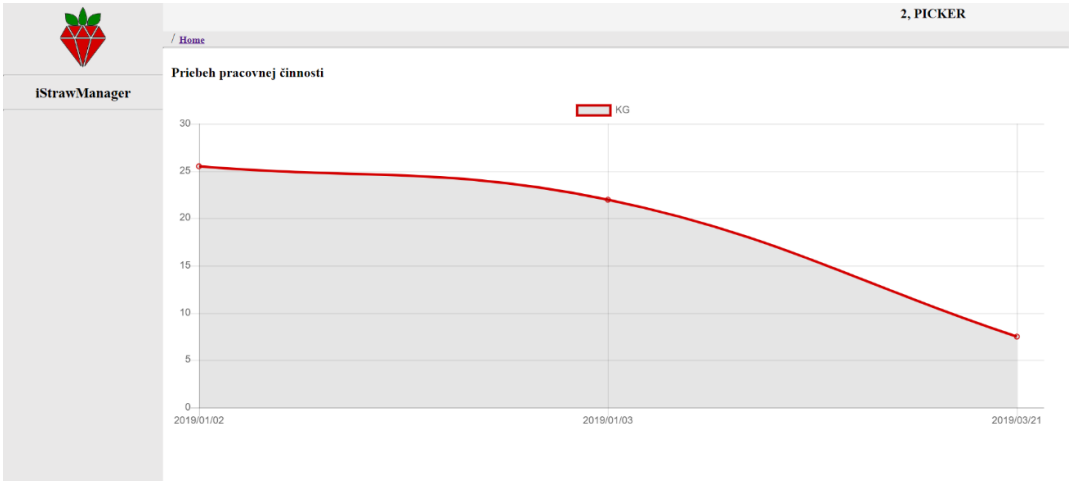
Obrázok C.2: Prihlasovacia stránka webovej aplikácie

C.3 Úvodná stránka webovej aplikácie

Úvodná stránka webovej aplikácie je zobrazená po korektnom prihlásení užívateľa. Podľa užívateľskej roly, ktorá je prihlásenému užívateľovi priradená, je definovaný typ obsahu úvodnej stránky. V prípade, že prihlásený užívateľ disponuje užívateľskou rolou *supervisor* alebo *admin*, obsah danej stránky predstavuje štatistické údaje o výkonnosti jednotlivých polí, ktoré sú graficky reprezentované.

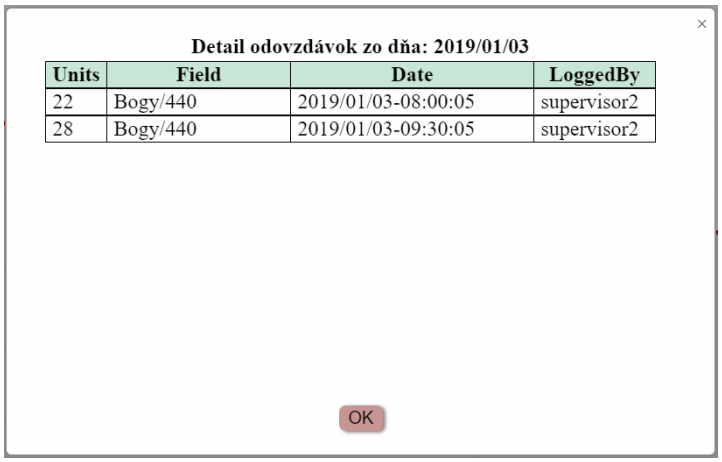
Obrázok C.31: Úvodná stránka webovej aplikácie užívateľa typu supervisor a admin

V opačnom prípade, kedy má prihlásený užívateľ priradenú užívateľskú rolu *picker*, úvodná stránka nesie štatistické údaje o jeho pracovnej výkonnosti využitím grafu (viz Obrázok C.4).



Obrázok C.4: Úvodná stránka webovej aplikácie po prihlásení užívateľom typu picker

Následnou interakciou s daným grafom je užívateľovi umožnené zobraziť detail požadovaného dňa, ktorý zodpovedá zoznamu všetkých odovzdávkových dát vytvorených v požadovanom termíne.



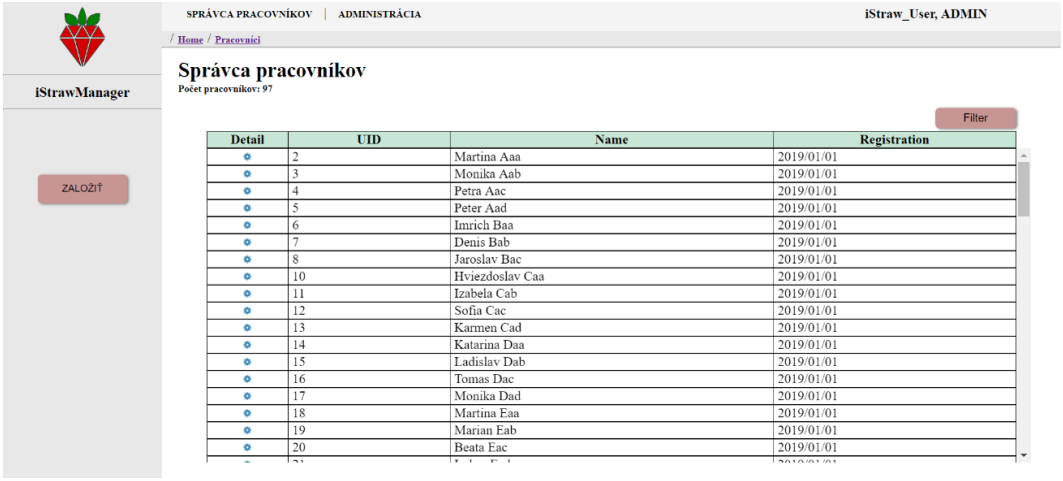
Obrázok C.5: Dialógové okno webovej aplikácie reprezentujúce detail odovzdávkových dát pracovníka

C.4 Správca pracovníkov

Sekcia poskytuje užívateľovi možnosť vytvárať, rušiť a editovať pracovníkov spolu s ich odovzdávkovými dátami. Prístupná je iba pre užívateľov s priradenou užívateľskou rolou *admin* alebo *supervisor*, pričom druhá zo spomenutých rolí má obmedzené, poprípade zamedzené použitie niektorých špecifických funkcií.

C.4.1 Prehľad pracovníkov

Kliknutím na položku ‘Správca pracovníkov’ v hornom menu webovej aplikácie je prihlásený užívateľ presmerovaný na stránku prehľadu pracovníkov.



Obrázok C.6: Úvodná stránka sekcie 'Správca pracovníkov'

Daná stránka poskytuje prihlásenému užívateľovi informáciu o počte všetkých pracovníkov, ktorí sú v aplikácii vedení ako zberači, spolu s tabuľkou, ktorej riadky reprezentujú jednotlivých pracovníkov (viz Obrázok C.6). Kliknutím na ozubené koliesko v stĺpci ‘Detail’ u riadku vybraného pracovníka, je prihlásený užívateľ presmerovaný na stránku detailu zvoleného pracovníka.

Filtrovanie dát

Webová aplikácia disponuje funkciou filtrovania dát, ktorá umožňuje prihláseným užívateľom vyfiltrovať zobrazené dáta špecifikovaním hodnôt predom definovaných atribútov. Filtračné atribúty sú užívateľovi zobrazené po kliknutí na tlačidlo ‘Filter’. Hodnoty daných atribútov sú po kliknutí na tlačidlo ‘Použiť filter’ aplikované na proces obmedzenia zobrazovaných pracovníkov.

Obrázok C.7: Filtračné atribúty, pomocou ktorých je umožnené filtrovať zobrazených pracovníkov

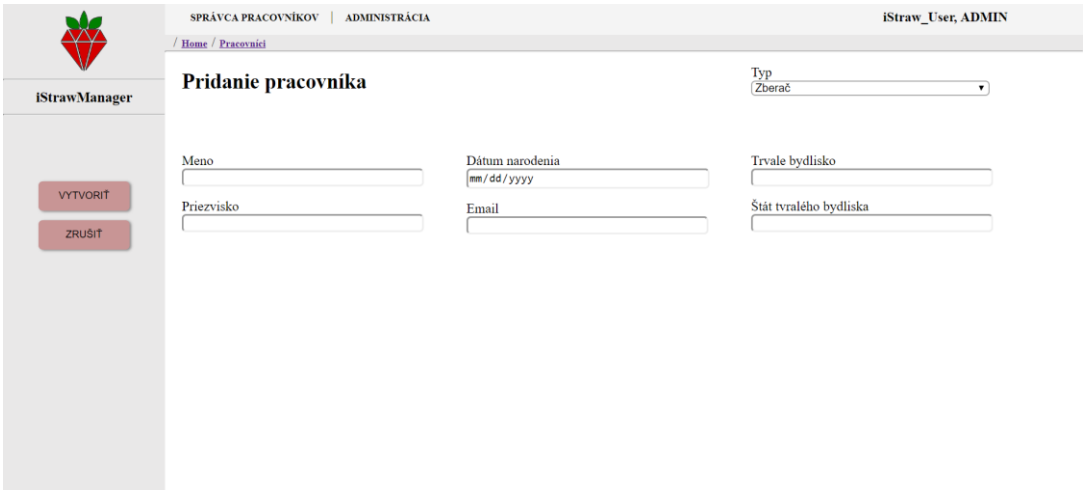
Filtračné atribúty, na základe ktorých je vykonávané filtrovanie dát:

- **UID -** identifikačné číslo pracovníka
- **Name -** meno a priezvisko pracovníka
- **Register -** dátum registrácie pracovníka

Využitie funkcie filtrovania dát je vhodné v situácií, kedy dôsledkom veľkého počtu pracovníkov nie je jednoduché sa v tabuľke orientovať, a tak daná funkcia môže ušetriť čas pri hľadaní.

C.4.2 Vytvorenie pracovníka

Prihlásený užívateľ s priradenou užívateľskou rolou *admin* má právo definovať nového pracovníka, a to kliknutím na tlačidlo ‘Založiť’, ktoré sa nachádza na úvodnej stránke sekcie ‘Správca pracovníkov’.



Obrázok C.8: Stránka definujúca vytvorenie nového pracovníka

Pri vytváraní pracovníka je potrebné vyplniť všetky povinné atribúty:

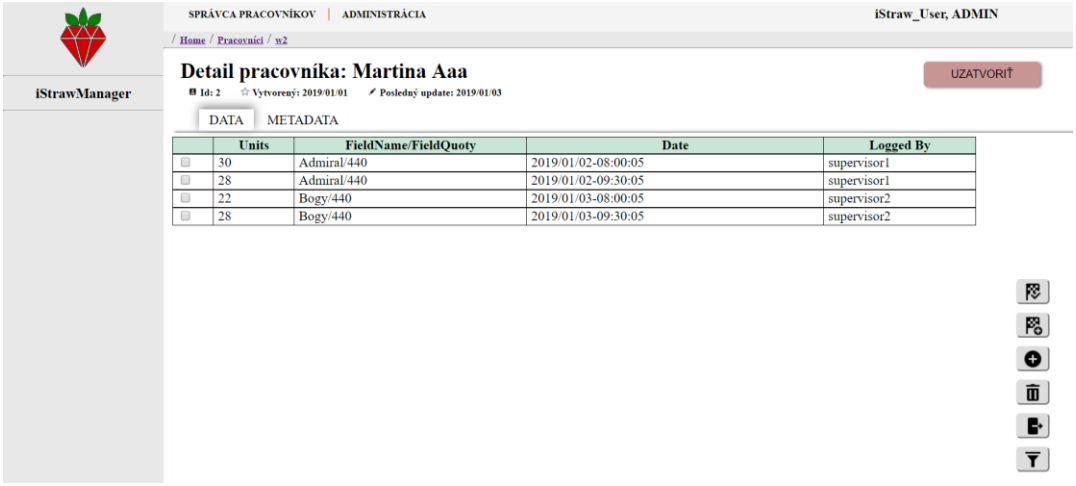
- meno pracovníka,
- priezvisko pracovníka,
- dátum narodenia pracovníka (podporovaný formát je *mm/dd/yyyy*),
- emailová adresa pracovníka, na ktorú budú zaslané prihlasovacie údaje do aplikácie (podporovaný formát je *meno@doména*),
- trvalé bydlisko,
- typ funkcie, ktorá bude pracovníkovi priradená (podporovaná hodnota je *zberač* a *rola*).

Tlačidlom ‘Vytvoriť’ potvrdíme vytvorenie nového pracovníka, prípadne tlačidlom ‘Zrušiť’ je proces vytvorenia pracovníka zrušený.

Pri vytváraní pracovníka sú kontrolované všetky atribúty spolu s ich vyplnenými hodnotami. Ak niektorý povinný atribút nie je vyplnený, alebo nemá podporovaný formát, nie je umožnené realizovať vytvorenie pracovníka.

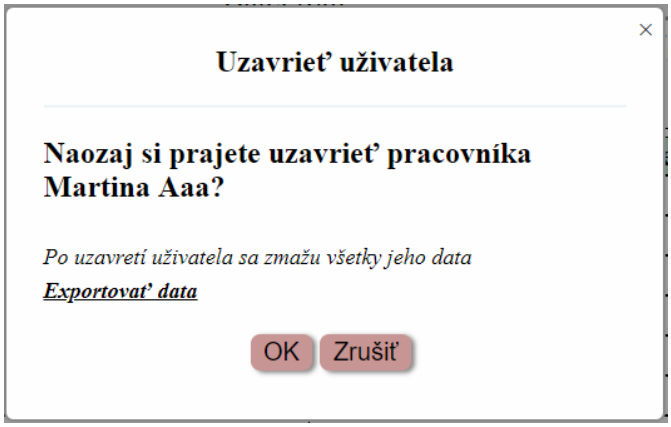
C.4.3 Detail pracovníka

Detail pracovníka zobrazuje informácie o danom pracovníkovi pomocou informačného panelu, kde môžeme vidieť identifikačné číslo pracovníka, dátum vytvorenia pracovníka, a dátum poslednej modifikácie odovzdávkových dát.
Daná stránka sa delí na dve sekcie: dáta a metadáta, medzi ktorými je umožnené sa pomocou záložiek pod informačným panelom preklikávať.



Obrázok C.9: Stránka detailu pracovníka

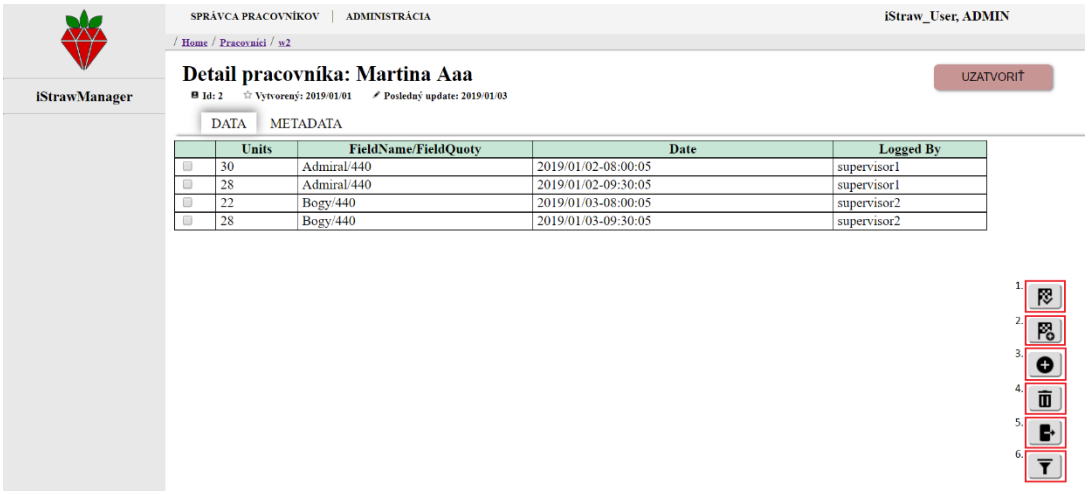
Tlačidlo ‘Uzatvoriť’ je zobrazené iba v prípade, že prihlásený užívateľ disponuje užívateľskou rolou *admin*, a slúži k odstráneniu pracovníka spolu so všetkými záznamami asociovanými s pracovníkom. Kliknutím na dané tlačidlo sa zobrazí potvrdzovací dialóg.



Obrázok C.10: Potvrdzovací dialóg uzatvorenia pracovníka

C.4.3.1 Sekcia dát

Záložka ‘DATA’ zobrazuje odovzdávkové dáta daného pracovníka, a poskytuje radu funkcií, ktoré s danými dátami pracujú.



Obrázok C.11: Sekcia 'DATA' detailu pracovníka s vyznačenými funkciami: 1. definovanie nového štartovacieho údaju, 2. zobrazenie/skrytie štartovacích dát z tabuľky, 3. definovanie nového odovzdávkového údaju, 4. zmazanie označených odovzdávkových dát, 5. export odovzdávkových dát pracovníka, 6. filtrovanie zobrazených odovzdávkových dát.

Vytvorenie štartovacieho záznamu

Aplikácia poskytuje funkciu definovania nového štartovacieho záznamu. Kliknutím na ikonu definovania nového štartovacieho záznamu je zobrazený dialóg, ktorý slúži k doplneniu hodnôt štartovacieho údaju, ktorý má byť vytvorený.

Obrázok C.12: Dialóg definujúci nový štartovací záznam

Povinné atribúty dialógu:

- **Field** - názov poľa vo formáte *názov/gramáž*, na ktorom má byť pracovníkovi vytvorený štartovací záznam
- **Date** - dátum vo formáte *mm/dd/yyyy hh:mm*, s ktorým sa bude štartovací záznam asociovať

Jednotlivé tlačidlá potvrdzujú/zrušia proces vytvorenia nového štartovacieho záznamu.

Zobrazenie štartovacích dát

Prednastavený režim zobrazovaných dát je nastavený na odovzdávkové dáta, pričom užívateľ môže tento režim meniť v zmysle dodatočného zobrazenia štartovacích záznamov. Kliknutím na ikonu zobrazenia štartovacích záznamov sa štartovacie dáta pridajú do tabuľky odovzdávkových dát so žltým podfarbením a so znakom ‘S’ v stĺpci ‘Units’ daného záznamu. V prípade, že chceme štartovacie dáta skryť, je nutné opätovne na ikonu kliknúť.

	Units	FieldName/FieldQuoty	Date	Logged By
☐	S	Admiral/440	2019/01/02-07:00:05	supervisor1
☐	30	Admiral/440	2019/01/02-08:00:05	supervisor1
☐	28	Admiral/440	2019/01/02-09:30:05	supervisor1
☐	S	Bogy/440	2019/01/03-07:00:05	supervisor2
☐	22	Bogy/440	2019/01/03-08:00:05	supervisor2
☐	28	Bogy/440	2019/01/03-09:30:05	supervisor2
☐	1	Admiral/440	2019/03/21-16:35:41	iStraw_User
☐	3	Admiral/440	2019/03/21-16:35:52	iStraw_User
☐	6	Admiral/440	2019/03/21-16:35:57	iStraw_User
☐	7	Admiral/440	2019/03/21-16:36:48	iStraw_User

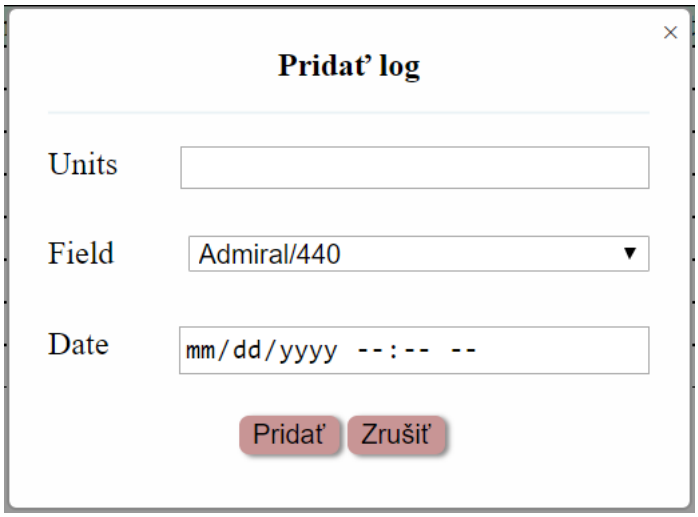
Obrázok C.13: Výzor tabuľky po zobrazení štartovacích záznamov

Zmazanie záznamov

V prípade, že prihlásený užívateľ zadáva požiadavku na zmazanie jednotlivých záznamov, či už odovzdávkových alebo štartovacích, môže použitím ikony smetného koša odstrániť označené záznamy. Po úspešnom zmazaní záznamov z databázy sú záznamy z tabuľky odstránené avšak všetky aplikované filtre ostanú zachované. V prípade, že záznamy sa nepodarilo odstrániť, užívateľ je o tomto stave informovaný.

Vytvorenie odovzdávkového záznamu

Ďalšou z poskytovaných funkcií je definovanie nového odovzdávkového záznamu. Kliknutím na ikonu nového odovzdávkového záznamu je zobrazený dialóg, ktorý slúži k doplneniu hodnôt odovzdávkového údaju, ktorý má byť vytvorený.



Obrázok C.14: Dialóg definujúci nový odovzdávkový záznam

Povinné atribúty dialógu:

- **Units -** počet odovzdaných jednotiek
- **Field -** názov poľa vo formáte *názov/gramáž*, na ktorom má byť pracovníkovi vytvorený odovzdávkový záznam
- **Date -** dátum vo formáte *mm/dd/yyyy hh:mm*, ku ktorému sa bude odovzdávkový záznam vzťahovať

Jednotlivé tlačidlá potvrdzujú/rušia proces vytvorenia nového odovzdávkového záznamu. V prípade, že jednotlivé povinné atribúty dialógu nie sú vyplnené, poprípade sú vyplnené nepodporovanými údajmi, nie je užívateľovi umožnené odovzdávkový záznam vytvoriť, a užívateľ je následne o tomto stave informovaný.

Export odovzdávkových dát

Aplikácia poskytuje možnosť exportu odovzdávkových dát pracovníka, ktorý je reprezentovaný pomocou PDF dokumentu. Proces exportu dát je zahájený kliknutím na ikonu exportu a zakončený následným stiahnutím vygenerovaného súboru obsahujúceho odovzdávkové dáta.

Logs of picker id: 2

Report generated Sat Mar 23 20:24:41 CET 2019

Detail odovzdávok pracovníka w_2			
Date	Field	Units	Price per Unit
2019/01/02-08:00:05	Admiral/440	30	0.35
2019/01/02-09:30:05	Admiral/440	28	0.35
2019/01/03-08:00:05	Bogy/440	22	0.43
2019/01/03-09:30:05	Bogy/440	28	0.43
Zarobená suma spolu:			41.8

Obrázok C.15: Príklad exportu odovzdávkových dát pracovníka

Informácie udržiavané vo vygenerovanom PDF dokumente:

- identifikačné číslo pracovníka,
- dátum a čas vygenerovania súboru,
- tabuľka, ktorá obsahuje odovzdávkové dáta a je definovaná stĺpcami ‘Date’, ‘Field’, ‘Units’ a ‘Price per Unit’. Hodnota v poslednom stĺpci je vyplnená iba v prípade, že cena za jednotku na danom poli v daný termín je stanovená. V opačnom prípade je hodnota nevyplnená a odovzdávkový záznam nie je započítaný do výsledného zárobku, ktorý je uvedený v záhlaví tabuľky.

Filtrovanie záznamov

V prípade veľkého počtu záznamov je výhodné využiť funkciu filtrovania, ktorá definovaním hodnôt daných atribútov môže obmedziť výsledné zobrazené dáta. Kliknutím na ikonu filtru sú zobrazené filtračné atribúty, ktorých špecifikovaním a následnou aplikáciou sa môže zúžiť výsledné množstvo zobrazených dát.

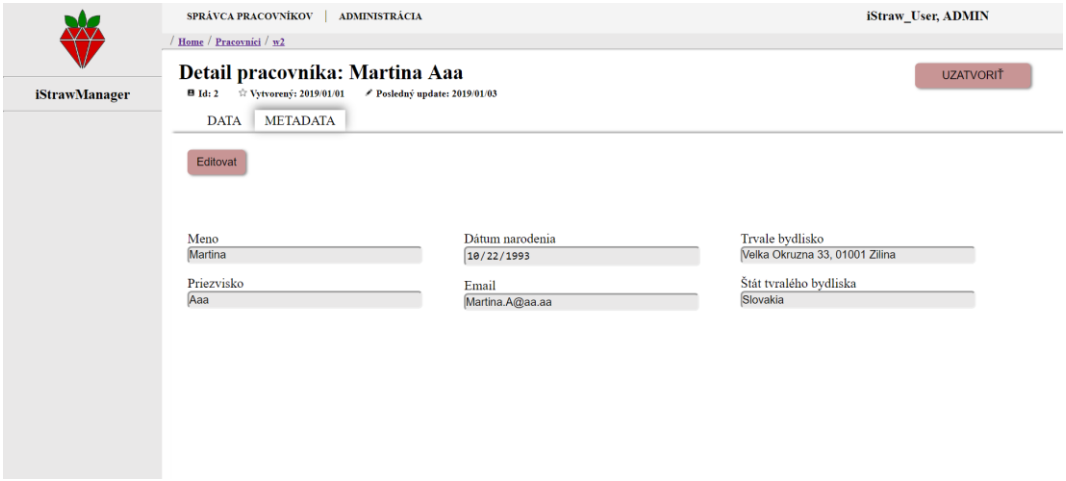
Obrázok C.16: Filtračné atribúty pre zúženie zobrazených dát

Filtračné atribúty, na základe ktorých je vykonávané filtrovanie dát:

- **LoggedBy** určuje meno užívateľa, ktorý vytvoril daný záznam,
- **DateFrom** špecifikuje dátum, od ktorého majú byť záznamy zobrazené,
- **DateTo** špecifikuje dátum, po ktorý majú byť záznamy zobrazené.

C.4.3.2 Sekcia metadát

Záložka ‘METADATA’ zobrazuje všeobecné informácie o pracovníkovi, ktoré sú totožné s atribútmi definovanými na stránke založenia nového pracovníka. Pre zmenu údajov o pracovníkovi je potrebné kliknúť na tlačidlo ‘Editovať’. Následne je pomocou jednotlivých tlačidiel umožnené vykonané zmeny uložiť alebo zahodiť.

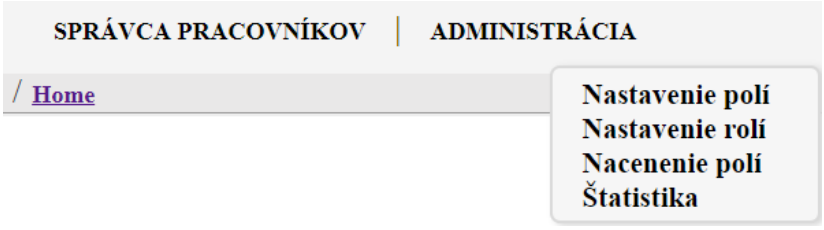


Obrázok C.172: Sekcia 'METADATA' detailu pracovníka

Sekcia je sprístupnená iba prihlásenému užívateľovi s priradenou užívateľskou rolou *admin*.

C.5 Administrácia

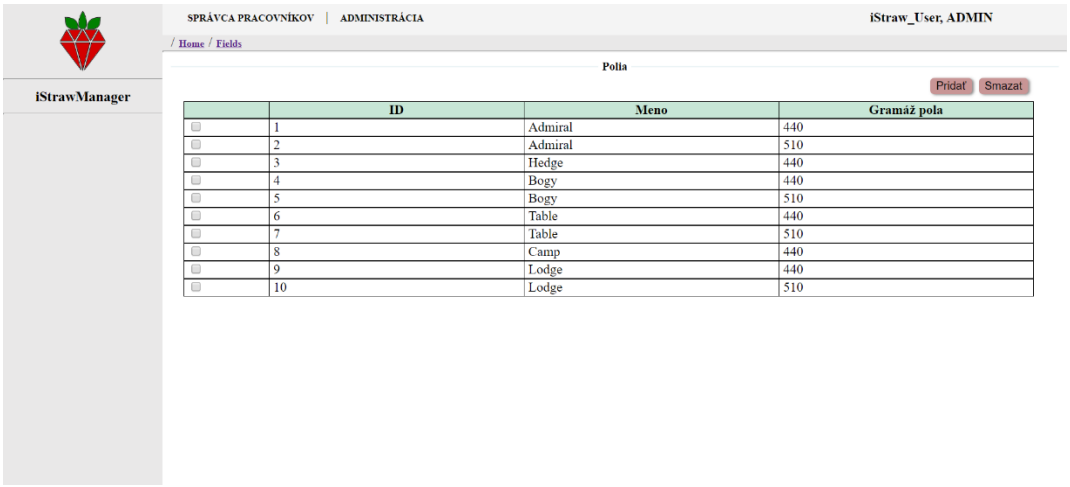
Funkcie, ktoré sú definované v sekcii administrácie sú sprístupnené iba prihláseným užívateľom disponujúcim užívateľskou rolou *admin*. Po kliknutí na položku ‘Administrácia’ v hornom menu webovej aplikácie je zobrazené rozbaľovacie menu obsahujúce jednotlivé stránky administrácie, na ktoré sa dá prekliknúť (viz Obrázok C.18).



Obrázok C.18: Zobrazenie rozbaľovacieho zoznamu po kliknutí na položku 'Administrácia'

C.5.1 Nastavenie polí

Aplikácia definuje stránku ‘Nastavenie polí’, ktorá umožňuje spravovať dáta reprezentujúce jednotlivé polia, na ktorých môže byť vykonávaný zber.



Obrázok C.19: Výzor stránky 'Nastavenie polí'

Vytvorenie nového poľa

Jednou z funkcií, ktoré stránka poskytuje je definovanie nového poľa. Kliknutím na tlačidlo ‘Pridať’ je zobrazený dialóg, ktorý definuje povinné atribúty vytváraného poľa.

Obrázok C.203: Dialóg definujúci nové pole

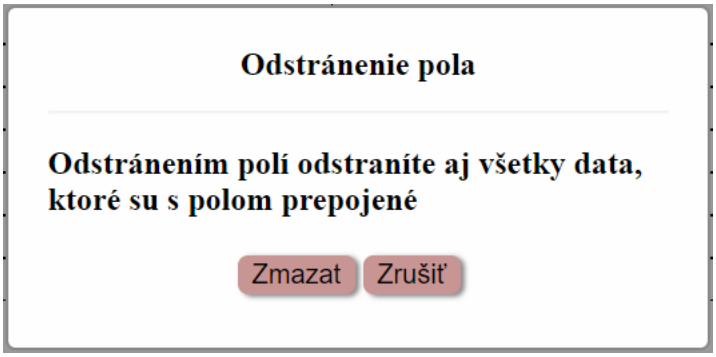
Povinné atribúty dialógu:

- **Meno -** meno vytváraného poľa
- **Gramáž -** gramáž jednotky na vytvorenom poli

Jednotlivé atribúty dialógu nemusia byť jedinečné, ale spoločne musia tvoriť jedinečný záznam. V prípade, že vytvárané pole je už v aplikácii definované, je užívateľovi zobrazená informačná správa o existencii vytváraného poľa a nie je povolené jeho vytvorenie.

Zmazanie existujúceho poľa

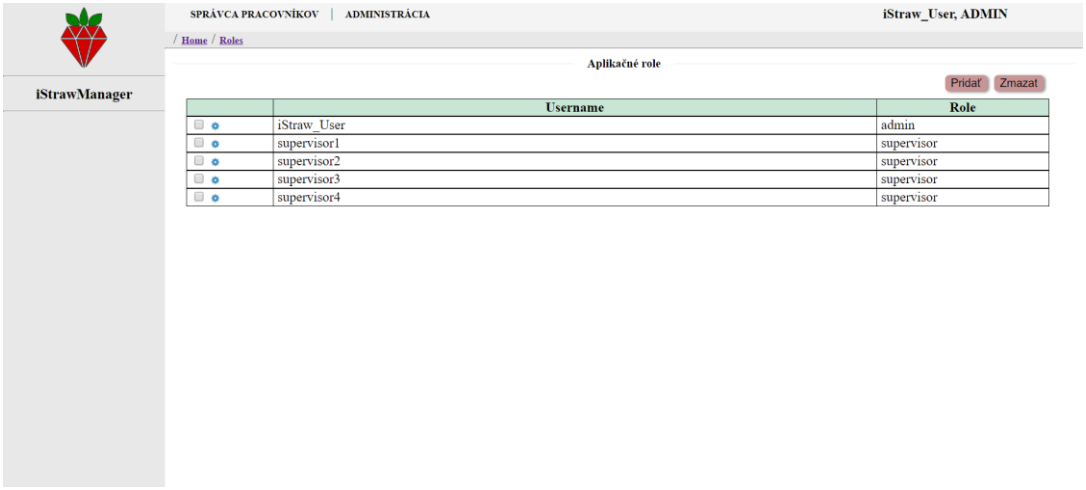
Ďalšou funkciou stránky je odstránenie existujúcich polí. Kliknutím na tlačidlo ‘Zmazať’ sa zobrazí potvrdzovací dialóg pre zmazanie vybraného poľa. Odstránením jednotlivých polí sa zmažú všetky záznamy, ktoré sú s daným poľom asociované.



Obrázok C.214: Potvrdzovací dialóg zmazania polí

C.5.2 Nastavenie rolí

Stránka ‘Nastavenie rolí’ definuje aplikačné role, ktoré reprezentujú pracovníkov s priradenými prístupovými právami typu *admin* alebo *supervisor*.



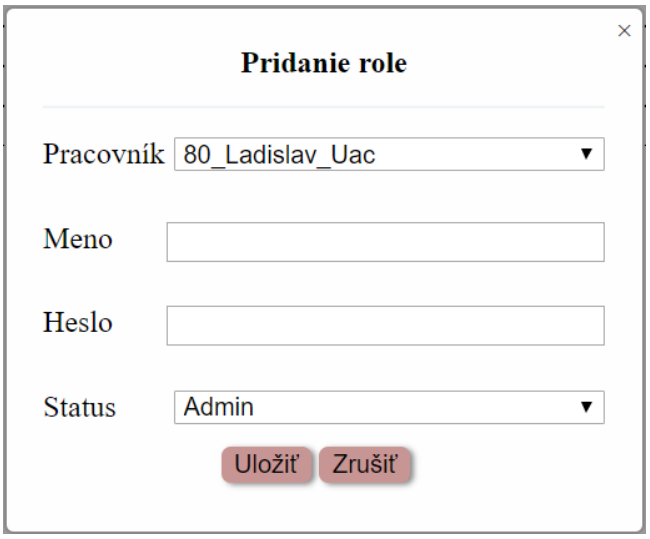
Obrázok C.22: Výzor stránky 'Nastavenie rolí'

Práva užívateľských rolí:

- ***supervisor*** disponuje oprávnením na vykonávanie funkcií pridávania, editácie a odstraňovania odovzdávkových dát jednotlivých pracovníkov,
- ***admin*** disponuje neobmedzenými právami pre prácu s dátami, ktorá umožňuje využívať všetky funkcie ponúkané webovou aplikáciou.

Vytvorenie aplikačnej role

Stránka umožňuje definovať novú aplikačnú rolu. Po kliknutí na tlačidlo ‘Pridať’ je zobrazený dialóg definujúci povinné atribúty vytváranej aplikačnej role.



Obrázok C.23: Dialóg definujúci novú aplikačnú rolu

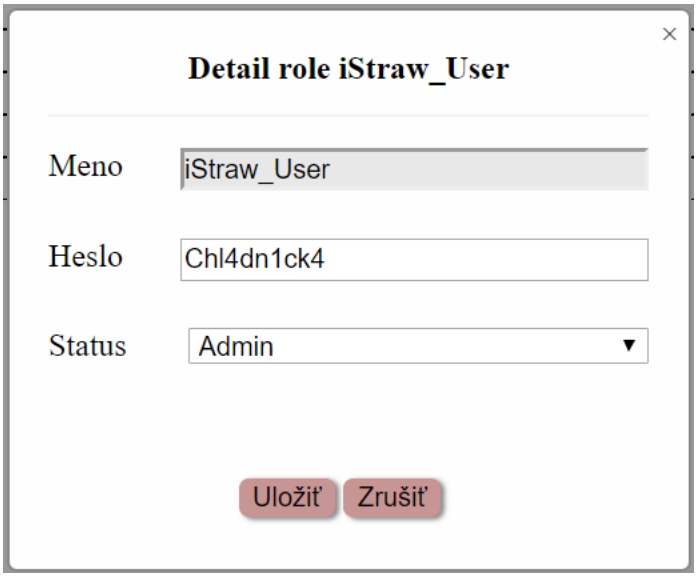
Povinné atribúty dialógu

- **Pracovník** - meno reprezentujúce pracovníka vo formáte *UID_Meno_Priezvisko*, s ktorým môže byť vytváraná aplikačná rola asociovaná (jednotliví pracovníci vznikajú procesom vytvorenia nového pracovníka s definovanou funkciou *rola*)
- **Meno** - prihlasovacie meno vytváranej aplikačnej role (jedinečná hodnota)
- **Heslo** - prihlasovacie heslo vytváranej aplikačnej role
- **Status** - užívateľská rola, ktorá bude vytváranej aplikačnej roli priradená

V prípade, že vytváraná aplikačná rola je už v aplikácií definovaná, je užívateľovi zobrazená informačná správa o jej existencii a nie je umožnené jej vytvorenie.

Editácia užívateľskej role

Aplikácia umožňuje editovať jednotlivé aplikačné role. Konkrétne, povoľuje zmenu hesla či zmenu priradenej užívateľskej role. Prihlasovacie meno aplikačnej role nie je v tomto kroku umožnené editovať. Detail role je zobrazený po kliknutí na ozubené koliesko definované na riadku, ktorý reprezentuje vybranú aplikačnú rolu.



Obrázok C.24: Dialóg definujúci detail aplikačnej role

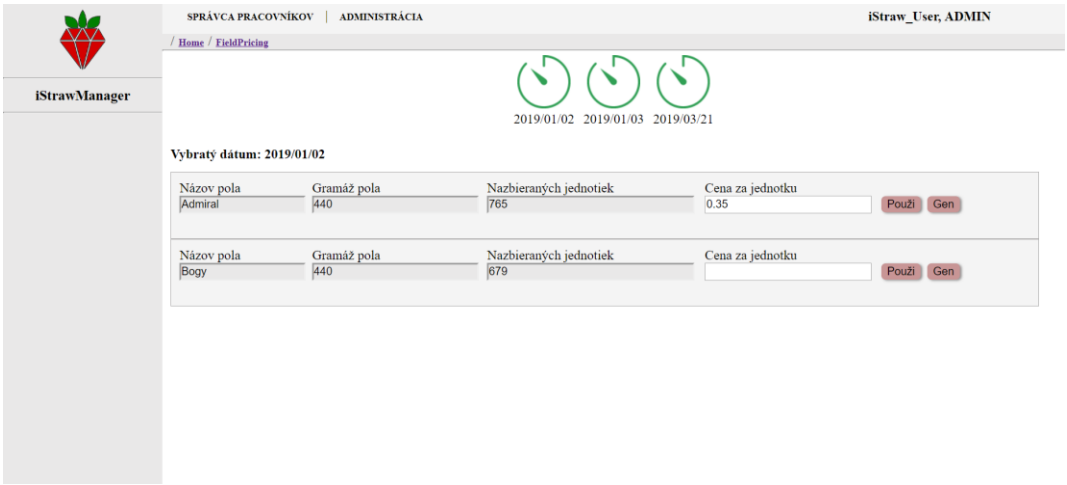
Zmazanie užívateľskej role

Jednotlivé záznamy zodpovedajúce aplikačným rolám je povolené odstrániť. Po odstránení musí existovať aspoň jedna aplikačná rola s prístupovými právami *admina*. V prípade, že po odstránení by neexistovala potrebná rola, nie je proces odstránenia vykonaný a užívateľ je oboznámený adekvátnou informačnou správou.

Pri odstránení aplikačnej role je odstránený aj pracovník, s ktorým je aplikačná rola asociovaná.

C.5.3 Nacenenie polí

Webová aplikácia definuje stránku ‘Nacenenie polí’, ktorá umožňuje pridávať cenu za jednotku na konkrétnom poli v určitý deň.

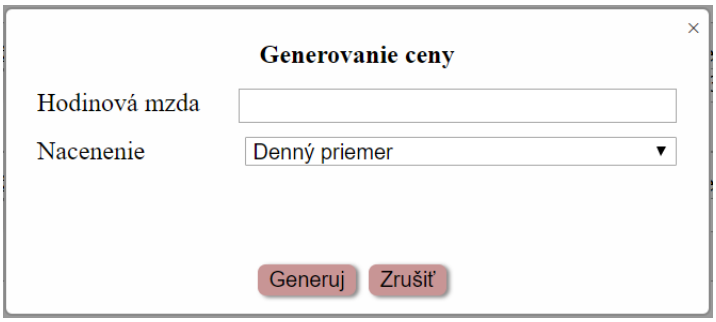


Obrázok C.25: Výzor stránky ‘Nacenenie polí’

Stránka obsahuje časovú os, ktorá zobrazuje zoznam všetkých dní, na ktorých sú definované odovzdávkové dáta. Kliknutím na určitý dátum sa aktualizuje výzor stránky spôsobom, že pod vybraným dátumom sú zobrazené riadky reprezentujúce jednotlivé polia, na ktorých sa vo vybraný termín vykonával zber. Definovať cenu za jednotku je umožnené dvoma spôsobmi, priamym zadaním hodnoty a jej následným uložením pomocou tlačidla 'Použiť' alebo využitím funkcie generovania ceny, ktorá je zahájená kliknutím na tlačidlo 'Gen'.

Proces generovania ceny

Po kliknutí na tlačidlo 'Gen' je zobrazený dialóg, ktorý slúži k nastaveniu základných atribútov procesu generovania ceny za jednotku.

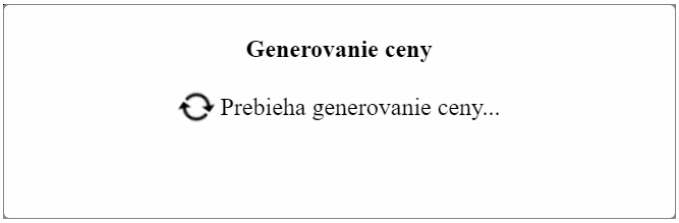


Obrázok C.26: Dialóg definujúci základné atribúty procesu generovania ceny

Povinné atribúty dialógu:

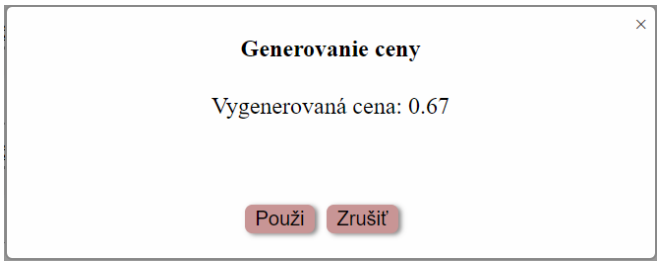
- **Hodinová mzda -** výška priemernej hodinovej mzdy, ktorá je pracovníkom poskytnutá
- **Nacenenie** - koeficient, na základe ktorého sa vykonáva výpočet. Povolené hodnoty atribútu sú:
    a. **Denný priemer** - cena za jednotku je počítaná ako $\frac{Hodinová\ mzda}{Priemerný\ počet\ nazbieraných\ jednotiek\ za\ hodinu}$
    b. **Hodinové minimum -** cena za jednotku je počítaná ako $\frac{Hodinová\ mzda}{Najmenší\ počet\ nazbieraných\ jednotiek\ za\ hodinu}$
    c. **Hodinové maximum -** cena za jednotku je počítaná ako $\frac{Hodinová\ mzda}{Najväčší\ počet\ nazbieraných\ jednotiek\ za\ hodinu}$

Vyplnením potrebných údajov a následným kliknutím na tlačidlo 'Generuj' je spustený proces generovania ceny a počas daného procesu je zobrazený nasledujúci dialóg (viz Obrázok C.27).



Obrázok C.27: Dialóg procesu výpočtu ceny

Po úspešnom vygenerovaní ceny za jednotku je zobrazený dialóg, ktorý obsahuje informáciu o vygenerovanej cene, ktorú je pomocou tlačidla ‘Použi’ umožnené aplikovať.

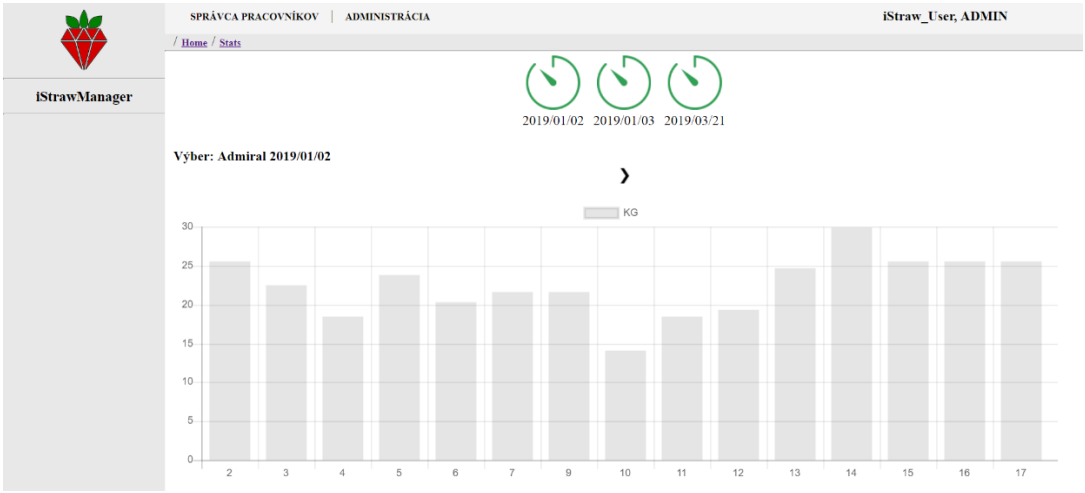


Obrázok C.28: Dialóg s vygenerovanou cenou

Generovanie ceny za jednotku je umožnené iba v prípade, ak všetci pracovníci, ktorí na konkrétnom poli v daný termín majú definované odovzdávkové záznamy, majú vytvorený štartovací záznam na danom poli. V opačnom prípade je užívateľ informačnou správou oboznámený o neexistencii štartovacích záznamov pre jednotlivých pracovníkov a proces generovania ceny za jednotku je zrušený.

C.5.4 Štatistika

Stránka ‘Štatistika’ slúži k zobrazeniu výkonnosti jednotlivých pracovníkov na vybraných poliach a vo vybraných termínoch.



Obrázok C.29: Výzor stránky 'Štatistika'

Kliknutím na špecifický dátum sa aktualizuje výzor stránky tak, že pod výberom sa zobrazí grafická reprezentácia výkonnosti pracovníkov, ktorí vykonávali zber vo vybranom termíne a na danom poli. Os x nesie informácie o identifikačných číslach pracovníkov a os Y reprezentuje počet odovzdaných jednotiek. V prípade, že vo vybranom termíne je definovaných viacero polí, na ktorých sa konal zber je nad grafom zobrazená riadiaca ikona, ktorá umožňuje preklikávať medzi jednotlivými poľami.