Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Mobilní přístup k podnikovému informačnímu systému K2

Plzeň 2017 Bc. Tomáš Balíček

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 18. května 2017

Bc. Tomáš Balíček

Poděkování

Děkuji Ing. Ladislavu Pešičkovi za odborné vedení mé diplomové práce a za cenné rady, které mi pomohly tuto práci vytvořit.

V Plzni, dne 18. května 2017

Bc. Tomáš Balíček

Abstract

This thesis solves mobile access to the K2 business information system. The aim of this thesis is to design and implement a mobile access to the K2 information system for company sales representative. They will have the necessary functions and data for real-time pricing of produced products. I focused on the multiplatform development of mobile applicatoins so that the resulting application could be used for iOS, Android and even for web browsers. The issue of mobile access is addressed with an emphasis on the security of the whole solution.

Abstrakt

Tato práce řeší mobilní přístup k podnikovému informačnímu systému K2. Cílem této práce je navrhnout a realizovat mobilní přístup k informačnímu systému K2 pro obchodní zástupce společnosti, ve kterém budou mít k dispozici potřebné funkce a data pro nacenění vyráběných produktů v reálném čase. Zaměřil jsem se na multiplatformní vývoj mobilních aplikací, aby výsledná aplikace mohla být používána jak uživateli iOS a Android, tak i pomocí webového prohlížeče. Problematika mobilního přístupu je řešena s důrazem na bezpečnost celého řešení.

Obsah

1 Úvod

Diplomová práce se zabývá problematikou mobilního přístupu k podnikovému informačnímu systému K2 pro obchodní zástupce společnosti Pebal s.r.o.. V práci nejprve popíšeme informační systémem K2, prozkoumáme možnosti přístupu k tomuto systému a analyzujeme činnosti, které budou obchodní zástupci k nacenění produktů v reálném čase potřebovat.

Dalším krokem bude navrhnutí aplikace, která bude mít přístup k datům z informačního systému K2 a bude dostupná jak na desktopu pomocí prohlížeče, tak na tabletech nebo telefonech pomocí mobilní aplikace. Mobilní aplikace musí být dostupná jak na platformě Android, tak na platformě iOS. Navrhneme vhodný databázový model, kde budou data uložena a zvoleny vhodné technologie, aby bylo vyhověno výše uvedeným požadavkům.

Poté navrženou aplikace realizujeme s ohledem na zabezpečení a offline režim v rámci možností.

Dále bude ověřena funkcionalita navržené a realizované aplikace pomocí unit testů a emulátorů. V posledním kroku se zaměříme na samotnou distribuci aplikace, kde budou shrnuty možnosti distribuce mobilních aplikací pro firmy a organizace v mobilních obchodech Google Play pro Android a Apple Store pro iOS.

2 Informační systém K2

V této kapitole bude popsán informační systém jako takový, budou popsány jednotlivé typy informačních systémů a poté se bude tato kapitola věnovat informačním systémem K2. V této kapitole bylo mimo jiné čerpáno z průvodce výběrem informačního systému od společnosti atmitec, který není veřejně dostupný. Průvodce byl zaslán po zaslání žádosti.

2.1 Informační systém

Informační systém je nástroj, který zná klíčové procesy napříč všemi odděleními firmy. V informačním systému jsou uložené veškeré důležité informace, data nebo dokumenty. Zaměstnanci ví, co mají dělat a vedení má přehled o tom, jak si firma vede nebo jak si stojí jednotlivá oddělení. Většina procesů je v informačním systému možné automatizovat a lidé ve firmě se pak mohou plně věnovat odborné práci místo administrativy, kterou za ně zvládne vyřešit podnikový software [1].

2.1.1 CRM, ERP, Ekonomické systémy, IS

V oblasti informačních systémů se vyskytují označení jako CRM, ERP, ekonomické systémy nebo IS, které budou dále vysvětleny:

CRM

CRM je zkratkou anglických slov *Customer Relationship Management*, neboli řízení vztahu se zákazníky. Jde tak o systémy pracující nejčastěji s databází zákazníků, která eviduje vzájemné vztahy a řídí marketingové aktivity či zákaznickou podporu.

Systémy ekonomické

Ekonomické systémy jsou poměrně úzce zaměřené a věnují se převážně účetnictví, mzdám a dalším aktivitám spojeným s ekonomikou podniku.

ERP

ERP systémy neboli *Enterprise Resource Planning* česky plánování podnikových zdrojů jsou již komplexnější systémy zaměřující se hned na několik firemních odvětví současně. Většinou jde o výrobu, sklady, logistiku, nákup a prodej či finance a podnikovou ekonomiku.

IS

IS je z terminologického hlediska trochu složitější. Obecně lze říci, že všechny výše uvedené skupiny firemního softwaru je možné označit jako informační systémy, neboť všechny systematicky pracují s informacemi. O informačním systému lze tedy mluvit jako o uceleném řešení, jenž je kombinací různých řešení, například CRM + ERP. Označení IS tedy využívají ty systémy, které jsou schopné pokrýt veškeré firemní aktivity najednou. To znamená, že umožňují pracovat jak s firemní ekonomikou, tak řešit vztahy se zákazníky, ale zároveň i výrobu, sklady a veškeré další firemní oblasti. To vše je provázáno v jednom SW řešení.

2.2 Co je informační systém K2

K2 je komplexní informační systém, který umožňuje pokrýt většinu aktivit ve firmě, jako je například výroba, řízení skladu, obchod, nákup, e-shop až po účetnictví. Dodavatelem tohoto informačního systému je firma K2 atmitec [2]. K2 je možné provozovat na vlastních serverech nebo využít cloudové služby přímo od společnosti K2 atmitec.

Přestože je většina informačních systémů uzavřená, K2 má vývojové prostředí otevřené. To je obrovská výhoda pro společnosti, které mají vlastní IT oddělení či zaměstnance, který zvládne programovací nástroje. A tak je

možné si K2 upravovat podle svých potřeb. Pro psaní vlastních skriptů v K2 je využit speciální skriptovací jazyk K2 Skript.

2.2.1 K2 Skript

Jednou z možností, jak provádět změny funkční a vizuální v rámci IS K2, je takzvaný K2 skript. Jedná se o jazyk syntakticky podobný Object Pascalu, nebo procedurálnímu SQL, přejímá od něj většinu datových typů, operátorů a rezervovaných slov. K2 skript je interpretován vlastním interpretem. K2 Skript slouží především k návrhu a tvorbě tiskových sestav [3].

2.2.2 Pracovní plocha K2

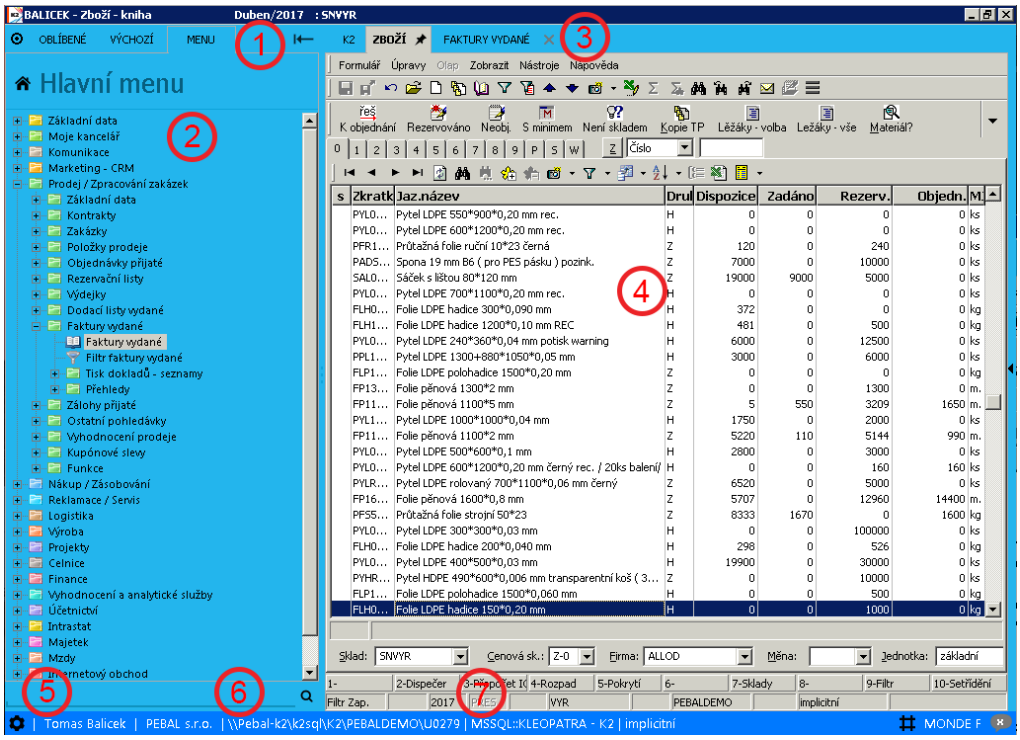
Na obrázku 2.1 můžeme vidět úvodní obrazovku informačního systému K2. Úvodní plochu si můžeme přizpůsobit plně svým potřebám a veškerá nastavení jsou ukládána. V této kapitole bylo čerpáno z dokumentace, která je dostupná na nainstalovaném systému K2.

1) Nabídka pro přechod mezi jednotlivými menu. Do záložky OBLÍBENÉ si můžeme uložit vlastní nejčastěji používané funkce, které jsou nám poté přístupné pro rychlejší přístup. V záložce VÝCHOZÍ jsou předvybrány nějaké základní nejpoužívanější funkce z jednotlivých modulů. Pod záložkou MENU se skrývá kompletní nabídka jednotlivých modulů.

2) Uzly jednotlivých modulů IS K2. Pomocí této stromové struktury můžeme přistupovat k jednotlivým částem IS. Zobrazeny jsou všechny moduly, ale záleží na přístupových právech každého uživatele, jestli má možnost jednotlivé uzly otevřít. Jedná se o následující uzly:

Základní data - obsahuje funkce pro nastavení základních číselníků, knih, parametrů, bankovních účtů a účetního rozvrhu.

Moje kancelář - evidence úkolů, notifikační systém, nastavení kalendáře. *Notifikace* - slouží k upozornění na určitou událost (např. potvrzení dokladu, zaplacení dokladu nebo příchozí hovor). *Workflow* - slouží k procesnímu řízení firmy, vedení ISO dokumentace apod. *Kalendář* - modul kalendář umožňuje



Obrázek 2.1: Pracovní plocha informačního systému K2.

správu soukromých a pracovních kalendářů spolu s kalendářem dokladů IS K2 v jednom místě.

Komunikace - tento modul je nadstavbou modulu Marketing, slouží ke zjednodušení komunikace s klienty a pro práci s interní elektronickou poštou.

Marketing - CRM - evidence obchodních informací, vedení údajů o partnerech, kontaktních osobách, aktivitách, nabídkách, kampaních a úkolech. *Nabídky/Poptávky* - modul umožňuje zpracování nabídek a poptávek.

Prodej/Zpracování zakázek - řízení výstupu zboží a služeb z firmy (účetní jednotky) k zákazníkům. Tento modul lze také nazvat Odbyt.

Nákup/Zásobování - řízení vstupu zboží a služeb do firmy (účetní jednotky) od dodavatelů.

Reklamace/Servis - slouží pro evidenci veškeré činnosti související s opravami a reklamacemi.

Logistika - řízení skladů, inventarizace skladů, řízení zásilkových služeb a podpora dopravy. *Sklad* - řízení skladových pohybů. *Inventury* - provádění inventur skladů. *Zásilkové služby* - podpora dodávek zboží zákazníkům prostřednictvím zásilkových společností. *Doprava* - sledování vozidel, nákladů na jednotlivá vozidla, vystavení příkazů k jízdě, řízení rozvozů.

Výroba - technologická příprava výroby, plánování operací a zdrojů, vyhodnocení výroby.

Projekty - řízení projektů zahrnující frontu práce, výkazy práce včetně definování projektových rolí a zdrojů.

Celnice - dovoz, vývoz, řízení celních skladů včetně realizace.

Finance - agenda modulů Banka, Pokladna, Platební kalendář, VP Banka a Kasa (ovládací program počítačové pokladny pro maloprodej).

OLAP - nástroj pro vyhodnocování dat a tvorbu analytických pohledů pomocí tzv. datových kostek (vícerozměrných matic).

Účetnictví - vedení účetnictví firmy včetně rozborů hospodaření, sestavení rozvah a výsledovek, peněžní toky atd.

Majetek - evidence všech druhů majetku; odpisování atd.

Mzdy - mzdy a personalistika.

Internetový obchod - modul slouží k pružné obchodní komunikaci prostřednictvím internetového rozhraní.

Správce - počáteční nastavení parametrů, uživatelských práv, databází atd.

3) Záložky pro přistup k otevřeným modulům IS. Jednotlivé moduly, které má uživatel aktuálně otevřené se zobrazují v tomto menu. Díky tomuto menu si mezi nimi uživatel může rychle pohybovat a může mít otevřeno více modulů současně.

4) Okno pro správu jednotlivých modulů. Zde je možno data editovat, přidávat, mazat, tisknout a podobně.

5) Zobrazení jména aktuálně přihlášeného uživatele.

6) Cesta k domovskému adresáři přihlášeného uživatele.

7) Zdroj dat IS. Jedná se o cestu k databázi, ke které je IS připojen. Tato cesta jde měnit v konfiguračním souboru IS.

2.2.3 Licencování IS K2

Informačním systému K2 je licencován pro jednotlivé uživatele, neplatí se tedy za systém jako takový, to znamená, že je zakoupen balíček určitého počtu uživatelů, například 10 a to je maximální počet uživatelů, který se mohou do systému současně přihlásit. V systému tedy může být zaregistrováno 100 různých uživatelský účtů, ale současně přihlásit se může pouze 10 uživatelů. Dále je možné nějakým uživatelům, například managementu, přiřadit výhradní právo a dané místo zarezervovat. Takže pokud bude mít firma zakoupen balíček o 10 uživatelích, tak například 2 může vyhradit vyššímu managementu a tito dva uživatelé se přihlásí vždy a zbylý počet bude omezen na 8 přihlášení. Takže i pokud 2 uživatelé z managementu přihlášeni v systému nebudou, devátý běžný uživatel se do systému nepřihlásí. Toto řešení je dle mého názoru šikovné, neboť pokud by se platilo za počet uživatelů v systému, kteří jsou jen zaregistrováni, pravděpodobně by to vedlo k zpřístupnění jednoho účtu více uživatelům, aby ušetřili a nad systémem by nebyla dostatečná kontrola. V tomto případě, platí firma jen za to, co skutečně využívá a to si myslím, že je správné řešení.

Každý uživatel se může do systému přihlásit vícekrát. Toto vícenásobné přihlášení uživatel využije především v případech, kdy například spouští operace náročné na čas. Může se tedy přihlásit znovu a v tom dalším přihlášení pokračovat v práci.

2.3 K2 ve firmě Pebal s.r.o.

Společnost Pebal využívá vlastních serverů, na kterých je provozován operační systém Windows Server 2008 a MySQL databáze a informační systém K2 ve verzi 7.

Níže bude popsán výrobní proces ve firmě Pebal, který bude důležitý pro návrh aplikace, protože budeme pracovat s jednotlivými atributy výrobních strojů.

Firma Pebal s.r.o. se zabývá výrobou obalového materiálu, která se skládá ze 3 hlavních kroků:

2.3.1 Extruze

Extruze je výrobní proces, při kterém vzniká z granulátové směsi výsledná fólie, se kterou je poté možné dále pracovat. Fólie mohou být mono nebo i vícevrstvé. Granulát je základní surovina, ze které je možno vyrábět další plastové výrobky. Druhů granulátu je nespočet a výsledné vlastnosti a kvalita produktu závisí na směsi použitých granulátů a zvolenému procesu výroby. K této fázi výroby se tedy váže i laboratoř, kde vznikají jednotlivé receptury fólií, které vznikají na extruzních linkách.

2.3.2 Tisk

Tisk je druhá fáze výroby, kdy se na vyrobenou fólii tisknou potřebné informace, jako je například název produktu, který bude ve výsledném sáčku uložen, čárový kód a další potřebné informace požadované zákazníkem. Z důvodu rychlosti se pro tisk využívá flexotisk. Nevýhodou flexotisku, oproti digitálnímu tisku, je nutnost výroby štočků před prvním tiskem, ale cena následného tisku je mnohonásobně nižší a rychlejší. Flexotisk se tedy nehodí pro tisk malého množství. Na obrázku 2.2 můžeme vidět jeden z tiskových strojů.

Obrázek 2.2: Fotografie z výroby společnosti Pebal s.r.o.

2.3.3 Konfekce

Konfekce je posledním krokem pro výrobu finálního výrobku. Konfekce je proces, při kterém dochází k svařování, překládání a řezání jednotlivých částí fólie a vzniká finální produkt, například pytel, taška nebo plachta. U konfekce se setkáme s následujícími parametry, se kterými bude potřeba pracovat při návrhu a realizaci výsledné aplikace:

Blokování

Blokování je seskupení výrobků do bloku. To znamená, že vyrobené sáčky budou například svařeny dohromady do bloku, ze kterého je možné jednotlivé sáčky odtrhávat.

Užitky

Užitky vysvětlím na příkladu. Budeme potřebovat fólii širokou 50 mm, ale na stroji můžeme vyrábět fólii širokou 200 mm, to znamená, že výslednou fólii nařežeme na 4 díly po 50 mm a užitky tedy budou 4.

Drát, gumička

Drát nebo gumičku můžeme využít při blokování. Vyrobené sáčky je možné dohromady svařit nebo můžeme použít místo toho drát, popřípadě gumičku.

Dutinka

Dutinka je papírová role, na kterou je namotána například fólie.

Sklad

Sklad je složení boků sáčku. Pokud chceme mít větší objem sáčku, můžeme využít takzvaný sklad. Boky sáčku jsou složeny uvnitř daného sáčku a po rozbalení máme v sáčku větší objem.

Klopa

Klopa je převis na sáčku. Na klopu je možné například nanést lepící pásku, pomocí které je následně možné daný sáček uzavřít. Nebo může být klopa využita k blokování, kdy jsou klopy svařeny k sobě a díky tomu vznikne blok sáčků, z kterého je možné jednotlivé sáčky odtrhávat.

V oddělení konfekce je celkem 5 různých výrobních strojů, které se liší a každý z nich má jiné parametry a umožňuje vyrábět různé produkty.

3 Možnosti přístupu k informačnímu systému K2

Přistupovat k informačnímu systému K2 je možné pomocí desktopové aplikace nebo webového rozhraní. Přičemž funkce webového rozhraní jsou omezeny a na web není přenesena plná funkcionalita. Firma Pebal využívá v současné době pouze desktopovou aplikaci. Dále K2 nabízí i mobilní aplikace jako je K2 portál, K2 move, K2 sense, K2 point. Tyto aplikace obsahují moduly analytických služeb, díky kterým lze získat aktuální informace o stavu firmy, jako je například aktivita obchodníků, přehled zákazníků a další informace o firmě. Tyto aplikace jsou dostupné pro mobilní platformy iOS a Android.

Problém, který bude diplomová práce řešit, je přístup obchodních zástupců do informačního systému, aby byli schopni si v terénu zkalkulovat požadované produkty výroby. Výroba se skládá ze tří hlavních částí, jak bylo uvedeno výše, extruze, tisk, konfekce. Finální produkt se nemusí a zpravidla neskládá ze všech třech částí, to znamená, že zákazník může požadovat dodání například pouze fólie nebo může například dodat fólii, kterou si vyrobil sám a na společnosti Pebal bude jen z dodané fólie vyrobit pytlíky nebo na fólii natisknout nějakou grafiku.

3.1 Současné řešení

Ve firmě Pebal pracují jak externí, tak interní obchodní zástupci. Možné řešení, jak problém s kalkulací jednotlivých produktů vyřešit, je implementace těchto procesů přímo do systému K2. Ale vzhledem k tomu, že ve firmě pracují i externí obchodní zástupci, je nežádoucí, aby každý z nich měl nainstalován klientskou aplikaci informačního systému na svém PC. Pro výpočet výsledné ceny jsou zapotřebí informace, které jsou uloženy v informačním systému K2 jako jsou například aktuální ceny materiálu. Proto v současné době firma k nacenění produktů používá tabulkový editor Microsoft Excel. Technik společnosti Pebal každý týden nahraje aktuální data do Excelu a dá tento soubor na sdílený disk firemního PC, kde si ho poté zaměstnanci

mohou stáhnout. Toto řešení je nevyhovující hned z několika důvodů. Zaprvé, pokud je potřeba něco změnit v průběhu týdne, je nutné kontaktovat všechny zaměstnance, aby si stáhli aktualizovaný soubor. Dále jsou všechny důležité informace pro výpočet ceny produktů uloženy v tomto Excelu. Zaměstnanci mají v Excelu zpřístupněnou pouze část pro zadávání hodnot a část s výsledky. Vzorce a další informace jsou skryty, ale jsou fyzicky stále v Excelu uloženy a zde tedy vzniká potenciální prostor pro zneužití a toto řešení není tedy zcela bezpečné a praktické, protože pokud by se Excel dostal do nesprávných rukou, mohlo by dojít k získání těchto tajných informací a toho by mohla využít případná konkurence, což je nežádoucí.

3.2 Požadované řešení

Současné řešení je tedy z těchto důvodů nevyhovující a je potřeba navrhnout takové řešení, aby obchodní zástupci měli k dispozici tento kalkulátor s reálnými daty, které budou aktuální v reálném čase a všechny důvěryhodné informace zůstanou na serverech firmy. Aplikace by měla být přístupná jak na desktopu pomocí prohlížeče s nutností zadání přístupových údajů, tak na mobilních zařízení pomocí mobilní aplikace. Aplikace by měla fungovat jak na zařízeních s operačním systémem iOS, tak Android. Řešení musí klást důraz na zabezpečení aplikace a offline režim v rámci možností. Pro firmu je v současné fázi klíčový kalkulátor konfekce a dle domluvy s firmou a vedoucím práce, se proto v diplomové práci bude primárně řešit tato část.

4 Návrh aplikace

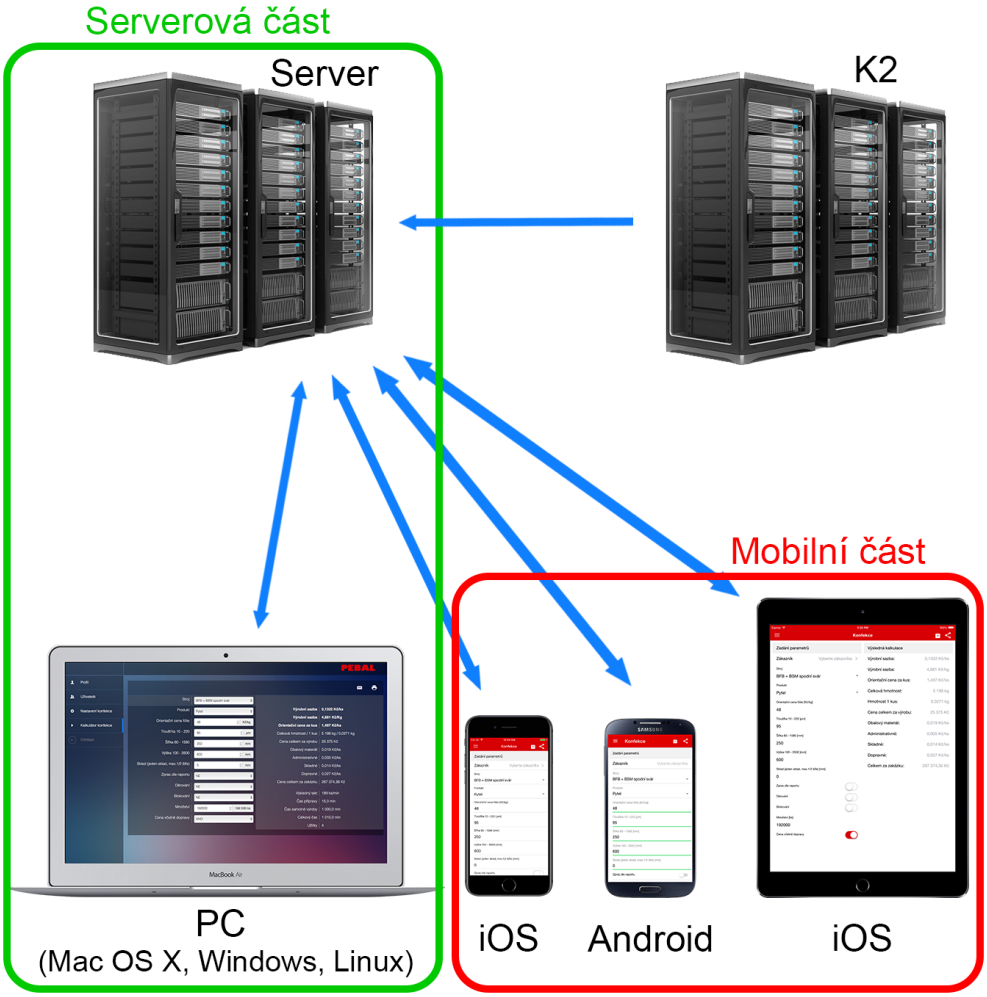
Cílem je navrhnout aplikaci, která bude umožňovat přihlášení obchodních zástupcům a také administrátorům, kteří budou moci spravovat, přidávat, blokovat jednotlivé obchodní zástupce. Aplikace bude mít přístup k aktuálním datům informačního systému K2, aby výpočty, které budou v aplikaci prováděny, byly vždy aktuální. Všechny výpočty musí probíhat na jednom centrálním místě, aby se změna výpočtového vzorce projevila ihned na všech zařízeních. Vytvořený systém musí být k dispozici jako nativní aplikace na mobilních platformách iOS a Android a zároveň musí být dostupná na desktopu prostřednictvím prohlížeče, aby ji bylo možné využívat pro operační systémy Windows, Mac OS X i Linux.

4.1 Celková struktura navrženého systému

Zobrazení navrženého systému můžeme vidět na obrázku 4.1. První částí systému bude serverová část, kde budou probíhat výpočty zadaných kalkulací a budou zde uložena data v relační databázi MySQL. Zároveň zde bude probíhat komunikace a načítání dat z informačního systému K2. Komunikace mezi vytvořenou serverovou aplikací a IS K2 je pouze jednostranná. Dochází zde pouze k načítání dat z IS K2, ale nikoliv k ukládání zpět do IS K2. Je to z toho důvodu, že získané zakázky již do K2 vkládá zakázkové oddělení a nikoliv obchodní zástupce.

Druhou částí systému bude mobilní část, tedy nativní aplikace pro mobilní zařízení, která bude komunikovat se serverem a uživatelé budou moci provádět kalkulace pohodlně na svých mobilních telefonech nebo tabletech.

Důležitým aspektem celého systému bude dynamický formulář, do kterého se budou zadávat data pro kalkulaci. Formulář můžeme vidět na koncových zařízeních na obrázku 4.1. Formulář se musí přizpůsobit typu zařízení, aby byl vždy maximálně přehledný a práce s ním byla příjemná. Formulář bude zároveň dynamický z hlediska jednotlivých polí, protože každý stroj potřebuje na vstupu jiné parametry.



Obrázek 4.1: Zobrazení navrženého systému.

4.2 Výběr vhodných technologií

První vhodnou technologii bude potřebné zvolit pro serverovou část aplikace. Tu část kde bude API, prostřednictvím kterého bude komunikovat mobilní aplikace a zároveň bude poskytovat uživatelské rozhraní pro přístup z běžného prohlížeče. V tomto rozhraní budou stejné funkce jako v mobilní aplikaci a zároveň administrátoři zde budou moci spravovat, přidávat, odebírat a blokovat jednotlivé uživatele aplikace a provádět další různá nastavení parametrů aplikace.

Druhou vhodnou technologii bude nutné zvolit pro mobilní aplikaci. Při volbě musíme brát v úvahu, že je potřeba, aby mobilní aplikaci bylo možné nainstalovat jak na mobilní platformu iOS od firmy Apple, tak na Android od firmy Google. Zde je tedy možné jít nativní cestou a pro iOS vytvořit aplikaci v programovacím jazyku Swift nebo Objective-C a pro Android vytvořit aplikaci v Javě. Nativní programování má zajisté výhodu v tom, že aplikace budou svižnější a budou vypadat nativně, ale velká nevýhoda je v tom, že je potřeba udržovat stejnou funkcionalitu ve dvou různých programovacích jazycích. A pokud bychom přemýšleli o vydání aplikace i pro Windows Phone, tak dokonce ve třech jazycích. Tato velká nevýhoda samozřejmě rapidně navyšuje náklady na vývoj a následnou údržbu aplikací, protože je nutné udržovat kód aplikace ve dvou nebo třech programovacích jazycích zároveň. Toto si uvědomují i velcí hráči na trhu, kteří mají ve svém portfoliu aplikace, které patří k nejstahovanějším na celém světě a to Facebook, Instagram, Messenger [4]. Všechny tyto aplikace vlastní společnost Facebook Inc, potažmo aplikaci Instagram vlastní společnost Instagram Inc, kterou Facebook v roce 2012 koupil [5]. Tyto aplikace jsou vyvíjeny v multiplatformním nástroji React Native, kterou vyvíjí sám Facebook [6] [7]. Je několik dalších multiplatformních nástrojů, které je možné pro multiplatformní vývoj mobilních aplikací využít:

Xamarin

Xamarin je multiplatformní nástroj, který vyvíjí společnost Microsoft [8]. Při vývoji multiplatformních mobilních aplikací se používá programovací jazyk C#. Xamarin je nástroj, který následně dokáže aplikaci napsanou v C# sestavit do nativního kódu Android, iOS, Windows a také dalších platforem jako je například Mac OS X nebo Apple Watch a mnoho dalších. Vzhledem k tomu, že Xamarin patří Microsoftu, tak je k dispozici i zdarma Visual Studio pro Windows nebo Xamarin Studio pro Mac, což je jedna z dalších předností Xamarinu a tomuto multiplatformnímu nástroji předurčuje zajímavou budoucnost.

React Native

Tento multiplatformní nástroj vyvíjený společností Facebook a pro vývoj aplikací v React Native je využíváno programovacího jazyku JavaScript. React Native stejně jako Xamarin dokáže aplikaci sestavit do nativní podoby,

ale již jen pro Android >= 4.1 nebo iOS >= 8.0, nikoliv pro další platformy [9].

Cordova - Ionic

Dalším z multiplatformních nástrojů je Apache Cordova [10], která využívá *WebView* k zobrazení aplikace. To znamená, že připravíme webovou aplikaci například pomocí frameworku Ionic [11]. Můžeme aplikaci samozřejmě vyvíjet i bez dalšího frameworku ale Ionic nám zajistí, že aplikace bude vypadat nativně, to znamená, že pomocí jednoho kódu bude aplikace vypadat jako nativní na iOS i na Androidu. Aplikaci tedy vyvíjíme pomocí programovacího jazyku JavaScript spolu s HTML5 a CSS. Ionic je založen na JavaScriptové knihovně Angular [12], která usnadňuje práci s JavaScriptem a HTML. Díky tomu, že je framework Ionic založen na Angularu, je práce s Ionicem velice rychlá a intuitivní. Aplikaci je možné sestavit do platforem iOS, Android nebo Windows Phone.

4.2.1 Serverová část - framework Nette

Pro serverovou část aplikace byl zvolen PHP framework Nette [13]. Tento PHP framework byl zvolen z toho důvodu, že jsem s ním pracoval již v minulosti a byl jsem s ním maximálně spokojen. Nette disponuje celou řadou modulů, které napomáhají, zjednoduší a celkově zpříjemňují vývoj. Výhoda Nette je, že se jedná o český framework a tudíž komunita kolem tohoto frameworku v ČR je velice rozšířená. To, že je Nette oblíbené dokazuje fakt, že řada velkých českých komerčních webů ho pro svůj vývoj používá. Jedná se například o weby jako: slevomat.cz, csfd.cz, socialbakers.com, uloz.to a stovky dalších [14]. Nette je založené na softwarové architektuře MVC.

MVC

Model-View-Controller je softwarová architektura, která rozděluje datový model aplikace (Model), uživatelské rozhraní (View) a řídící logiku (Controller) do tří nezávislých částí. Tím se jednak aplikace zpřehledňuje a zároveň umožňuje testování jednotlivých částí samostatně.

Zabezpečení

Dalšími přednostmi Nette je jeho zabezpečení [15]. Nette dbá na základní zranitelnosti jako jsou například:

I. Cross-Site Scripting (XSS) je metoda narušení webových stránek zneužívající neošetřených výstupů. Útočník pak dokáže do stránky podstrčit svůj vlastní kód a tím může stránku pozměnit nebo dokonce získat citlivé údaje o návštěvnících. Proti XSS se lze bránit jen důsledným a korektním ošetřením všech řetězců. Přitom stačí, aby bylo jen jednou opomenuto a celý web může být rázem kompromitován.

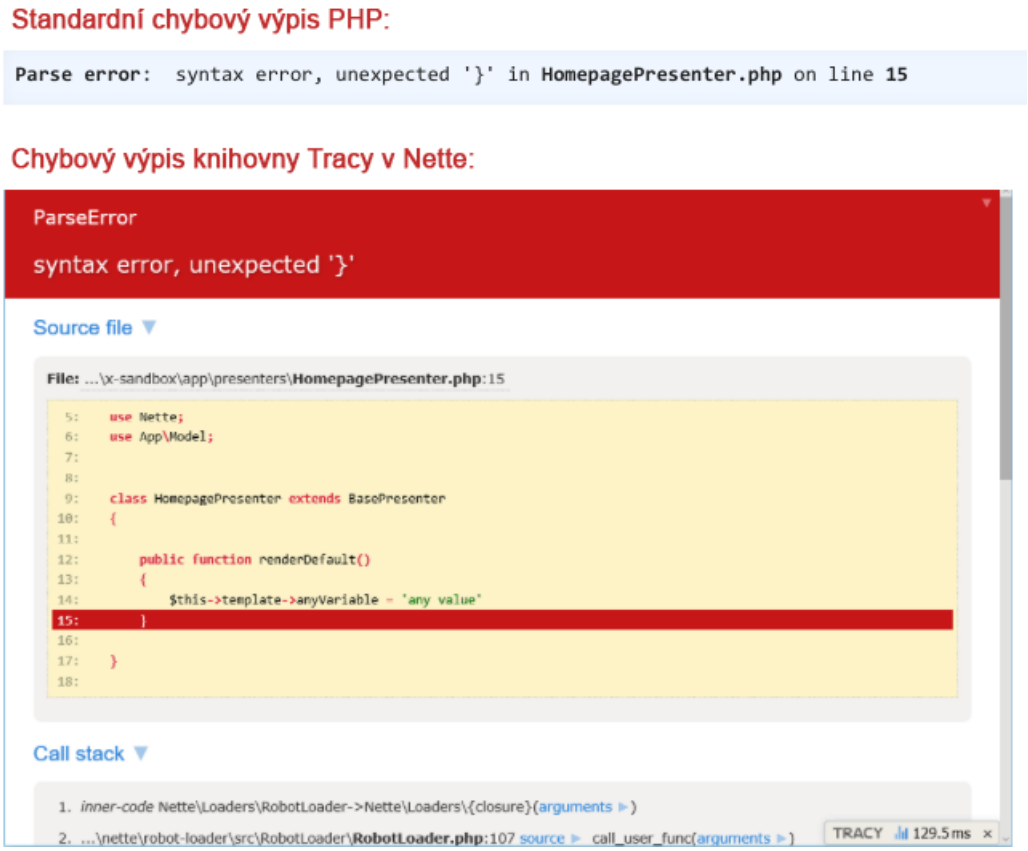
Příkladem útoku může být podstrčení upravené URL adresy uživateli, pomocí které injektujeme do stránky svůj kód. Když aplikace nebude výstupy řádně ošetřovat, vykoná skript v prohlížeči uživatele. Tímto způsobem můžeme uživateli například zcizit identitu. To znamená, že útočník odcizí session danému uživateli a může se následně vydávat za něj.

Nette framework přichází s revoluční technologií Context-Aware Escaping, která zbaví uživatele frameworku rizika Cross-Site Scriptingu. Všechny výstupy totiž ošetřuje automaticky a tak se nemůže stát, že by programátor na něco zapomněl.

II. Session hijacking, session stealing, session fixation. Se správou session je spojeno hned několik typů útoků. Útočník buď zcizí nebo podstrčí uživateli své session ID a díky tomu získá přístup do webové aplikace, aniž by znal heslo uživatele. Poté může v aplikaci provádět cokoliv, aniž by o tom uživatel věděl. Obrana spočívá ve správné konfiguraci serveru a PHP.

Přičemž Nette Framework nakonfiguruje PHP automaticky. Programátor tak nemusí přemýšlet, kterak session správně zabezpečit a může se plně soustředit na tvorbu aplikace. Vyžaduje to však povolenou funkci *ini_set()*.

Dalšími nástroji, kterými Nette disponuje je například knihovna *Tracy*. Tato knihovna dokáže vizualizovat chyby a výjimky do takové podoby, aby bylo snadné se v nich zorientovat a ve výpisu nalézt co nejvíce informací, které nám pomůžou k odhalení chyby. Na obrázku 4.2 je vidět, jak vypadá standardní chybový výpis PHP, oproti chybovému výpisu knihovny Tracy z frameworku Nette. Pokud nastane nějaká chyba v produkčním režimu, Nette



Obrázek 4.2: Rozdíl v zobrazení chybové hlášky v PHP a knihovny Tracy v Nette.

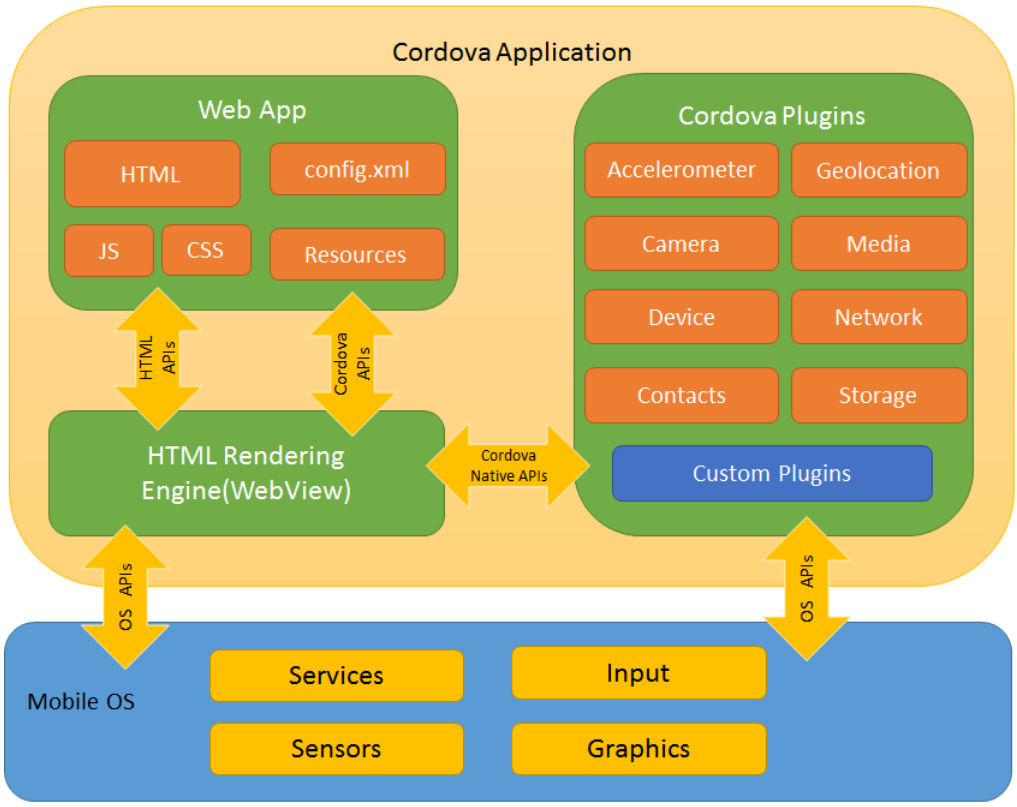
nezobrazí kompletní informace o chybě, protože by se mohlo jednat o cílenou chybu nějakého útočníka, který by ji mohl zneužít a zobrazí pouze informaci, že na serveru došlo k chybě a veškeré bližší informace uloží do HTML souboru do složky */log* na serveru. Následně upozorní administrátora e-mailem, aby mohl na danou chybu zareagovat a popřípadě ji opravit.

Nette dále disponuje vyzrálým objektovým návrhem, který využívá nových vlastností PHP a dle nezávislého testu [16] patří k těm nejrychlejším frameworkům vůbec.

Nette je šířeno pod open-source licencí BSD. Framework je tedy možné používat zdarma například i v komerčních projektech [17].

4.2.2 Mobilní část - Cordova + Ionic

Pro mobilní část aplikace byl zvolen framework Cordova [10] v kombinaci s frameworkem Ionic [11]. Cordova potažmo Ionic byl zvolen, protože Ionic je založen na frameworku Angular od společnosti Google, s kterým již delší dobu pracuji. Výhodou Cordovy je, že disponuje velkým množstvím nativních pluginů, které je možné bezplatně stáhnout. Další výhodou je, že Ionic má vlastní CLI a po jeho instalaci je tedy možné většinu správy aplikace provádět pomocí příkazů v příkazové řádce.



Obrázek 4.3: Zobrazení architektury frameworku Cordova.

Na obrázku 4.3 je vidět architektura frameworku Cordova. Ve *WebView* nám běží webová aplikace, která může být jen čisté HTML a JavaScript nebo můžeme použít již zmíněný framework Ionic, aby vývoj naší aplikace byl rychlejší a výsledná aplikace vypadala nativně na iOS, Androidu i Windows Phone s použitím stejného kódu. Pokud chceme ve frameworku Cordova použít nějaký HW nástroj nebo jinou nativní část mobilního zařízení jako je například kamera, je potřeba si do aplikace přidat vhodný plugin, který ko-

munikuje s *WebView* a předává potřebné informace mezi nativním kódem a *WebView*. Pluginů je k dispozici velké množství, ale pokud by jsme takový, který nám vyhovuje nenašli, je možné si ho vytvořit. Pro tvorbu pluginu je již nutné použít nativního programovacího jazyku dané platformy, tedy pro Android je to Java a pro iOS Objective-C a pro Windows Phone C#.

Ionic je šířen pod licencí MIT [18] a Cordova pod licencí Apache [19].

4.3 ER model

Vytvořený ER model aplikace můžeme vidět na obrázku 4.5. Databáze se skládá celkem z 18 tabulek. Vytvořený model je umístěn na serveru stejně jako framework Nette. Na obrázku 4.1 bude umístěn v serverové části označené zeleným ohraničením.

Users

V tabulce *users* jsou uloženy informace o registrovaných uživatelích systému jako jejich jméno, příjmení, e-mail, id role (cizí klíč z tabulky role), hash hesla a informace o blokaci uživatele. Jako username uživatele pro přihlášení je použit e-mail, který musí být unikátní u každého uživatele.

Users Login History

V tabulce *users_login_history* jsou uloženy informace o přihlášení jednotlivých uživatelů do systému. Jsou zde zaznamenávány jak úspěšné, tak neúspěšné pokusy. V tabulce se uchovávají informace s id uživatele (cizí klíč do tabulky users), který se do systému přihlašoval, datum a čas pokusu o přihlášení, IP adresa, ze které se uživatel přihlásil, aby bylo možné v případě opakovaného neúspěšného přihlášení dohledat jeho IP adresu a případně ji zablokovat na serveru, informace o tom jestli se jedná o přihlášení z webové aplikace nebo mobilní: 1 je mobilní, 0 je webová a informace o úspěšnosti přihlášení.

Users Subordinate

Tabulka *users_subordinate*, kde jsou uloženi podřízení jednotlivých uživatelů. Pokud má uživatel roli BDM (viz další odstavec), je možné k němu přiřadit podřízeného.

Roles

V tabulce *roles* jsou uloženy role uživatelů. Uživatelé mohou být role admin, user nebo BDM. Admin má v systému kontrolu nad všemi uživateli a má přístup do sekce nastavení parametrů kalkulace. User a BDM mají stejná přístupová práva, ale BDM jsou nadřízení jednotlivých userů a je tedy možné BDM přiřadit usery jakožto jejich podřízené. Tato funkcionalita bude využita v pozdější fázi projektu.

Update History

Tabulka *update_history* uchovává informace o změnách hodnot v systému. Změny v systému může provádět uživatel s rolí administrátora a pokud například změní jméno uživatele nebo hodnotu nastavení nějakého stroje, je to zaznamenáno zde. Aby bylo dohledatelné kdy, kdo a kde co změnil v případě výskytu nějaké chyby v systému. V tabulce je tedy uchována informace s id uživatele (cizí klíč do tabulky users), data, která byla odeslána, tabulka, která byla změněna a ID změněného záznamu.

Customers

V tabulce *customers* jsou uloženi zákazníci, kteří se aktualizují z informačního systému K2.

Konfekce Machine

Tabulka *konfekce_machine* uchovává informace o strojích určených ke konfekci. Jsou zde základní informace o strojích, které jsou dále potřebné k vý-

počtu zadané kalkulace. Jedná se o informace jako název stroje, hodinový tarif stroje a další různé koeficienty potřebné k výsledné kalkulaci.

Konfekce Product

V tabulce *konfekce_product* jsou uchovány informace o produktech, které lze na jednotlivých strojích vyrábět. V tabulce je tedy id stroje (cizí klíč do tabulky konfekce_machine) a u každého produktu jsou ještě další informace jako váhové koeficienty potřebné k výpočtu.

Konfekce Hollow

V tabulce *konfekce_hollow* jsou uloženy informace o dutinkách. Je zde id stroje (cizí klíč do tabulky konfekce_machine), ke kterému se dutinka váže a cena dutinky.

Konfekce Uzitky Kg

Tabulka *konfekce_uzitky_kg* slouží pro získání užitků na základě váhy. Každý stroj má jiné nastavení užitků. V této tabulce je tedy id stroje (cizí klíč do tabulky konfekce_machine) a hodnota užitků pro určité množství. Hodnoty v tabulce můžeme vidět na obrázku 4.4. Pokud tedy máme id stroje rovno 1 a váha je například 500 kg, výsledná hodnota užitků se musí vejít mezi hodnoty *from_value* a *to_value*. Počet užitků na základě váhy bude v tomto případě 4.

Konfekce Price Coefficient

V tabulce *konfekce_price_coefficient* jsou uloženy informace o koeficientu ceny. Každý stroj má různé hodnoty, proto je zde id stroje (cizí klíč do tabulky konfekce_machine) a podle zvolené jednotky minut nebo kilogramů je určen koeficient na základě zadané hodnoty, která se musí vejít mezi *from_value* a *to_value*.

id	from_value	to_value	value	konfekce_machines_id
1	0	100	1	1
2	100	300	2	1
3	300	1000000	4	1
6	0	200	2	2
7	200	10000000	2	2
9	0	10000000	1	3
10	0	10000000	1	4
13	0	300	1	5
14	300	1000000	2	5

Obrázek 4.4: Tabulka konfekce_uzitky_kg.

Konfekce Codes Limit

Tabulka *konfekce_codes_limit* slouží pro vytvoření kódu, který nám následně určí hodnotu taktu a užitku stroje. V tabulce opět najdeme id stroje (cizí klíč do tabulky konfekce_machine), ke kterému se záznam váže a podle parametrů výška, šířka a tloušťka získáme potřebné písmeno, z kterého je následně složen kód.

Konfekce Codes

V tabulce *konfekce_codes* jsou uloženy kódy, které získáme pomocí tabulky *konfekce_codes_limit* a pomocí tohoto kódu následně získáme hodnotu taktu stroje a užitků.

Konfekce Blocking

V tabulce *konfekce_blocking* je uložen koeficient blokování, který získáme podle množství kusů.

Konfekce Wire

Tabulka *konfekce_wire* je číselník, kde jsou uloženy informace o gumičkách. Pokud je možné u daného stroje navolit gumičky, v této tabulce jsou uloženy informace o jednotlivých typech gumiček. Typy gumiček jsou načítány z IS K2.

Konfekce Form

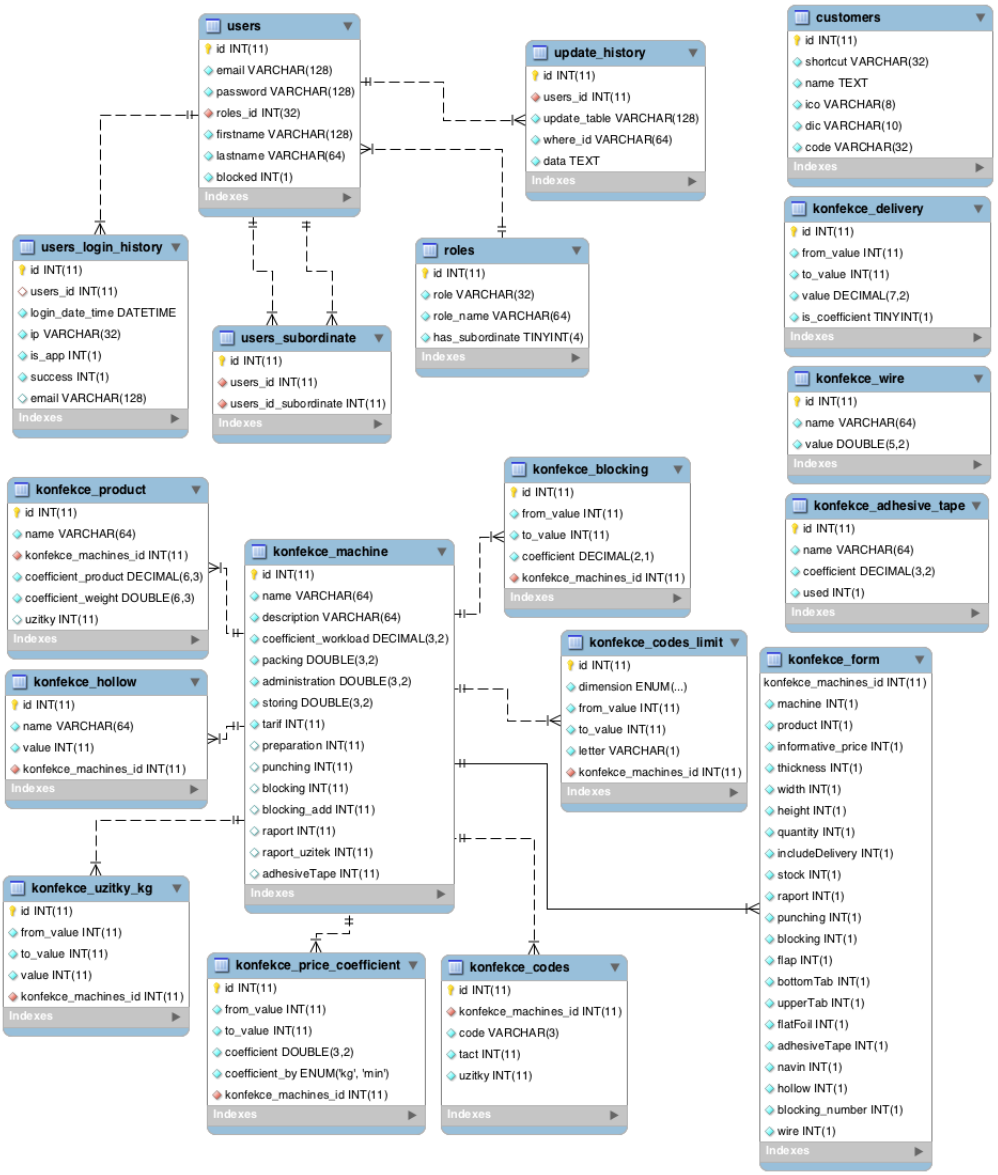
Tabulka *konfekce_form* slouží pro dynamické přizpůsobení formuláře pro výpočet kalkulace. Ve výrobě je několik strojů na konfekci a každý stroj má jiné parametry, které je nutné pro finální výpočet ceny nastavovat. To znamená, že po zvolení stroje, pro který chceme kalkulaci provést se nám zobrazí jen pole ve formuláři, které je nutné vyplnit. V tabulce nalezneme id stroje (cizí klíč do tabulky konfekce_machine) a jednotlivé sloupce tabulky reprezentují pole ve formuláři. Pokud má stroj nastaveno v daném sloupci hodnotu 1, znamená to, že se pole zobrazí a pokud 0, pole se zobrazovat nebude a pro daný stroj není potřeba hodnotu ve formuláři vyplňovat.

Konfekce Delivery

Tabulka *konfekce_delivery* je číselník, který nám udává hodnotu nebo koeficient dopravy podle množství.

Konfekce Adhesive Tape

Tabulka *konfekce_adhesive_tape* je také číselník a jsou zde uloženy informace o lepících páskách. Pokud stroj umožňuje výrobu s lepící páskou, zde jsou uloženy jednotlivé typy pásek. Typy pásek jsou načítány z IS K2.



Obrázek 4.5: Navržený ER model vytvářeného systému.

5 Realizace aplikace - serverová část

Pro realizaci serverové části aplikace bude využit framework Nette popsaný v kapitole 4.2.1. Při realizaci bude zapotřebí vyřešit problém s plovoucí řadovou částkou, aby výpočty byly přesné.

5.1 Příprava prostředí

Prvním krokem pro realizaci serverové části bylo stažení samotného frameworku Nette. To je možné provést dvěma způsoby. Manuálním stažením frameworku ze stránek Nette nebo využitím Composeru. Pokud se rozhodneme pro využití Composeru, využijeme následující příkaz.

composer create-project nette/web-project nazev-projektu

Celý framework se nám stáhne do složky *nazev-projektu* a můžeme s ním začít pracovat.

Pro použití výše uvedeného příkazu je nutné mít nainstalovaný nástroj Composer, který slouží pro správu závislostí v PHP. Tento nástroj nám poslouží i dále pro instalaci dalších knihoven do našeho projektu, nevyužijeme ho tedy jen jednou pro inicializaci projektu.

Na vývoj aplikace bylo využito IDE PHPStorm, který je standardně placený, ale pro studijní účely je možné využít licenci zdarma.

5.1.1 Spuštění aplikace

Pro spuštění aplikace v PHP je zapotřebí interpret PHP skriptů. Jako lokální server byla využita aplikace MAMP ve verzi zdarma. Lokální využití má rychlejší odezvu, ale bylo zapotřebí zpřístupnit server i z vnější sítě pro testování mobilní verze aplikace uživateli, kteří nebyly v lokální síti. Pro tyto účely bylo využito hostingových služeb společnosti Wedos a.s., kde byl zakoupen základní hosting v ceně 363 Kč na rok.

5.2 Plovoucí řádová čárka

Jeden z problémů, kterému bylo potřeba věnovat pozornost je plovoucí řadová čárka neboli floating number. Je to problém, který není spjat přímo s PHP ale ukládáním čísel v paměti jako takovým. Na tuto problematiku narazíme například při výpočtu, který je vidět na obrázku 5.1, konkrétně při výpočtu proměnné $c = $a - $b.

```php
$a = 36;
$b = 35.99;

$c = $a - $b;
//$c => 0.009999999999998

$d = bcsub($a, $b);
//$d => 0.01000000000000000000
```

Obrázek 5.1: Znázornění problému plovoucí řadové čárky.

V tomto případě bychom očekávali výsledek 0.01, ale pokud si proměnou $c vypíšeme zjistíme, že výsledek je ve skutečnosti 0.009999999999998. Tento problém je způsoben tím, že float a double jsou ukládány jako ostatní čísla v podobě nul a jedniček. A desetinná čísla není možné reprezentovat do binární podoby z důvodu omezeného místa v paměti. Proto 1/3 uložíme do paměti s omezenou délkou desetinných míst, například jako 0.33333. Podobný problém vzniká i u 1/100 neboli 0.01. Pokud je číslo převedeno do binárního formátu a poté převedeno zpět do desetinné podoby, vznikne zde daná nepřesnost.

Tento problém má za následek desítky životů v 90. letech 20. století. Jeden z nich se stal 25.10.1991, kdy mobilní raketový systém MIM-104 zabil 28 amerických vojáků, když minul cíl a dopadl do kasáren v Saúdské Arábii [20]. Na základě vyšetřování bylo zjištěno, že za příčinu mohla nepřesnost ve výpočtu zapříčiněna právě plovoucí řadovou částkou.

5.2.1 Řešení v PHP

V PHP jsem využil jako řešení knihovnu *BCMath*, která výsledek požadované matematické operace nevrací jako float, kde by vznikl daný problém, ale používá string. Pro výpočet můžeme využít funkce z této knihovny **bcadd** pro sčítání, **bcdiv** pro dělení, **bcmul** pro násobení, **bcsub** pro odčítání. Na

obrázku 5.1 můžeme vidět řešení tohoto problému u proměnné $d, kde je výsledek již správně 0.01.

Před tím než začneme využívat funkce *BCMath*, je potřeba si určit s jakou přesností má tato knihovna při matematických operacích počítat a k tomu slouží funkce *bcscale*. Tato funkce má jediný parametr a tím je počet míst za desetinou čárkou. Při výpočtu proměnné $d byl tento parametr nastaven na 20 a proto můžeme vidět, že na obrázku 5.1 má proměnná $d za desetinnou čárkou skutečně 20 míst.

5.3 GUI

Uživatelské rozhraní je řešeno s důrazem na jednoduchost a intuitivní ovládání. U webového rozhraní byla použita bootstrap HTML5 šablona *Glazzed-Admin-Template*, která byla zakoupena za částku 10$ v přepočtu za 250 Kč na portálu *http://www.pixeden.com*. Tato šablona obsahuje HTML a CSS kód, který je již částečně responzivní pro základní využití a obsahuje naformátované menu, tabulky, formuláře a další komponenty, které budeme pro naši práci potřebovat. Bylo by samozřejmě možné si grafiku a jednotlivé komponenty navrhnout od základu a HTML a CSS kód si vytvořit, ale vzhledem k ceně šablony a množství práce, která by bylo nutné vynaložit, to není ideální řešení. Navíc šablona se nějaký čas již prodává, a tak je otestována i dalšími uživateli v různých prohlížečích.

5.4 Struktura aplikace

Jak bylo zmíněno výše, Nette je MVC, takže aplikace je rozdělena na tři základní části. První část je model, který se stará o ukládání a načítání dat z databáze, druhá je view, což jsou v případě Nette šablony s koncovkou *.latte* a poslední je controller, který se stará o komunikaci mezi view a modelem, v Nette jsou to takzvané presentery. V Nette je možné rozdělit aplikaci do několika modulů a to je vhodné například pokud máme aplikaci, která má více částí jako je frontend, kde přistupují například zákazníci a backend, ke kterému je nutné mít přístupové údaje a obsluhují ho většinou správci systému. V našem případě budou k systému přistupovat pouze zaměstnanci firmy, kteří mají přiděleny přístupové údaje, a tak bude vytvořen pouze jeden

modul.

5.4.1 Popis konfigurace

Pro základní konfiguraci nám slouží konfigurační soubor *app/config/config.neon*, kde jsou nastaveny přístupy do databáze aplikace i do databáze IS K2. Dále se zde konfigurují služby aplikace (takzvané services), pluginy a další různé parametry. Pokud chceme spustit aplikaci na lokálním disku, je zde připraven ještě soubor *config.local.neon*, kde si nastavíme přístupy do databáze na localhostu a nemusíme tedy měnit údaje, pokud chceme aplikaci nasadit na ostrý provozní server.

5.4.2 Router

Routování je obousměrné překládání mezi URL a akcemi aplikace. Router zajistí, že když do prohlížeče zadáme adresu, ví kterému presenteru má daný požadavek předat a obráceně. Router se nám zároveň stará o hezké URL ve tvaru */uzivatele/pridat* namísto */?presenter=uzivatele&action=pridat*. Vytvořený router je umístěn v souboru *app/router/RouterFactory.php*. V tomto souboru je i specifikován defaultní presenter a akce, aby aplikace věděla, pokud do prohlížeče zadáme adresu bez parametrů, například mojedomena.cz, jaký obsah má zobrazit. Vytvořený router můžeme vidět na obrázku 5.2.

```
$router[] = $frontRouter = new Route( mask: '[<locale en|cs>/]<presenter>/<action>[/<id>]', array(
    'module' => 'Front',
    'locale' => 'cs',
    'presenter' => 'Konfekcecalculator',
    'action' => 'default',
    'id' => NULL
), flags: Route::SECURED);
```

Obrázek 5.2: Nastavení routeru pro routování aplikace.

5.4.3 Controller

Obdobou controlleru je v Nette presenter. Jedná se o vrstvu, která zpracovává požadavky uživatele a na jejich základě volá příslušnou aplikační logiku. V naší aplikaci nalezneme vytvořené presentery na dvou místech a to v

app/presenters/ kde se nachází obecnější presentery, které mohou být využívány ve více modulech a dále presentery modulu, které nalezneme ve složce *app/FrontModule/presenters*, které se starají o zpracovávání požadavků daného modulu. Nejdříve si popíšeme obecnější presentery:

BasePresenter

Jedná se o základní presenter, který dědí od *Nette\Application\UI\Presenter*. Od tohoto presenteru následně dědí všechny presentery aplikace. BasePresenter tedy slouží pro funkce a nastavení, které mají být dostupné ve všech presenterech. Tento presenter se v projektu nachází již po jeho stažení a je tedy nutné ho pouze upravit a jeho kostru využijeme pro tvorbu dalších presenterů v aplikaci.

PDFBuilder

Jedná se o presenter, který byl vytvořen pro generování PDF s kalkulací, kterou je možné si stáhnout nebo vytisknout, popřípadě odeslat PDF e-mailem. Pro generování PDF byla použita knihovna mPDF. Tento presenter se tedy stará o vygenerování potřebného HTML kódu z šablony, který je pomocí mPDF vyexportován do požadovaného PDF souboru a uložen na disk.

EmailBuilder

Presenter *EmailBuilder* byl vytvořen pro odesílání e-mailů a tvorbu e-mailů. Email je tvořen pomocí předdefinovaných šablon, to znamená, že pokud chceme odeslat nový e-mail vytvoříme si nový objekt *EmailBuilder*, kterému předáme název šablony a data. Data nahradí v šabloně předdefinované konstanty a e-mail je možné odeslat pomocí metody *sendEmail()*. Této metodě předáme odesílatele, příjemce, popřípadě přílohu k e-mailu a e-mail je odeslán.

ErrorPresenter, Error4xxPresenter

Presentery, které jsou volány pokud dojde k nějaké chybě. *Error4xxPresenter* pokud server vrátí stavový kód HTTP začínající 4 tedy chyba na straně kli-

enta například 404 neboli Not Found (požadovaný dokument nebyl nalezen). Třída *ErrorPresenter* je k dispozici pro ostatní stavové chyby.

Další vytvořené presentery jsou již umístěny přímo v modulu a slouží pro odbavování požadavků klienta. Jedná se o presentery umístěné ve složce: *app/FrontModule/presenters*.

BasePresenter

Modul *FrontModule* má svůj vlastní *BasePresenter*, který dějí od presenteru *BasePresenter* popsaného výše a tento presenter je následně děděn presentery v tomto modulu. Díky tomuto dědění se při požadavku na FrontModule nejdříve vykonán tento *BasePresenter*. Zde tedy můžeme ověřit, že uživatel, který zadá požadavek je přihlášen a pokud přihlášen není, vždy ho přesměrujeme pouze na *SignPresenter*, aby byl nucen se přihlásit. Bylo by samozřejmě možné i tento *BasePresenter* vynechat, ale to bychom poté museli ověřovat přihlášení uživatele v každém presenteru zvlášť a náš kód by se zbytečně opakoval. BasePresenter je tedy sdílený pro všechny presentery v tomto modulu.

AjaxPresenter

AjaxPresenter je třída, která byla vytvořena pro odbavování základních ajaxových požadavků. Jsou zde obecné funkce pro přidání, editaci jednotlivých řádků v tabulce databáze. Nachází se zde odbavení požadavku našeptávače, funkce pro stažení PDF ze serveru nebo funkce pro načtení dat z tabulky do JavaScriptové části aplikace. Data jsou následně vracena zpět ve formátu JSON.

SignPresenter

Třída, která se stará o přihlašování a odhlašování uživatelů a zároveň se stará o ukládání všech pokusů o přihlášení jak úspěšných tak neúspěšných.

ProfilePresenter

Pokud se uživatel přihlásí, *ProfilePresenter* se stará o zobrazení základních informací přihlášeného uživatele. Dále se zde vyřizují požadavky na změnu hesla, kde je ověřováno, že uživatel zadal původní heslo správně a nové heslo zadal 2x shodně.

UserPresenter

UserPresenter je třída, která je k dispozici jen uživatelům s rolí Administrátora. Stará se o zobrazení historie přihlášení uživatelů a zobrazení registrovaných uživatelů v systému. V kombinaci s *AjaxPresenter* jsou obsluhovány požadavky na přidání, editaci a blokaci jednotlivých uživatelů.

KonfekcesettingsPresenter

Jedná se o třídu, která je pouze pro uživatele s rolí administrátora a slouží pro nastavení parametrů konfekce. Pro editaci jednotlivých parametrů je využíván *AjaxPresenter*.

KonfekcecalculatorPresenter

KonfekcecalculatorPresenter je třída, která se stará o výpočet kalkulace konfekce. Tato třída obsluhuje zaslaná data odeslaná z formuláře pomocí ajaxového požadavku a výsledek kalkulace vrací ve formátu JSON. Dále je možné generovat PDF s výsledkem kalkulace a nebo výslednou kalkulaci odeslat na e-mail s PDF, které je přiloženo jako příloha. Formulář, z kterého jsou data odesílána můžeme vidět na obrázku 5.3.

CronPresenter

CronPresenter slouží pro aktualizování dat z databáze K2 do databáze aplikace. Pro přístup k tomuto presenteru není potřeba být přihlášen, protože je potřeba, aby šel spustit automatizovaně pomocí cronu v určený čas a data se tak aktualizovala automaticky bez nutnosti se do aplikace přihlašovat.

ErrorPresenter

ErrorPresenter má jedinou metodu a to *renderPermission()*. Slouží nám k tomu, pokud uživatel chce přistoupit k nějaké části aplikace, ke které nemá příslušná oprávnění. Poté je tedy jeho požadavek přesměrován na *ErrorPresenter*, který uživateli zobrazí informaci, že pro přístup k této části nemá dostatečná oprávnění.

5.4.4 Model

Model je datový základ aplikace. Jednotlivé modely, které byly vytvořeny jsou v adresáři *app/model/*. Model nám slouží převážně pro načítání a ukládání dat do databáze. S modely komunikují jednotlivé presentery, pokud potřebují například získat potřebná data z databáze. Framework Nette nám poskytuje speciální vrstvu *Nette\Database* pro pohodlnější práci s databází. Práci s touto vrstvou si ukážeme na příkladu, kdy chceme z tabulky s názvem *tabulka1* získat záznam s hodnotou primárního klíče rovno 1.

database->table('tabulka1')->get(1)

Nette za nás složí SQL dotaz a vrátí nám požadované hodnoty.

V aplikace bylo vytvořeno celkem 5 tříd modelu:

Ajax

Třída, která se stará o vyřizování obecných ajaxových požadavků na serverovou část aplikace.

Konfekce

Třída, která se stará o proces kalkulace. Jsou zde metody pro získání potřebných koeficientů a ostatních informací potřebných k výpočtu kalkulace.

KonfekceSettings

Tato třída vrací informace z tabulek, které jsou potřebné ke kalkulaci konfekce a administrátor aplikace má možnost data uložené v těchto tabulkách měnit. Jedná se například o nastavení jednotlivých strojů a jejich koeficientů.

UserManager

Třída *UserManager* se stará o autentizaci uživatelů, kteří se chtějí do systému přihlásit. V této třídě dochází k ověřování správnosti zadaných přihlašovacích údajů.

UserInfo

Třída, která pracuje s informacemi o uživateli. Má na starosti ostatní akce okolo uživatelů, kromě samotné autentizace.

Cron

Třída, která se stará o přístup k databázi IS K2 a z této databáze načítá data, která následně ukládá do databázového modelu vzniklého systému.

5.4.5 View

View je vrstva, která má na starosti zobrazit výsledek požadavku. V případě Nette je to šablonovací systém Latte. Latte je systém, který zpříjemní práci a zabezpečí výstup před zranitelnostmi jako je XSS [21]. View je vázáno na presentery, to znamená, že každý presenter, které zobrazuje nějaké informace na veiw má svoji šablonu s koncovkou souboru *.latte*, která se použije pro zobrazení informací. Pokud presenter pouze vrací informace ve formátu JSON, pro ajaxový požadavek šablonu nepotřebuje. Šablony, které byly vytvořeny nalezneme v naší aplikaci, stejně jako presentery na dvou místech. Šablony pro obecné presentery EmailBuilder, PDFBuilder jsou umístěny v *app/templates/* a dále pro presentery modulu umístěné v *app/FrontModule/templates*.

Latte

Šablonovací systém Latte nám umožňuje pracovat efektivně včetně dědění jednotlivých šablon a tím se vyhneme zbytečnému opakování kódu. V kořenovém adresáři šablon modulu *FrontModule* proto můžeme nalézt *@layout.latte*. Jedná se hlavní šablonu tohoto modulu, kde si můžeme specifikovat informace, které by se jinak v každé šabloně opakovali: kostra HTML dokumentu, menu a jsou to také například importy JavaScriptových souborů a importy kaskádových stylů. Latte toho ale nabízí mnohem více. V Latte také můžeme využít podmínek, cyklů, chytrého generování odkazů a mnoho dalšího. Práce v tomto šablonovacím systému je tedy velice příjemná a hodí se i pro větší projekty, neboť kód je logicky rozdělen do jednotlivých souborů podle presenterů a jejich akcí a je tedy snadné se v něm orientovat.

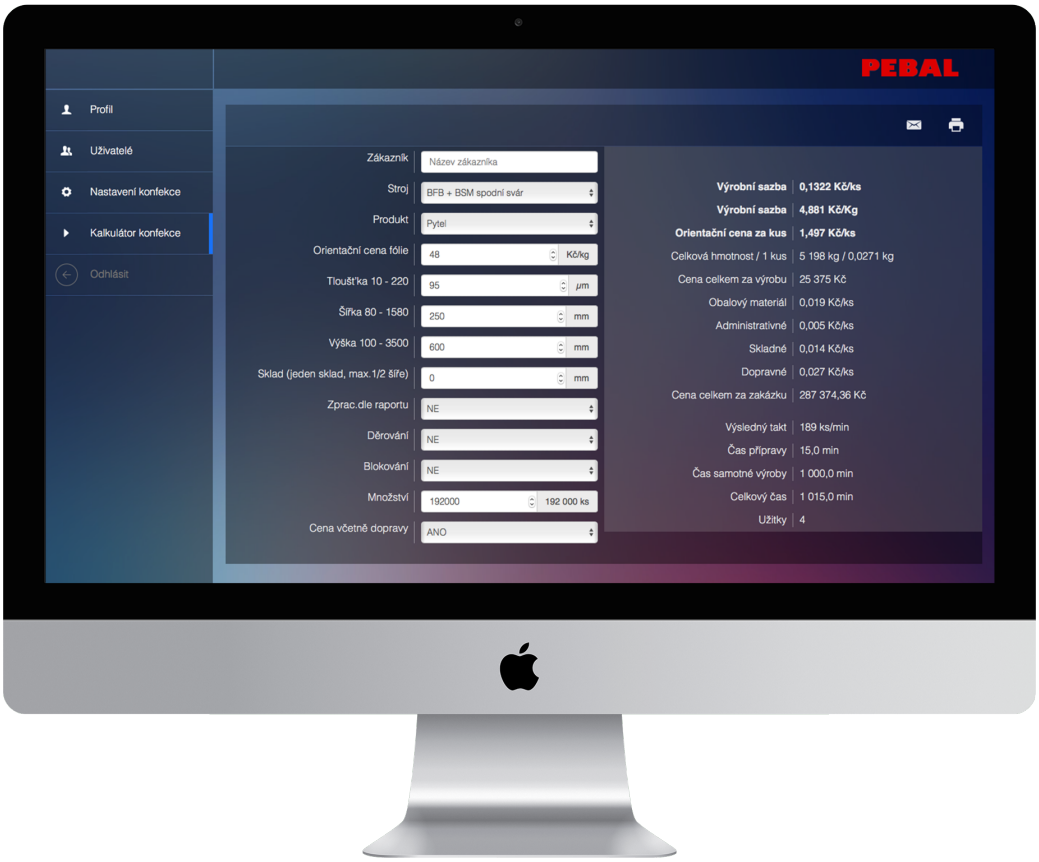
Formulář kalkulace

Jedním z důležitých prvků aplikace je formulář, kde jsou zadávána data pro výpočet kalkulace, který můžeme vidět na obrázku 5.3. Zcela vlevo je hlavní menu aplikace, uprostřed můžeme vidět pole pro zadání vstupních hodnot kalkulace a v pravé části již výsledky zadané kalkulace.

Zde je nutné dbát zřetel i na praktičnost použití, protože kdyby byl použit pouze šablonovací systém a PHP, práce s ním by nebyla pro obchodní zástupce příjemná. Pro každé získání výsledků kalkulace by bylo nutné celý formulář, respektive stránku, odeslat na server, tam data zpracovat a načíst novou stránku s výsledkem. To by znamenalo celou stránku obnovit a vše načíst znovu. Tomuto chování se můžeme vyvarovat díky JavaScriptu, který pracuje na straně klienta a vykonává potřebné funkce v rámci prohlížeče a není tedy nutné celou stránku obnovovat a odesílat ji na server.

Další důvod proč bylo nutné využít JavaScript, je validace formuláře již na straně klienta a jeho dynamické upravování a přegenerování. Je to zapříčiněno tím, že je nutné kalkulovat konfekci pro 5 různých strojů a každý stroj má různé parametry a omezení.

Při použití JavaScriptu je vhodné využít nějakou knihovnu, která má již připravené funkce pro práci s formuláři, komunikaci se serverem a mnoho dalšího. Pro tyto účely byla využita knihovna AngularJS od společnosti Google.



Obrázek 5.3: Formulář pro zadávání parametrů a výsledky kalkulace na desktopu.

AngularJS

JavaScriptový framework AngularJS byl využit pro kontrolu validity vstupů, načítání a odesílání dat na pozadí bez nutnosti obnovovat stránku. Tento framework je navržen tak, že odděluje aplikační logiku od té zobrazovací a pokud zjistí, že se hodnota v aplikační vrstvě změnila, automaticky aktualizuje hodnotu v zobrazovací vrstvě. Toto je jedna z hlavních předností Angularu. Pro presentery aplikace, kde bylo potřeba využít tento framework byl vytvořeny vlastní Angular controllery. Tyto controllery jsou umístěny v adresáři *www/js/frontend/angular/controllers*.

Přednosti tohoto frameworku a jeho využití si ukážeme na formuláři pro výpočet kalkulace. Tento formulář obsluhuje controller *Konfekcecalculator-*

```
<tr>
    <td class="text-right">
        {_general.konfekce_calculator.input.width}
        {{ getInputLimits('width', konfekce.machine).min }} - {{ getInputLimits('width', konfekce.machine).max }}</td>
    <td ng-class="{l}'calc-error' : calcForm.width.$invalid{r}">
        <div class="input-group">
            <input type="number"
                   required name="width"
                   class="form-control"
                   ng-model-options='{l} debounce: debounce {r}'
                   ng-model="konfekce.width"
                   min="{{ getInputLimits('width', konfekce.machine).min }}"
                   max="{{ getInputLimits('width', konfekce.machine).max }}">
            <div class="input-group-addon">{_general.konfekce_calculator.input.width_unit}</div>
        </div>
    </td>
</tr>
```

Obrázek 5.4: HTML kód pole obsluhovaného frameworkem AngularJS.

Controller. Zde se při inicializaci stránky načtou informace o strojích, produktech a další potřebné informace z databáze pomocí ajaxového požadavku. Nenačítají se všechny informace ihned po inicializaci, protože zákazníků je větší a načtení stránky by se tím zpomalilo. Ve formuláři je tedy našeptávač, který po zadání prvního písmena nabídne zákazníky, které tomuto požadavku vyhovují.

Podle zvoleného stroje se zobrazí potřebná pole pro zadávání vstupních dat. Kód jednoho z polí můžeme vidět na obrázku 5.4. Ve formuláři je celkem 22 takových polí. JavaScriptový controller zajišťuje validitu formuláře, aby nebylo možné odeslat vstupní data, která nejsou validní, protože každý stroj má určitá omezení jako je například maximální šířka fólie nebo výška. Nemůže tedy přesáhnout určitý limit, ale navíc je k tomuto limitu připočítána výška klopy, která se také ve formuláři mění. Takže pokud může být výška například v rozmezí 50 až 100 mm a my nastavíme 80 mm, je vše v pořádku a formulář je validní, ale pokud u nějakého stroje navíc připočítává klopa a my výšku klopy nastavíme na 30 mm, tak v tento okamžik se stává pole výšky nevalidní a je potřeba o tom náležitě informovat uživatele aplikace. Zobrazení tohoto omezení můžeme vidět na obrázku 5.5. Kde červené zabarvení pole značí, že je nevalidní a není možné kalkulaci odeslat.

Obrázek 5.5: Zobrazení pole ve formuláři pokud není validní.

Framework Angular se také stará o odesílání dat z formuláře na server, abychom dostali výsledky kalkulace v pozadí. Vše se tedy děje zcela auto-

matizovaně a po změně hodnoty ve formuláři není nutné klikat na tlačítko odeslat, protože model, který je formuláři přiřazen je v controllerem kontrolován a jakmile se změní hodnota ve formuláři provede se kontrola validace a pokud je formulář validní, dojde k jeho odeslání a zobrazení výsledků. Pokud píšeme číslo, které se skládá z 5 cifer, například 10 000, nedojde k odeslání formuláře 5x za sebou postupně s hodnotami 1, 10, 100, 1000 a 1000. K tomuto účelu slouží atribut *ng-model-options*, který je nastaven u všech polí a zde je možné si nastavit prodlení, které bylo v našem případě nastaveno na hodnotu 300 ms, která se ukázala při testech jako optimální. Pokud tedy uživatel píše číslo s více ciframi, formulář po každém stisknutí klávesy počká 300 ms a pokud v tomto rozmezí nedojde k dalšímu stisknutí klávesy, formulář odešle.

Pro složitější validaci jednotlivých polí bylo nutné využít takzvané *directives*. Direktivu si na specifikujeme v controlleru a pak její název přidáme jako atribut do pole kde ji chceme využít. Tato direktiva se nám stará o složitější validace jak bylo uvedeno výše. Například u pole Dolní záložka je omezení 20 - 80 mm ale zároveň je možné dolní záložku nevyužít takže je možné zadat i hodnotu 0 a na požadavek firmy je možné zadat i prázdný řetězec, který se rovná nule. Pro tyto specifická omezení byly vytvořeny speciální direktivy a jednu z nich můžeme vidět na obrázku 5.6.

```
.directive('bottomtab', function() {
    return {
        require: 'ngModel',
        link: function(scope, elm, attrs, ctrl) {
            ctrl.$validators.bottomtab = function(modelValue, viewValue) {
                if (ctrl.$isEmpty(modelValue)) {
                    return true;
                }
                viewValue = parseInt(viewValue);

                if (viewValue === 0 || (viewValue >= 20 && viewValue <= 80)) {
                    return true;
                }

                return false;
            };
        }
    };
});
```

Obrázek 5.6: Kód direktivy pro kontrolu validity pole dolní záložky.

Less

K view patří i kaskádové styly neboli CSS a pro usnadnění práce s nimi byl využit CSS pre-processor Less [22], který nám k CSS přidává možnost využití funkcí, které můžeme opakovat nebo například proměnných. Pokud má náš projekt specifikované základní barvy v Less si je můžeme uložit do proměnné a v celém projektu využívat pouze tyto proměnné. To znamená, že v budoucnu pokud budeme chtít změnit například barvu, stačí nám na jednom místě změnit tuto proměnou a nemusíme tuto barvu měnit na desítkách míst. Ze souboru Less je následně sestaven výsledný CSS soubor.

5.4.6 Pluginy

V aplikaci byly použity i pluginy třetích stran. Pokud potřebujeme do projektu takový plugin nebo knihovnu přidat, použijeme Composer, který byl využit i pro samotné stažení frameworku Nette. V aplikaci byly využity následující pluginy:

mPDF

Knihovna mPDF je knihovna, která z UTF-8 encoded HTML vygeneruje PDF soubor [23]. K instalaci pluginu byl využit následující příkaz.

composer require mpdf/mpdf

Kdyby translator

Překladač Kdyby slouží pro více jazyčné využití aplikace. Překladač je možné do aplikace nainstalovat pomocí *composeru*:

composer require kdyby/translation

Při využití tohoto pluginu je text z šablon seskupen v konkrétním souboru a pokud by v budoucnu bylo nutné aplikaci rozšířit do jiných jazyků, například angličtiny, bude stačit přeložit tento soubor *app/lang/general.cs_CZ.neon* do příslušného jazyka a nebude tedy nutné zasahovat do většiny šablon projektu.

6 Realizace aplikace - mobilní část

Mobilní aplikace bude realizována s využitím frameworku Cordova a frameworku Ionic z důvodů popsaných v kapitole 4.2.2. Pro práci s frameworkem Cordova potažmo frameworkem Ionic je potřeba mít nainstalované Ionic CLI, které nám umožní vytvořit nový projekt Ionicu a následně aplikaci sestavit. K instalaci tohoto CLI je zapotřebí mít nainstalován i Node.js, se kterým se nainstaluje NPM neboli Node Package Manager (správce balíčků), pomocí kterého toto CLI stáhneme a nainstalujeme [24].

npm install -g ionic cordova

6.1 Příprava prostředí

Pokud máme CLI nainstalované můžeme vytvořit prázdný projekt a ten následně testovat popřípadě sestavit pomocí příkazů níže.

ionic start app blank –appname "Pebal Kalkulátor" - Příkaz pro vytvoření projektu aplikace.

ionic serve - Příkaz pro testování aplikace v prohlížeči.

ionic build ios –prod –release - Příkaz pro sestavení produkční verze aplikace pro platformu iOS.

ionic build android –prod –release - Příkaz pro sestavení produkční verze aplikace a vygenerování APK souboru.

6.2 TypeScript

Framework Ionic je založen na frameworku Angular, který byl použit i ve view serverové části aplikace jen s tím rozdílem, že nová verze Ionic 2, která je použita právě pro naší mobilní aplikaci je již vyvíjena v Angular 2 a ten je napsán v TypeScriptu a nikoliv v čistém JavaScriptu [25].

TypeScript je open-source nadstavba programovacího jazyku JavaScript,

kterou spravuje firma Microsoft. Přidanou hodnotou TypeScriptu je typování, třídy, moduly, rozhraní a další vlastnosti, které známe z objektově orientovaného programování. Kód, který napíšeme v TypeScriptu je nutné následně zkompilovat do JavaScriptu.

Jak bylo zmíněno výše, TypeScript je nutné zkompilovat a o tuto práci se stará CLI Ionicu. Pokud spustíme testování aplikace v prohlížeči pomocí příkazu *ionic serve* v příkazovém řádku můžeme pozorovat informace o kompilaci vždy, když nějaký zdrojový soubor změníme a uložíme. Aplikace se tedy automaticky překompiluje a v prohlížeči dojde k jejímu obnovení.

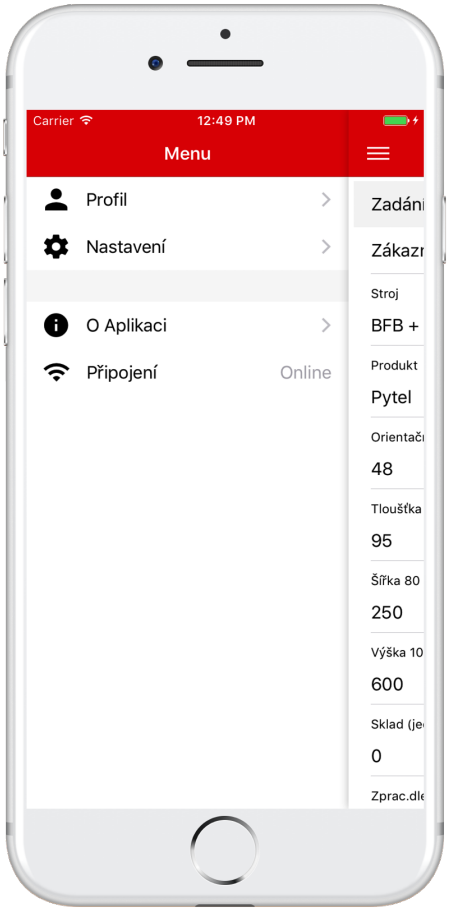
Po kompilaci se všechny soubory s koncovkou *.ts* přeloží do čistého JavaScriptu a sloučí se do jediného výsledného souboru *main.js*, který je navíc minimalizován, to znamená, že se zdrojový kód optimalizuje z hlediska velikosti názvů proměnných a funkcí. Takže pokud máme například funkci se jménem *mojeSuperFunkce*, kompilátor ji přejmenuje například jen na jedno písmenný název *m* a tento název se v celém souboru nahradí. Výsledná kompilace zdrojových kódů z adresáře *src/*, kde primárně píšeme kód aplikace, se zkompiluje do adresáře *www/*.

6.3 Struktura aplikace

Vzhledem k tomu, že Ionic 2 je založen na frameworku Angular 2, struktura aplikace se tedy odvíjí od tohoto frameworku.

6.3.1 App

V adresáři *src/app/* se nachází základní konfigurace projektu. V souboru *app.module.ts* jsou deklarace všech komponent, které potřebujeme do projektu přidat a které jsme vytvořili. V souboru *app.component.ts* je inicializace aplikace, kde si specifikujeme základní informace jako je například stránka, která se má zobrazit po startu aplikace. V souboru *app.html* je hlavní šablona, která bude zobrazena u všech stránek, takže v této šabloně jsem specifikoval postranní menu, které potřebujeme mít všude. Rozvržení menu můžeme vidět na obrázku 6.1, kde je aplikace spuštěna na zařízení Apple iPhone 7.



Obrázek 6.1: Menu aplikace na zařízení Apple iPhone 7.

6.3.2 Pages

Pages neboli stránky jsou základní stavební kámen aplikace a jsou umístěny v adresáři *src/pages/* a v tomto adresři má každá stránka svůj vlastní podadresář, ve kterém jsou umístěny 3 soubory, které můžeme vidět na obrázku 6.2. V souboru s koncovkou *.html* je umístěna HTML šablona, která určuje rozvržení a vzhled dané stránky, která se zkladá z připravených komponent Ionic a ty nám zaručí, že stránka se stejným zdrojovým kódem bude vypadat jinak na platformě iOS a jinak na platformě Android [26]. V souboru s koncovkou *.ts* je aplikační logika dané stránky psaná v TypeScriptu. Posledním souborem v tomto adresáři je *.scss*, což jsou kaskádové styly s využitím SASS [27]. SASS funguje na stejné bázi jako LESS, využitý ve view serverové části aplikace a popsaný v kapitole 5.4.5. SASS pomáhá k dobré organizaci CSS

stylu a zjednodušení práce s ním. V aplikaci bylo vytvořeno 6 následujících stránek:

Obrázek 6.2: Struktura jedné stránky ve frameworku Ionic.

Login

Stránka login je úvodní stránka, která se zobrazí vždy po načtení aplikace. Pokud se uživatel již v minulosti přihlásil, má předvyplněný e-mail, ale heslo musí z bezpečnostních důvodů zadat po každém spuštění.

Profile

Stránka profilu zobrazuje základní informace o přihlášeném uživateli, jako je jeho jméno a e-mail. Na této stránce se může uživatel také odhlásit z aplikace.

Settings

Na stránce nastavení je možné změnit si heslo pro přihlášení do aplikace. Pro desktopovou i mobilní aplikaci je jeden a ten samý účet. Pokud tedy dojde ke změně hesla na mobilní platformě, je potřeba toto nové heslo začít používat i na desktopové verzi aplikace.

Calculation

Stránka kalkulace je hlavní stránkou aplikace. Na obrázku 6.3 můžeme vidět kalkulaci na zařízení Apple iPad Air 2, kde je využit větší displej tabletu a kalkulace je rozdělena do dvou sloupců, aby při zadávání vstupních parametrů nebylo nutné scrollovat a všechny informace byly viditelné na první pohled. Na této stránce je také možné vygenerovat PDF soubor s kalkulací, který se

následně otevře v prohlížeči, kde je možné s ním dále pracovat nebo odeslat PDF jako přílohu do e-mailu. Rozpracovanou kalkulaci si také můžeme uložit do zařízení a vrátit se k ní později.

Formulář pro kalkulaci v mobilní aplikaci funguje na stejné bázi jako formulář v desktopové verzi aplikace, ale vzhledem k tomu, že webová aplikace využívá framework AngularJS a mobilní aplikace je založena již na novějším Angularu 2, který je psán v TypeScriptu, nebylo možné využívat totožný kód.

Pro validaci formuláře v Angularu 2 byla využita třída *FormGroup* [28]. Při vytváření nové instance *FormGroup* je jako parametr předán objekt, ve kterém si na specifikujeme jednotlivá pole formuláře a přiřadíme jim validátory ať základní validátory, které jsou v knihovně jako například *required* ten nám říká že toto pole je povinné nebo si vytvoříme speciální validátory pro dané pole, kde je validace složitější jako například validace výšky a dalších parametrů, které závisí na dalších polích.

Customer-search

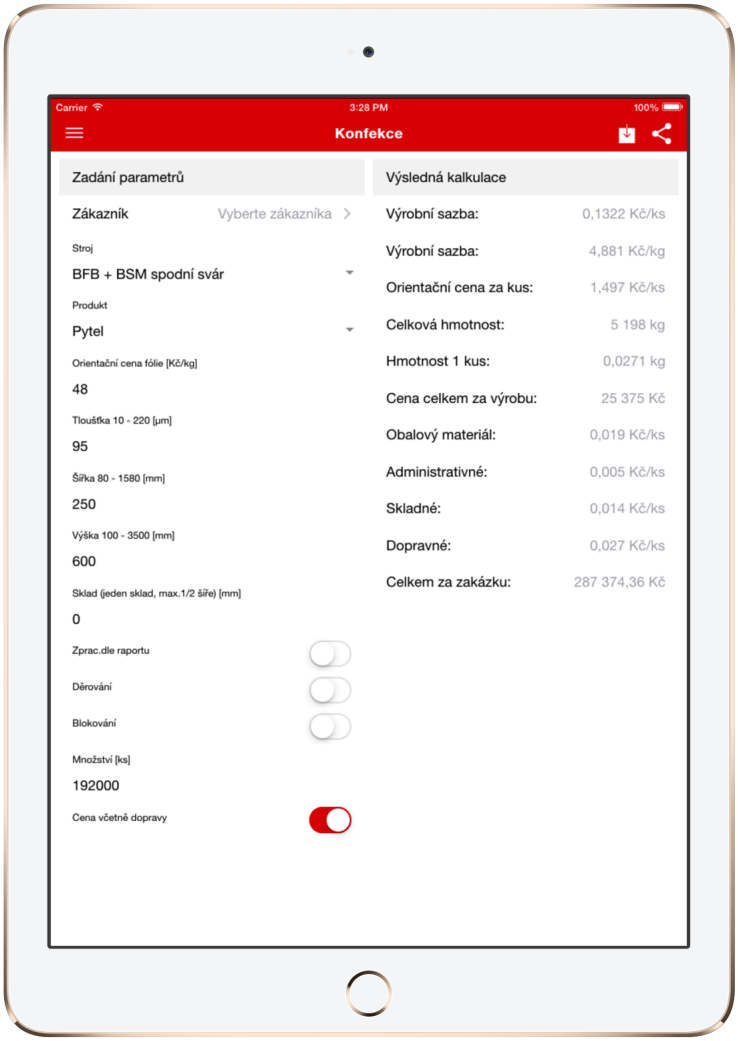
U každé kalkulace je možné specifikovat zákazníka, který je převzat z informačního systému K2 a vzhledem k velkému množství zákazníků je načítání dynamické na základě zadaného názvu. Na této stránce si do pole vyhledat napíšeme první znaky z názvu zákazníka a v seznamu níže se nám zobrazí zákazníci, kteří tomuto názvu odpovídají. Po kliknutí na zákazníka se zákazník vybere a je zpět zobrazena stránka s kalkulací.

About

Stránka se základními informacemi o aplikaci jako je například verze nainstalované aplikace.

6.3.3 Classes

Vytvořené třídy jsou umístěny v adresáři *app/classes/*. Třídy slouží pro deklaraci objektu, ze kterého si poté můžeme vytvořit instanci. Díky TypeScriptu si můžeme jednotlivé proměnné typovat a přiřadit jim datový typ



Obrázek 6.3: Zobrazení kalkulace konfekce na zařízení Apple iPad Air 2.

námi vytvořené třídy. Třídy nám slouží k zpřehlednění kódu a odstranění základních chyb, kterých se můžeme dopouštět pokud bychom vytvářeli jen základní objekty bez typování. V aplikaci jsou vytvořeny dvě třídy: *Customer* pro přenášení informací o jednotlivých zákaznících a třída *User*, kde máme informace o přihlášeném uživateli.

6.3.4 Pipes

Pipes jsou funkce, které nám slouží pro transformaci dat a můžeme je využít například v šablonách. Vytvořené pipes jsou umístěny v adresáři *app/pipes/* a byly vytvořeny celkem 3. Pipe byla například použita k filtraci produktů, které se vážou k určitému stroji. Využití můžeme vidět níže:

**ngFor="let item of konfekce_product.data | machinePipe:form.value.machine"*

Kde *ngFor je direktiva cyklu v Angular 2 a v proměnné *konfekce_product.data* jsou uloženy všechny produkty pro všechny stroje. Proto využijeme námi nadefinovanou Pipe *machinePipe*, které za dvojtečkou definujeme ID stroje, který nás zajímá. Toto ID je uloženo v proměnné *form.value.machine* a cyklus nebude pracovat se všemi produkty všech strojů, ale jen s těmi, které nás v současné chvíli zajímají.

6.3.5 Services

Services jsou služby, pomocí kterých si sdílíme data mezi jednotlivými stránkami nebo kód, který se často opakuje. Vytvořené služby jsou umístěny v adresáři *app/services/* a byly vytvořeny celkem 4 služby. Jedna z takových služeb je *AjaxService*, ve které jsou nadefinovány funkce pro komunikaci se serverem. Pokud stránka *Calculation* potřebuje získat data s výslednou kalkulací ze serveru, předá vstupní data této službě konkrétně funkci *konfekceCalculation()*, která se následně spojí se serverem a pokud vše proběhne v pořádku, vrátí zpět data s výsledky kalkulace.

6.4 Ikona a úvodní obrazovka

Nedílnou součástí každé mobilní aplikace je i ikona aplikace a úvodní obrazovka (tzv. splash screen), která se zobrazí po kliknutí na ikonu a uživatel ji vidí do úplného načtení aplikace. Ionic má pro tuto problematiku připravenou velice zajímavou funkci a tou je:

ionic resources

Tento příkaz nám nahraje vytvořenou ikonu a úvodní obrazovku na server,

kde dojde ke zpracování těchto obrázků a vygenerování potřebných velikostí pro různé velikosti telefonů a tabletů. Takže je nutné připravit ikonu ve velikosti 1024 x 1024 pixelů a úvodní obrazovku ve velikosti 2208 x 2208 pixelů a vložit je do adresáře *resources/* a poté spustit výše uvedený příkaz. O vše ostatní se již postará tato funkce. Vygeneruje potřebné velikosti a odkazy na jednotlivé vygenerované obrázky vloží do konfiguračního souboru *config.xml*.

6.5 GUI

Aplikace má jiné uživatelské rozhraní kalkulátoru pro mobilní zařízení a tablety. Pokud je aplikace spuštěna na tabletu, část pro zadávání hodnot je v levém sloupci a část s výsledky kalkulace je v pravém sloupci. Všechna data jsou tedy přehledně na jedné obrazovce bez nutnosti scrollování. Pokud je aplikace spuštěna na mobilním zařízení, tam by již toto rozvržení nebylo možné a výsledky kalkulace jsou tedy umístěny pod formulář pro zadávání hodnot, takže pro zobrazení výsledků je již nutné scrollovat.

6.6 Pluginy

Pro instalaci pluginů třetích stran je nutné využít správce balíčků NPM. Níže si ukážeme instalaci pluginu pro vícejazyčnost aplikace.

npm install @ngx-translate/core –save

Tento plugin nám slouží na úrovni Angularu, není zde potřeba využívat nativní části zařízení a instalaci tohoto pluginu tedy můžeme vidět jen v adresáři modulů *node_modules*. Některé pluginy ale nativní kód ke svému fungování potřebují, a proto je nutné instalovat nativní část pomocí Ionic CLI. Jeden z takových pluginů je například plugin pro zjištění aktuální verze aplikace. Ten nainstalujeme pomocí následujícího příkazu.

ionic plugin add cordova-plugin-app-version

Tento příkaz stáhne plugin do složky */plugins*, kde můžeme vidět všechny instalované pluginy a jsou zde umístěny nativní kódy pro všechny platformy, pro které budeme chtít sestavit aplikaci. Pokud se tedy podíváme na konkrétní plugin *cordova-plugin-app-version*, tak v adresáři *plugins/cordova-plugin-*

app-version/src budou podadresáře android i ios, pro které chceme aplikaci následně sestavit. V aplikaci byly využity následující pluginy a knihovny:

Translate

Plugin *ngx-translate* nám slouží pro vícejazyčnost aplikace. To znamená, že texty v aplikaci nejsou umístěny v šablonách, ale jsou na jednom místě a pokud bude potřeba aplikaci rozšířit do dalších jazyků, například angličtiny, bude stačit přeložit tento soubor a přidat možnost volby jazyka do aplikace. Plugin je dostupný na adrese: *https://github.com/ngx-translate/core*.

Moment

Plugin *angular2-moment* jsem využil pro zobrazení datumu a času u uložených kalkulací. Pluginu předáme pouze čas a on nám vrátí datum ve formátu například *před 10 minutami* nebo *před 1 dnem*. Slouží tedy k rychlejší orientaci časové informace. Plugin je dostupný na adrese:
https://github.com/urish/angular2-moment.

App Version

Plugin *cordova-plugin-app-version* již potřebuje ke své funkčnosti nativní kód. Tento plugin nám vrátí informaci o verzi aplikace, která je poté zobrazena na stránce *O aplikaci*. Plugin je dostupný na adrese: *https://github.com/whiteoctober/cordo*
plugin-app-version.

Google Analytics

Plugin *google-analytics-plugin* slouží pro propojení aplikace s analytickým nástrojem Google Analytics. Plugin je dostupný na adrese:
https://github.com/danwilson/google-analytics-plugin.

Network

Plugin *cordova-plugin-network-information* slouží pro získávání informací o připojení k internetu v mobilní aplikaci. Takto můžeme ověřit, že k odeslání kalkulace nedošlo z důvodu připojení k internetu a uživateli zobrazit bližší informace o důvodu neodeslání kalkulace. Plugin je dostupný na adrese: *https://github.com/apache/cordova-plugin-network-information.*

V aplikaci jsou použity další pluginy, jako je například *keyboard* nebo *splashscreen*, ale ty jsou již součástí základní instalace frameworku Cordova.

6.7 Verzování aplikace

Pro správu verzí mobilní i serverové části aplikace byl využit verzovací systém Git spolu se službou Bitbucket, dostupné na *bitbucket.org*. Tato služba nám umožňuje verzovat a ukládat data v cloudu na vzdálených serverech, a tak máme data uložená i mimo lokální disk. Tato služba je zdarma při využití účtu do 5 uživatelů.

Bitbucket také nabízí bug tracking system, který byl využit k přehlednému zaznamenávání chyb a dalších požadavků na aplikaci.

7 Zabezpečení aplikace

Zabezpečení aplikace je velice důležité, protože se v aplikaci nacházejí citlivé informace. Pokud by zabezpečení bylo bráno na lehkou váhu, mohlo by to mít neblahý dopad na fungování celé společnosti, protože by se tato citlivá data mohla dostat například do rukou konkurence. Pro zabezpečení je tedy důležité, aby do aplikace měly přístup jen oprávněné osoby a aby data, která se přenáší mezi klientem a serverem byla šifrována a nebylo možné je odposlouchávat.

7.1 Autentifikace uživatelů

Základním požadavkem v zabezpečení aplikace je, aby byla chráněna přístupovými údaji, před neoprávněnými uživateli. Uživatel je tedy před vstupem do aplikace požádán o zadání e-mailu a hesla. Při registraci nového uživatele do aplikace, se do databáze ukládá pouze otisk neboli hash daného hesla. Pro vytvoření hashe je použita třída *Nette\Security\Passwords* [29], která se postará o zahashování hesla pomocí moderního algoritmu *bcrypt* [29]. Hashovací funkce *bcrypt* je adaptivní funkce, to znamená, že u ní můžeme záměrně zvýšit počet iterací, čímž dojde k jejímu zpomalení. Tím je zajištěna ochrana proti útokům hrubou silou. Pokud by tedy došlo k odcizení databáze uživatelů, nemělo by to vést k prolomení hesla uživatelů, protože z ukradeného hashe není možné zrekonstruovat původní podobu hesla.

Aplikace si ukládá každé přihlášení do databáze na serveru do tabulky *users_login_history* a shromažďuje informace o přihlášení. U každého přihlášeného uživatele je uložena IP adresa, datum přihlášení, čas přihlášení a jestli se přihlašuje z webového prohlížeče nebo z mobilní aplikace. Pokud se uživateli nepovede přihlásit, je toto nezdařené přihlášení také zaznamenáno. Administrátor aplikace tedy všechna jednotlivá přihlášení vidí v administraci aplikace a má možnost například uživatele okamžitě zablokovat. Pokud je uživatel na nějakém zařízení přihlášen a dojde k zablokování, ihned při pokusu o nějakou akci dojde k odhlášení a do odblokování administrátorem, se daný uživatel nemůže do aplikace přihlásit. Pokud je uživatel zablokován, po pokusu o přihlášení se mu zobrazí hláška, že byl zablokován a je nutné kontaktovat administrátora.

Uživatelé aplikace jsou při nečinnosti automaticky odhlášeni. A to tak, že v prohlížeči je to již po 20 minutách nečinnosti a v mobilním zařízení po 120 minutách nečinnosti. U mobilních aplikací je doba prodloužena z toho důvodu, že zaměstnanci mají mobilní zařízení navíc chráněna heslem, které je požadováno při každém zhasnutí displeje. Tato doba, kdy mají být uživatelé automaticky odhlášeni se dá nastavit v konfiguračním souboru Nette *app/config/config.neon*

7.2 Šifrovaná komunikace

Další bezpečnostním aspektem, kterému je potřeba se věnovat je, aby komunikace mezi klientem a serverem byla šifrovaná. A tedy využití HTTPS umožňující zabezpečenou komunikaci v síti. HTTPS využívá protokol HTTP spolu s protokolem SSL nebo TLS [30]. TLS je nástupce SSL a jedná se o protokol, který umožňuje aplikaci komunikovat po síti šifrovaně a zabraňuje odposlouchávání a falšování zpráv. V případě této aplikace je využití HTTPS nezbytné, pokud například někdo ze zaměstnanců používal aplikaci v kavárně, kde by administrátor sítě nebo kdokoliv jiný odposlouchával nezabezpečenou aplikaci. Celkem snadno by mohl odposlechnout přihlašovací údaje uživatele aplikace.

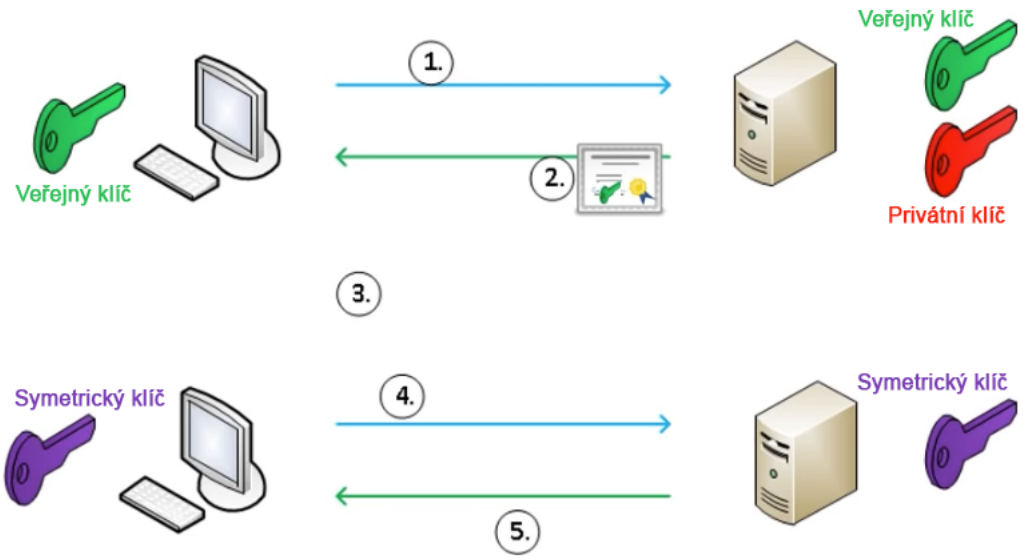
7.2.1 Navázání šifrované komunikace

Při zabezpečené HTTPS komunikaci klient - server je autentizován server, to znamená, že dojde k ověření totožnosti serveru, pokud má vystaven důvěryhodný certifikát od certifikační autority. Pokud důvěryhodný certifikát nemá, je šifrovaná komunikace stále možná, ale budeme prohlížečem upozorněni, že bychom s daným nedůvěryhodným serverem neměli komunikovat. Aby došlo k šifrované komunikaci mezi klientem a serverem a klient si mohl ověřit, že může serveru důvěřovat a je skutečně tím za koho se vydává, je před zahájením komunikace mezi klientem a serverem potřeba několika kroků. Protokoly SSL respektive nástupce TLS jsou založeny na asymetrické kryptografii tedy veřejném a privátním klíči. Veřejný klíč má k dispozici prohlížeč, aby mohl zprávu, kterou odesílá na server zašifrovat, ale již ji nemůže dešifrovat, to

může pouze server, který má k dispozici privátní klíč.

Na obrázku 7.1 je znázorněno navázaní zabezpečené komunikace mezi klientem a serverem.

1) Klient požádá server o SSL/TLS komunikaci.
2) Server odpoví s SSL/TLS certifikátem, který obsahuje veřejný klíč.
3) Klient si certifikát ověří.
4) Klient vygeneruje symetrický veřejný klíč a zašle ho serveru.
5) Šifrovaná komunikace mezi klientem a serverem může začít.



Obrázek 7.1: Navázání zabezpečená komunikace mezi klientem a serverem.

7.3 Certifikáty

Aby bylo možné používat HTTPS, musí být na serveru nainstalován certifikát. Ty můžeme rozdělit na dva základní typy důvěryhodný a nedůvěryhodný. Nedůvěryhodný si můžeme vystavit sami a je zdarma. Důvěryhodný je vystaven legitimní důvěryhodnou certifikační autoritou jako je například Comodo, RapisSSL, GeoTrust a další. U důvěryhodného certifikátu zajistíme bezpečnost přenášených dat a zároveň uživateli dodává důvěru tím, že v adresní liště prohlížeče vidí adresu webu zeleně nebo je u některých prohlížečů

zámeček (zobrazení zabezpečené komunikace se v jednotlivých prohlížečích liší a také záleží na použitém druhu certifikátu). Jednotlivé důvěryhodné certifikáty se liší cenou a tím co nabízejí. Do ceny se promítne jaký druh certifikátu zvolíme, od jaké certifikační autority certifikát zakoupíme a na jak dlouhou dobu certifikát pořídíme.

7.3.1 Druhy certifikátů

Jsou 3 základní druhy certifikátů [31], které se liší cenou a způsobem ověřování daného vlastníka domény. Tedy následnou důvěryhodností. Každý prohlížeč zobrazuje druh použitého certifkátu jinak. Na obrázku 7.2 jsem znázornil zobrazení certifikátu na prohlížečích Chrome, Safari a Firefox. Jednotlivé prohlížeče tedy i rozhodují o tom, jestli uživateli zobrazí daný web jako důvěryhodný nebo zobrazí upozornění, že daný web je nedůvěryhodný a měli bychom danou stránku opustit. Například firma Google Inc. oznámila, že jejich prohlížeč Chrome nebude zobrazovat certifikáty vydané společností Symantec jako důvěryhodné z toho důvodu, že tato společnost pochybila při vystavování důvěryhodných certifikátů a byla za to takto potrestána [3]. Je tedy důležité při vybírání certifikátu dbát i na vydavatele daného certifikátu a ne jen na cenu.

DV (Domain Validation)

Certifikáty s ověřením vlastníka domény. Jedná se o automatizovaný proces v reálném čase, který ověřuje, že žadatel o certifikát je vlastníkem dané domény. Ověření probíhá třemi různými způsoby, ze kterých si žadatel může vybrat. První způsob je odesláním e-mailu na adresu admin@mojedomena.cz, kde žadatel klikne na vložený link a tím potvrdí, že je vlastníkem mojedomena.cz. Druhým způsobem je nahrání specifického souboru, který nám ověřovatel vygeneruje na server, kam odkazuje naše doména, například domena.cz/overeniABC123.html a posledním způsobem jak žadatele ověřit je nahrání náhodného řetězce, který nám je vygenerován a následné vložení tohoto řetězce do TXT záznamu DNS mojedomena.cz. Tento proces je tedy plně automatizován a po splnění jednoho z výše uvedených ověření je nám certifikát obratem vystaven. Toto neplatí u následujících dvou typů certifikátů. U OV i EV je již nutný zásah zaměstnance certifikační autority, který musí ověřit zda má žadatel na vydání certifikátu právo. Certifikáty DV jsou

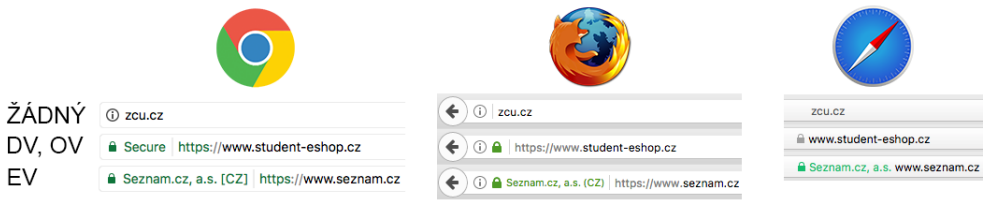
vydávány s platností na 1-3 roky.

OV (Organization Validation)

Certifikáty ověřené na této úrovni obsahují ověřené informace o vlastníkovi certifikátu a návštěvník si tak může být jistý, že web je skutečně provozovaný tím, za koho se vydává. Zvyšuje tak úroveň důvěry. Certifikáty OV jsou vydávány s platností na 1-3 roky.

EV (Extended Validation)

Rozšířené ověření představuje nejvyšší úroveň důvěry a prohlížeče to dávají najevo zelenou barvou označující název firmy nebo instituce. Získání tohoto certifikátu je velmi zdlouhavá záležitost, a proto by každý měl zvážit, jestli tento certifikát skutečně potřebuje. Tyto certifikáty jsou vystavovány s platností 1-2 roky a vzhledem ke složitosti ověřování je rozumnější rovnou zvolit 2 roky.



Obrázek 7.2: Zobrazení druhů certifikátů na jednotlivých prohlížečích.

7.3.2 Získání certifikátu

Pro testovací účely tohoto projektu byl zakoupen DV certifikát. Pro zakoupení certifikátu jsem zvolil server *https://cheapsslsecurity.com*, kde jsem zakoupil certifikát od certifikační autority RapisSSL v ceně $7,99 v přepočtu

necelých 200 Kč. Pokud bych zakoupil certifikát rovnou na 3 roky, byla by cena nižší a to $6,66 na rok v přepočtu 166 Kč.
Po zaplacení daného poplatku můžeme rovnou přejít na ověření domény. Nejprve je nutné si vygenerovat CSR soubor (Certificate Signing Request). O tento soubor můžeme požádat administrátora serveru nebo použít generátor na adrese: *https://cheapsslsecurity.com/ssltools/csr-generator.php*. Po vygenerování CSR si uložíme CSR řetězec a privátní klič. Dále vložíme CSR řetězec do formuláře na webu a zvolíme způsob ověření domény. Po úspěšném ověření domény nám již nic nebrání ke stažení certifikátu od certifikační autority. V našem případě nalezneme certifikát ke stažení v sekci *Complete Orders*, kde si ho můžeme kdykoliv po přihlášení stáhnout.

7.3.3 Instalace certifikátu

Pokud úspěšně projdeme procesem získání certifikátu, můžeme přejít k jeho instalaci. Pro instalaci certifikátu budeme potřebovat samotný certifikát, který nám poskytla certifikační autorita a privátní klič, který jsme si vygenerovali společně s CSR řetězcem. Pro testovací účely byl využit hosting od společnosti Wedos a.s., kde je možné v administraci hostingu intuitivně certifikát nainstalovat. K instalaci je nutné aktivovat *HTTPS s vlastním certifikátem na doméně* a poté do příslušného formuláře vložit privátní klič a do dalšího formuláře vygenerovaný certifikát. Další nutnou úpravou pro funkčnost certifikátu je upravit *.htaccess* a v tomto souboru aktivovat automatické přesměrování na *https://*. Do tohoto souboru bylo nutné přidat následující řádky viz obrázek 7.3. Tyto dva řádky nám říkají, že pokud není již HTTPS aktivováno, je nutné přesměrovat dotaz z protokolu HTTP na protokol HTTPS.

```
RewriteCond %{HTTPS} !=on
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

Obrázek 7.3: Aktivace HTTPS v .htaccess.

7.3.4 Ověření certifikátu

To, že je certifikát správně nainstalován na serveru ověříme jednoduše zadáním webové adresy do prohlížeče s protokolem HTTPS tedy *https://mojedomena.cz*

a nebo jsou k dispozici sofistikovanější metody k ověření. Jedna taková služba se nachází na adrese: *https://www.ssllabs.com/ssltest/*. Ssllabs nám ověří funkčnost námi nainstalovaného certifikátu a zobrazí i rating zabezpečení.

8 Offline práce v aplikaci

Možnost offline práce v aplikaci je z hlediska technologií realizovatelná, ale vzhledem k požadavkům na aplikaci není možná plná funkcionalita. Je to dáno tím, že výpočty kalkulací je potřeba udržet aktuální a často se mění parametry výpočtu na základě ceny vstupních komodit. Pro firmu Pebal je tedy velice důležité, aby byly výpočty vždy aktuální a probíhaly centralizovaně na serveru, ke kterému se připojují klienti se žádostí o výpočet. Dalším důvodem je nutnost uživatele před vpuštěním do aplikace ověřit a to také v tomto případě není možné, pokud by byla aplikace offline, protože by nebylo možné v reálném čase blokovat uživatele. Na základě těchto požadavků byla snaha maximálně usnadnit offline použití aplikace s dodržením toho, že výpočty a ověřování uživatelů musí probíhat online.

8.1 Lokální úložiště

Abychom v aplikaci mohli pracovat offline, je zapotřebí uložit data do zařízení, aby byla dostupná i při dalším spuštění aplikace, protože offline zařízení nemá možnost uložit data na vzdálený server. Pro uložení dat na straně zařízení můžeme využít následující možnosti.

8.1.1 LocalStorage

LocalStorage poskytuje jednoduché klíč-hodnota úložiště a je podporováno všemi platformami dostupnými pro framework Cordova [32].

Výhody

- Je podporováno všemi Cordova platformami.
- Jednoduché synchronní použití (není potřeba využívat callback).
- Není potřeba využívat dalších pluginů.

Nevýhody

- Mohou být uložena jen data, která je možné převést na řetězec.
- Omezená velikost dat uložených v localStorage na většině zařízení okolo 5MB.

8.1.2 SQLite

SQLite poskytuje API pro ukládání dat ve strukturované databázi. Data jsou z databáze získávána pomocí standardních SQL dotazů. Hodí se tedy pro složitější struktury a objemnější data.

Výhody

- Možnost v datech vyhledávat pomocí SQL dotazů.
- Dobrý výkon díky možnosti indexování dat.
- Možnost využití transakcí pro ukládání a načítání dat.

Nevýhody

- Není podporováno všemi platformami frameworku Cordova.
- Nutnosti instalace pluginu pro práci s SQLite.
- Složitější využití.

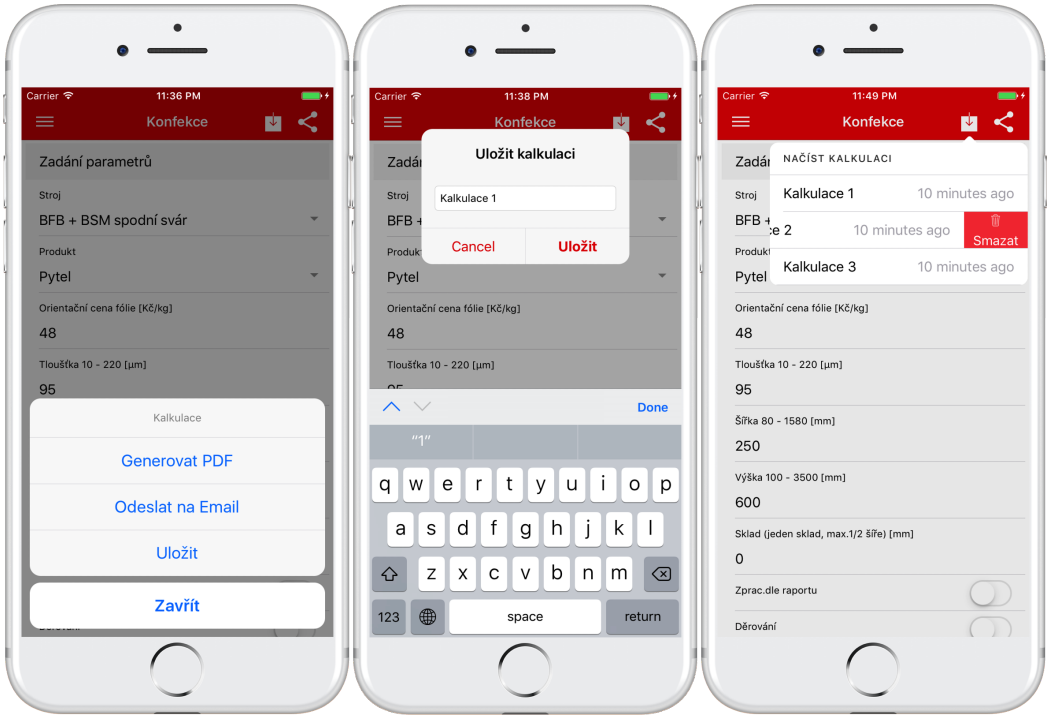
8.2 Navržené řešení

K uložení dat z rozpracované kalkulace bylo využito úložiště LocalStorage, které splňuje vše, co od daného úložiště potřebujeme. Při uložení rozpracované kalkulace jsou data z formuláře uložena do LocalStorage a při načítání aplikace jsou opět tato data načtena a je možné s nimi pracovat.

Pokud dojde v terénu k výpadku internetového připojení, je možné s aplikací pracovat. Není možné z výše popsaných důvodů získat aktuální výpočet

kalkulace, ale pokud je uživatel v aplikaci přihlášen, může si rozpracovanou kalkulaci uložit na později. Když je opět připojení k internetu dostupné, může se k ní vrátit a kalkulaci dokončit.

V pravém horním rohu stránky kalkulace stačí kliknout na ikonku a ve spodní části displeje se zobrazí action sheet, kde zvolí uložit (obrázek 8.1 první zařízení zleva). Poté se zobrazí okno pro zadání názvu kalkulace (obrázek 8.1 zařízení uprostřed) a daná kalkulace se uloží do localStorage v zařízení. Když je uživatel opět online, může si dané kalkulace opět načíst po kliknutí na ikonku v pravém horním rohu. Jednotlivé kalkulace jsou seřazeny podle datumu a času, kdy byly uloženy. Pokud uživatel již uloženou kalkulaci nepotřebuje, může v seznamu uložených kalkulací pomocí swipe gesta vlevo danou kalkulaci vymazat (obrázek 8.1 první zařízení zprava).



Obrázek 8.1: Uložení a následné vymazání uložené kalkulace.

9 Ověření funkcionality

Ověření funkcionality je v každém systému velice důležité a testování by nemělo být nijak opomíjeno. Vzhledem k tomu, že se jedná o práci, která bude využívána společností, která dosahuje ročně stamiliónových obratů, tak případné chyby by mohly mít nedozírné následky. Pro odhalení chyb byly použity unit testy pro serverovou část aplikace, dále proběhlo testování v simulátorech pro mobilní aplikaci a testování v samotném prohlížeči pro webovou část aplikace. Pokud i přesto došlo k nějaké chybě, v aplikaci je sofistikované logování, které v případě chyby na vše upozorní a je poté možné chybu odstranit.

9.1 Unit testy

Pro automatické ověřování zdrojového kódu jsou použity Unit testy. Pro použití Unit testů bylo nutné doinstalovat do Nette modul *Tester* [33].

9.1.1 Instalace

Unit testy nejsou součástí základního balíčku Nette, proto je nutné si je nejdříve doinstalovat. Pro stažení a instalaci Tester balíčku je možné použít *composer* a nainstalovat ho pomocí následujícího příkazu:

composer require –dev nette/tester

9.1.2 Testovací scénáře

Testy jsou rozděleny do 7 tříd. Celkem bylo vytvořeno 25 testů se 109 assercemi. Asserce je ověření správnosti výsledné hodnoty. Pokud máme test na přihlášení uživatele, můžeme například pomocí jedné asserce ověřit zda je uživatel úspěšně přihlášen a pokud jsme přihlašovali uživatele se jménem Tomáš, zda hodnota se jménem je rovna jménu Tomáš.

KonfekceTest v této třídě je testován model Konfekce, funkce v tomto modelu jsou potřebné pro výpočet kalkulace konfekce. Jsou zde ověřovány koeficienty potřebné pro výpočet.

UserinfoTest v této třídě jsou testovány funkce, které vrací informace o uživatelích, kteří jsou v databázi.

UserManagerTest zde jsou testována funkce pro autentizaci uživatele, který se chce přihlásit do systému.

CronTest zde je testována funkčnost aktualizace dat z IS K2 do databáze vytvořené aplikace.

AdhesiveTypePriceTest v této třídě se testují výpočty cen jednotlivých lepících pásek na klopu.

KonfekceCalculationTest zde jsou testovány vzorce pro výpočet cen produktů k jednotlivým strojům konfekce.

HollowPriceTest v této třídě se testuje výpočet ceny dutinky.

9.1.3 Spuštění Testeru

Pokud máme nainstalován Tester a vytvořené testovací scénáře, nic nám nebrání ve spuštění Unit testů. Pro spuštění použijeme příkaz níže:

vendor/bin/tester tests/ -p php

První část příkazu *vendor/bin/tester/* je cesta, kde se nachází skript pro provádění testů a druhá část *tests/* je cesta k testovacím scénářům. Je možné uvést konkrétní soubor, který se má otestovat například *tests/mujTest01.phpt* a nebo pouze složku a to znamená, že se provedou všechny testy, které jsou v dané složce. Poslední část příkazu *-p php* je cesta k PHP interpretu, který se má spustit.

9.1.4 Vyhodnocení provedených testů

Po dokončení všech testů můžeme vidět vyhodnocení na obrázku 9.1. Na tomto obrázku vidíme, že bylo úspěšně otestováno 7 testovacích tříd. Pokud se takto spouští více testovacích tříd na jednou, každá úspěšně otestovaná třída reprezentuje tečku. Na obrázku tedy vidíme 7 teček za sebou. Třídy se testují postupně podle abecedního pořadí. Pokud tedy u nějaké testovací třídy dojde k chybě, zobrazí se místo tečky písmenko *F*.

Obrázek 9.1: Výsledek Unit testů v Nette.

9.2 Logování aplikace

Pokud nastane na serverové části aplikace chyba nebo výjimka, vše je zalogováno do adresáře */log* a je obratem odeslán e-mail s informací o chybě. Pokud v reálném provozu nastane nějaká chyba, administrátor je o ni obratem informován pomocí e-mailu a může na ni zareagovat. K logování chyb byla využita knihovna *Tracy*, která je součástí instalačního balíčku Nette. Ve složce */log* je vytvořen soubor, ve kterém je velice podrobně popsáno k jaké chybě došlo, na jakém řádku chyba vznikla a další informace jako například jaký uživatel byl přihlášen, když k dané chybě došlo. Logování je tedy nezbytnou součástí aplikace a díky *Tracy* je možné velice rychle na chyby reagovat a následně je analyzovat a opravit.

9.3 Testování mobilní verze

I mobilní verzi aplikace bylo nutné otestovat. Při vývoji s frameworkem Cordova můžeme pracovat s řadou testovacích nástrojů.

9.3.1 Ladící konzole

Základním prvkem pro testování je ladící konzole, do které je možné vypisovat kontrolní log pomocí příkazů, které můžeme vidět na obrázku 9.2. Výhodou těchto výpisů je, že se zobrazují pouze v konzoli a nemusí se při sestavení ostré verze nijak odstraňovat.

```
console.log("Logovací zpráva");
console.warn("Varovací zpráva");
console.error("Chybová zpráva");
```

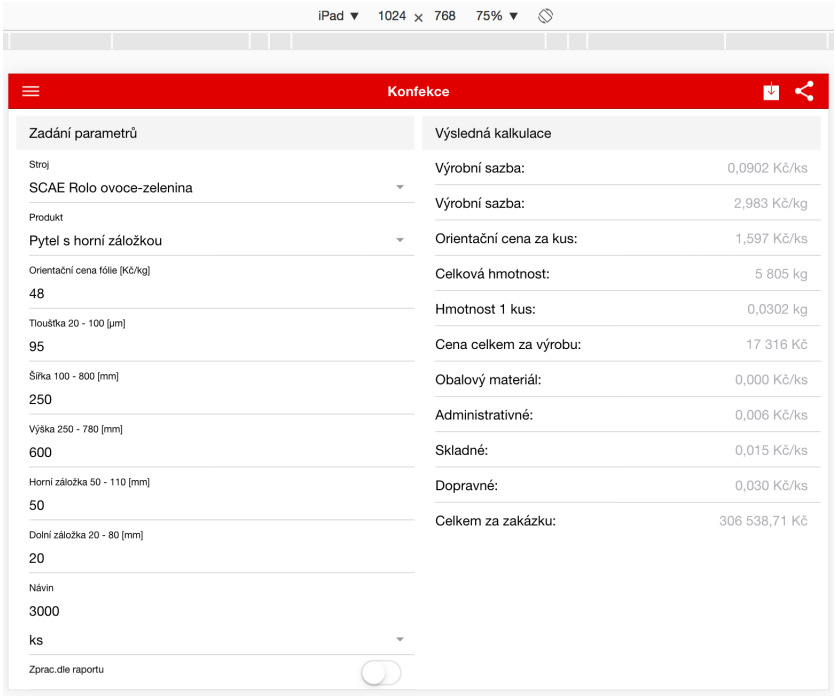
Obrázek 9.2: Přikazy pro výpis do ladící konzole.

9.3.2 Ladící nástroje v prohlížeči

Ladící nástroje v prohlížeči můžeme použít díky tomu, že framework Cordova využívá webovou aplikaci, která je spuštěna ve *WebView*. Emulaci aplikace v prohlížeči Google Chrome můžeme vidět na obrázku 9.3. Velká výhoda tohoto testování je, že webový prohlížeč se nachází prakticky na každém PC a poskytuje hodně možností. Můžeme se snadno dívat na obsah jednotlivých elementů, přidávat zarážky pro debuggování JavaScriptového kódu nebo sledovat síťovou komunikaci, pokud naše aplikace komunikuje přes síť. Jedna z hlavních výhod, kterou jsem pro vývoj používal, je možnost emulace konkrétního zařízení.

V Google Chrome si můžeme navolit, aby se prohlížeč přeměnil na libovolné mobilní zařízení. Tato transformace není dokonalá, ale pro základní testování je naprosto dostačující a šetří čas, protože změny, které provedeme v kódu můžeme obratem vidět v prohlížeči. Snadno takto můžeme zjistit jestli zvolené rozložení jednotlivých elementů je vhodné nebo nikoliv. Na obrázku 9.3 můžeme vidět emulaci tabletu Apple iPad. V Google Chrome jsou

předdefinována některá zařízení jako jsou telefony iPhone od společnosti Apple nebo vlajkové lodě s operačním systémem Android. Pokud bychom chtěli otestovat nějaký specifický model, můžeme si nakonfigurovat svoje vlastní zařízení. Stačí znát požadované rozlišení displeje, řetězec user agenta a device pixel ratio, což je poměr mezi hardwarovým rozlišením displeje a rozlišením, které skutečné zařízení zobrazuje neboli takzvaný pixelový poměr.



Obrázek 9.3: Emulace zařízení Apple iPad v prohlížeči Google Chrome.

9.3.3 Emulátor

Testování mobilní verze aplikace v prohlížeči je velice rychlé a pohodlné, ale je dostačující jen pro základní testování. Je to způsobeno tím, že testujeme samotnou webovou aplikaci, ale již nemůžeme otestovat komunikaci naší webové aplikace s nativními částmi. Pokud chceme například otestovat, jak se aplikace chová při zobrazení SW klávesnice na displeji zařízení, je již nutné využít emulátor nebo nahrát aplikaci do reálného zařízení. Vzhledem k tomu, že se jedná o multiplatformní vývoj, je potřeba testovat jak na emulátoru pro operační systém iOS, tak pro Android. Toto má drobné omezení, protože pokud chceme testovat simulátor pro iOS, je nutné pracovat na operačním systému Mac OS X od společnosti Apple. Spuštění emulátoru pro

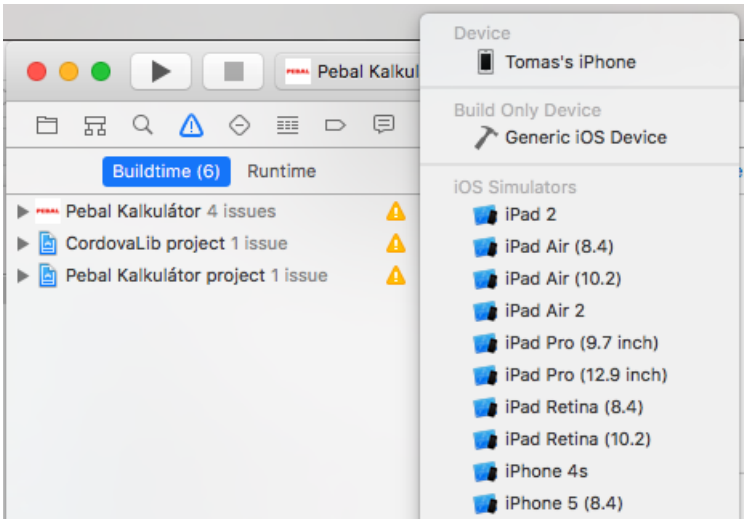
Android je možné na více platformách, můžeme využít jak Mac OS X tak i Windows.

Práce s emulátorem pro platformu iOS je velice příjemná a rychlá. Pro otevření aplikace v emulátoru iOS můžeme použít více cest. Aplikaci můžeme otevřít přímo z terminálu Macu pomocí následujícího příkazu s využitím CLI Ionic:

ionic emulate ios –target=”iPhone-6s”-lc

Kde říkáme, že chceme zobrazit naší aplikaci v emulátoru iOS emulující zařízení Apple iPhone 6S.

Další možností je využít vývojové prostředí Xcode od společnosti Apple a využít tak grafického rozhraní tohoto prostředí k zobrazení simulátoru [34]. Xcode můžeme vidět na obrázku 9.4 a před spuštěním si musíme vybrat v jakém zařízení chceme emulovat současně s verzí iOS. Pokud máme k PC připojeno i reálné zařízení v tomto seznamu se nám objeví a můžeme si zvolit i to. Po zvolení zařízení stačí kliknout na tlačítko s ikonkou *Play* a emulátor se načte.



Obrázek 9.4: Vývojové prostředí Xcode od společnosti Apple.

9.3.4 Zařízení

Emulátor je reálnému zařízení velice blízko a testování v emulátoru odhalí většinu případných chyb, ale otestování aplikace v reálném zařízení nahradit nemůže.

Android zařízení

Pokud chceme aplikaci nahrávat do telefonu přes USB kabel připojený k PC, je pro první spuštění potřeba provést několik nastavení, pokud v zařízení již nejsou aktivní. Toto nastavení se jmenuje režim ladění USB. Ladění USB je nutné aktivovat k tomu, aby PC viděl toto zařízení jako vývojářské a mohl do telefonu nainstalovat naši aplikaci. V defaultním nastavení telefonu režim ladění USB aktivní není, proto je potřeba ho před první instalací aplikace do telefonu přes USB aktivovat. Pokud máme Android 4.4 a novější je potřeba nejdříve aktivovat režim pro vývojáře, protože ten je v defaultním nastavení schován. Vývojářský režim aktivujeme tak, že v nastavení telefonu klikneme 7x na položku *číslo sestavení*. Poté se nám zobrazí nová volba *Ladění USB*. Na tuto volbu klikneme a pak najdeme volbu *Povolit režim ladění s připojeným zařízením USB*. Po tomto kroku je již možné nainstalovat aplikaci přímo z PC pomocí USB [35]. Instalaci provedeme pomocí Ionic CLI a to následujícím příkazem v terminálu:

ionic run android

Druhou možností, jak dostat naši aplikaci do zařízení, je vygenerovat soubor APK a ten si do zařízení přesunout a poté ho ručně v daném zařízení nainstalovat. Tato možnost je časově náročnější než první zmíněná, ale hodí se například v tom případě pokud chceme poslat aplikaci nějaké třetí osobě, aby ji otestoval.

Toto otestování aplikace a nahrání aplikace do zařízení je zdarma a není nutné registrovat se do Google Play.

Poslední možností, jak aplikaci otestovat v reálném zařízení, je registrovat se do Google Play a zaplatit jednorázový poplatek 25$, který budeme muset uhradit, když budeme chtít aplikaci distribuovat. Konzole Google Play nám umožňuje nahrát aplikaci a následně ji uvolnit pro testování. Google play umožňuje zpřístupnit aplikaci otevřenému či uzavřenému testování.

iOS zařízení

Pokud chceme instalovat aplikaci do zařízení s operačním systémem iOS, je potřeba mít již zakoupenou vývojářskou licenci od společnosti Apple, která je vydávána oproti Androidu pouze na jeden rok a jeho cena je 2799 Kč. Tato licence nám postačí i pro distribuci aplikace v Apple Store. Cena je ve srovnání s platformou Android mnohonásobně vyšší. Pokud tedy máme uhrazen tento poplatek, je potřeba si na webové stránce vygenerovat certifikát typu *iOS Development*, který si následně nainstalujeme do operačního systému Mac OS X. Pokud tedy máme tento certifikát, můžeme nainstalovat aplikaci přes USB kabel do zařízení iOS. To můžeme opět provést dvěma způsoby, zaprvé pomocí Ionic CLI pomocí následujícího příkazu v terminálu:

ionic run ios

Nebo můžeme využít vývojového prostředí Xcode. Postup je prakticky stejný jako při spouštění iOS emulátoru. Pouze je nutné jako cílové zařízení vybrat naše připojené zařízení přes USB kabel jak je vidět na obrázku 9.4.

Poslední možností jak otestovat aplikaci v reálném zařízení s iOS je využít TestFlight. TestFlight je online servis, který je nám zpřístupněn po zaplacení vývojářské poplatku a je to sofistikovaný způsob, jak testovat aplikaci nejen na svém zařízení, ale i na zařízení ostatních uživatelů, který vlastní zařízení s iOS. Pokud chceme tento online servis využít, je nutné aplikaci pomocí Xcode nahrát do iTunes Connect, které následně slouží i pro distribuci aplikací viz níže. Poté co se nám podaří aplikaci nahrát do iTunes Connect, přepneme se do sekce TestFlight a je potřeba přidat uživatele, který budou mít možnost naši aplikaci testovat.

První skupina je ta, která má také uhrazen roční vývojářsky poplatek takzvaní *iTunes Connect Users* a druhá skupina jsou externí testeři neboli *External Testers* a tímto testerem se může stát kdokoliv, kdo má založený účet u Apple neboli má své vlastní Apple ID (Apple ID je e-mail, který je spojen se založením Apple účtu). Mezi těmito skupinami je jeden zásadní rozdíl, a to že *iTunes Connect Users* se odešle aplikace prakticky okamžitě, externím uživatelům nikoliv. Pokud chceme využívat externí testery je nutné, aby aplikaci nejdříve schválil zaměstnanec Apple, což může trvat i několik dní, ale zpravidla je to do druhé dne schváleno. Poté co odešleme aplikaci k testování, uživatelům přijde e-mail s informací, že byla uvolněna nová verze pro testování a můžou si ji stáhnout v aplikaci TestFlight, kterou je nutné si stáhnout. Pokud ji mají již nainstalovanou ve svém telefonu při odeslání

aplikace k testování, přijde jim rovnou notifikace s informací, že byla vydána nová verze pro testování. V iTunes Connect poté můžeme sledovat, zda si naši nově uvolněnou testovací verzi daný uživatel již stáhl do svého zařízení.

9.3.5 Testovací scénáře

Testování mobilní verze aplikace bylo rozděleno na 4 základní části. Aplikace byla postupně testována v prohlížeči, na emulátoru a naposled v reálném zařízení.

Přihlášení

První testovací scénář simuloval přihlášení uživatele, kde bylo otestováno přihlášení uživatele s neexistujícím emailem, špatným heslem, blokovaného uživatele, úspěšné přihlášení a odhlášení uživatele.

Kalkulace

Druhý testovací scénář se zabýval kalkulací. Do formuláře byly zadávány různé hodnoty, které byly mimo rozsah stroje, hodnoty, které nebyly číselné (například místo šířky 10 mm byla zadána hodnota XYZ). Dále bylo proběhlo vygenerování PDF souboru s výslednou kalkulací a odeslání vygenerovaného PDF do emailu.

Offline práce

Další část testování byla zaměřena na offline práci v aplikaci, pokud je uživatel v aplikaci přihlášen. Byla vytvořena kalkulace, která byla poté v offline režimu uložena a po připojení zařízení k internetu byla kalkulace opět načtena i se zobrazením výsledků kalkulace.

GUI

Poslední částí testování bylo ověření uživatelské rozhraní na různých zařízeních. K tomuto testování byl využit převážně emulátor z důvodu simulace na různých zařízeních. Cílem tohoto testu bylo ověřit zda se aplikace zobrazuje správně na mobilním telefonu, ale i na tabletu.

9.3.6 Vyhodnocení provedených testů

Výše uvedené testy proběhly v pořádku. Při neúspěšném přihlášení aplikace se vždy vrátila odpovídající hláška, na kterou mohl uživatel patřičně zareagovat. Při zadávání špatných hodnot do kalkulace, formulář v daném poli zčervenal, takže uživatel mohl hodnotu opravit. Uložení kalkulace a opětovné načtení v režimu offline také proběhlo v pořádku a při testech GUI se aplikace zobrazila správně, jak na mobilních zařízeních, tak na tabletech. Všechny testy tedy proběhly úspěšně.

9.4 Testování v reálném provozu

Vzhledem k tomu, že na diplomové práci spolupracuji se společností Pebal s.r.o. již od konce roku 2015, byla aplikace testována v průběhu zaměstnanci Pebalu. Na testování se podílelo přibližně 25 zaměstnanců Pebalu a na základě jejich zpětné vazby byly dolaďovány drobné úpravy. Od začátku roku 2017 byla aplikace nasazena do ostrého provozu a obchodní zástupci ji využívají k nacenění produktů.

10 Distribuce mobilní verze aplikace

Aby si uživatelé mohli aplikaci pohodlně nainstalovat do svého zařízení, je potřeba ji umístit do obchodu dané platformy. Vzhledem k tomu, že se jedná o firemní aplikaci, jsou tu rozšířené možnosti, jak aplikaci distribuovat, aby ji neměl možnost si stáhnout každý, ale má to svá omezení.

10.1 Google Play

U mobilní platformy Android bychom mohli použít i instalaci pouze pomocí vyexportovaného souboru APK. Pokud bychom chtěli obchodnímu zástupci umožnit instalaci aplikace, stačilo by mu pouze zaslat daný soubor na e-mail, zaslat link, kde si aplikaci může stáhnout nebo mu nahrát naši aplikaci na SD kartu a poté by byl schopen si aplikaci do svého zařízení nainstalovat. Tato možnost ale není úplně ideální v případě, kdy chceme aplikaci dostat mezi nějaké větší množství zaměstnanců a nebo chceme vydat aktualizaci dané aplikace, protože musíme celý proces rozšíření mezi uživatele absolvovat znovu a Google Play nám pomůže udělat spoustu práce za nás.

10.1.1 Nahrání aplikace do obchodu Google Play

Druhou možností, jak aplikaci mezi jednotlivé zaměstnance rozšířit, je tedy využít obchodu Google Play. Abychom mohli Google Play využít, je nejprve nutné uhradit registrační poplatek ve výši $25 v přepočtu 625 Kč, který je jednorázový. Pokud máme poplatek uhrazený, můžeme se přihlásit do Google Play Console na adrese: *https://play.google.com/apps/publish* a vytvořit novou aplikaci, kam je nutné nahrát vytvořený APK soubor s naší aplikací. Soubor APK vytvoříme pomocí Ionic CLI a příkazu níže:

ionic build android –prod –release

Flag *–prod* není povinný, ale je velice důležitý, protože dojde k minimalizaci JavaScriptových souborů a celkovému zrychlení chodu aplikace, především při jejím startu, takže start aplikace se obratem zrychlí o několik sekund oproti vývojové verzi aplikace. Zrychlení aplikace je na úkor rychlosti sesta-

vení, ale to je jednorázový proces. Při běžném testování je proto lepší flag *–prod* nevyužívat, protože bychom čekali déle na sestavení. Pokud máme APK soubor vygenerovaný, je nutné ho podepsat naším privátním klíčem, který je důležitý pro vydání dalších verzí aplikace. Pokud budeme chtít vydat novou verzi naší aplikace, musíme vygenerovaný APK soubor podepsat stejným privátním klíčem jako předtím, jinak by nám v Google Play nebylo umožněno aktualizovaný soubor nahrát a ztratili bychom možnost aktualizace aplikace. Poté můžeme aplikaci nahrát do Google Play a vyplnit potřebné informace o aplikaci. Pokud máme vše v Google Play připraveno, můžeme aplikaci zveřejnit klasickým způsobem, kdy bude aplikace zpřístupněna široké veřejnosti nebo využít možnosti omezeného publika.

10.1.2 Uzavřené testování

První možností, jak aplikaci šířit přes Google Play pro omezené publikum, je vydání aplikace v alfa verzi pro uzavřené publikum. Tímto docílíme výhod využití Google Play, ale zároveň poskytneme aplikaci jen vybraným uživatelům, nad kterými máme kontrolu a není nutné využití firemního účtu v Google Play, který si popíšeme níže.

10.1.3 Soukromá aplikace (Private Channel)

Druhou možností, jak šířit aplikaci, je využití takzvaného *Google Play Private Channel*. Tato možnost, ale není jen o zaškrtnutí checkboxu v Google Play, ale je potřeba vytvořit speciální vývojářsky účet, který bude vázán na konkrétní organizaci. Není tedy možné v jednom účtu distribuovat veřejné aplikace a zároveň soukromé aplikace nějaké organizace [36]. Účet je vázán na organizaci a ne na doménu, takže v případě, že organizace má více domén například mojedomena.cz a mojedomena.com tuto soukromou aplikaci budou moci stahovat jak uživatelé s uživatelským jménem honza@mojedomena.cz, tak john@mojedomena.com. U tohoto modelu distribuce narazíme v tom případě, pokud někteří uživatelé používají pro firemní účely i své vlastní zařízení, kde jsou přihlášeni pod svým osobním účtem. V tomto případě to lze, ale obejít využitím soukromé aplikace s kombinací s uzavřeným testováním pro uživatele, kteří využívají svá zařízení.

10.2 Apple Store

Pro distribuci aplikací na zařízení s iOS, není možné využít šíření pomocí APK souboru jako u Androidu. U iOS, pokud chceme nahrát aplikaci do zařízení, musíme přes USB kabel s využitím vývojového prostředí Xcode nebo využít online servisu iTunes Connect. iTunes Connect je součástí vývojářské licence od Apple v ceně 2799 Kč na rok. Licence od Apple je tedy několikanásobně dražší než licence u Androidu a navíc je pouze na jeden rok. Šíření aplikace přes USB kabel není v reálném provozu možné z časových důvodů, takže je nutné využít iTunes Connect. Zde máme opět více možností, jak aplikaci k jednotlivým uživatelům dostat. Nejdříve je ale nutné samotnou aplikaci do iTunes Connect nahrát.

10.2.1 Nahrání aplikace do iTunes Connect

Pokud máme zakoupenou vývojářskou licenci od Apple a nainstalovaný platný certifikát typu *iOS Distribution*, můžeme si v iTunes Connect vytvořit novou aplikaci. Poté co je prostředí v iTunes Connect připravené, je potřeba sestavit produkční verzi aplikace pomocí následujícího příkazu:

ionic build ios –prod –release

Flag *–prod* má stejný význam jako u platformy Android a dojde tedy opět k zrychlení aplikace a je tedy dobrý tento flag využít. Poté, co je aplikace sestavena, otevřeme si ve vývojovém prostředí Xcode soubor s koncovkou *.xcodeproj*, který se vygeneruje v adresáři *platforms/ios/*. V Xcode si vygenerujeme takzvaný archiv, který následně můžeme nahrát do iTunes Connect. Zde máme opět více možností, jak aplikaci rozšířit. Jednak klasickým způsobem mezi veřejné uživatele nebo pro soukromou skupinu uživatelů.

10.2.2 Uzavřené testování TestFlight

První možností uzavřené skupiny je online servis TestFlight určený pro testování mobilních aplikací iOS, který je v ceně vývojářské licence od Applu a je součástí iTunes Connect. TestFlight je převážně k testování aplikace, ale můžeme ho využít i pro distribuci, pokud nechceme využívat *Volume Purchase Program for Business* viz níže. Je tu opět výhoda, že uživatel, kte-

rému chceme aplikaci zpřístupnit nemusí mít pouze firemní zařízení, ale i své osobní. Toto samozřejmě pro nějaké firmy může být nežádoucí, pro jiné výhodou.

10.2.3 Soukromá aplikace (VPP)

Volume Purchase Program for Business neboli program hromadných nákupů je program, který firmám, školám nebo institucím usnadňuje vydání nebo nákup aplikací v AppStoru ze zařízeních dané instituce, který dorazil do České Republiky v roce 2016 [37]. Aby bylo tedy možné v iTunes Connect zpřístupnit aplikaci nějaké organizaci, je potřeba, aby daná organizace byla registrována v programu VPP. Tento program je opět vázán na organizaci a daná organizace jednotlivým uživatelům vytváří účty se stejnou doménou například *uzivatel1@mojedomena.cz*.

Uživatel musí být na zařízení přihlášen pod svým účtem v organizaci a nemá možnost nainstalovat si aplikaci na své vlastní zařízení. To má opět své výhody a nevýhody. Pokud v organizaci skončí, jeho účet může být zablokován a on již nebude mít přístup k těmto firemním aplikacím. Pokud mu firma dává přístup k nějakým placeným aplikacím, je toto určitě velká výhoda, protože pokud by nechala tyto placené aplikace instalovat na osobní zařízení s osobním Apple ID, po ukončení pracovního poměru daného zaměstnance v organizaci, by mu aplikace zůstala a novému zaměstnanci by musela zakoupit opět novou licenci. Nevýhodou je, pokud uživatel používá pro práci osobní zařízení a v tomto případě není možné program VPP využít, protože aplikace zveřejněné v tomto programu, je možné instalovat pouze do zařízeních, kde je uživatel přihlášen pod Apple ID dané organizace.

Spolu s tímto programem je možné využít i *Mobile Device Management MDM*, který nám dává možnost určit, jaká aplikace bude zpřístupněna jakému účtu. To se může hodit například tehdy, pokud chceme danou aplikaci zpřístupnit jen určitému oddělení v organizaci nebo pokud je nějaká aplikace placená a tudíž zakoupíme jen licence pro ty, kteří ji skutečně budou využívat a následně jim díky MDM dáme k této aplikaci přístup.

Pokud tedy chceme naši aplikaci zpřístupnit v programu VPP v iTunes Connect v sekci *Pricing and Availability*, zvolíme v části *Volume Purchase Program* možnost ***Available privately as a custom B2B app***. Kde je nutné, ihned po zvolení této možnosti, zadat Apple ID dané organizace, jinak není možné záznam uložit.

11 Návrh dalších rozšíření

Jak bylo uvedeno výše, celková výroba společnosti Pebal se skládá z extruze, tisku a konfekce. Diplomová práce se převážně věnovala konfekci z důvodu největší priority pro firmu Pebal. Dalším rozšířením této aplikace tedy bude rozšíření a realizace aplikace i pro tisk a extruzi.

Další rozšíření by se mohlo týkat využití přihlášení pomocí otisku prstů. Pro iOS je tedy možné využít Touch ID a pro Android Fingerprint Authentication. U tohoto přihlášení bude nutné, aby se uživatel poprvé přihlásil do systému pomocí klasických přihlašovacích údajů, které by se následně uložily do zařízení a v příštím přihlášení se bude ověřovat pouze otisk prstu a aplikace provede přihlášení do systému pomocí uložených přihlašovacích údajů.

Dalším z rozšíření může být překlad aplikace do více jazyků v případě potřeby. Aplikace je již připravená, takže je možné aplikaci přeložit bez větších zásahů do samotného kódu.

Dále by bylo možné mobilní aplikaci rozšířit i o fotografie jednotlivých produktů. Pokud si obchodní zástupce zvolí v kalkulátoru nějaký produkt, bylo by možné zobrazit k produktu nějakou galerii, aby bylo možné produkt ukázat zákazníkovi.

12 Závěr

Cílem práce bylo navrhnout a implementovat aplikaci pro mobilní přístup k informačnímu systému K2, kterou budou primárně využívat obchodní zástupci společnosti Pebal s.r.o..

V rámci této práce byl prozkoumám informační systém K2 ve společnosti Pebal s.r.o. včetně prozkoumání výrobních procesů v této společnosti. Byly analyzovány možnosti přístupu k tomuto informačnímu systému a také bylo analyzováno současné a požadované řešení pro kalkulaci ceny vyráběných produktů.

Po analýze proběhl návrh vytvářené aplikace s návrhem databázového modelu pro ukládání dat a byly vybrány vhodné technologie. Pro serverovou část systému byl zvolen framework Nette a pro multiplatformní mobilní aplikaci byl zvolen framewrok Cordova dohromady s frameworkem Ionic.

Navržený systém byl implementován ve zvolených technologiích s důrazem na celkové zabezpečení aplikace a offline práci v aplikaci. Při implementaci serverové části, kde probíhají výpočty, byl vyřešen problém s plovoucí řadovou částkou. A také bylo navrženo další rozšíření aplikace.

V serverové části aplikace byla ověřena funkcionalita pomocí unit testů a mobilní verze aplikace byla testována v emulátoru a následně i v reálných zařízeních. Nakonec byla navržená aplikace otestována v reálném provozu, kde ji již po dobu půl roku využívá přibližně 25 obchodních zástupců společnosti.

Vzhledem k tomu, že se jedná o firemní aplikaci, která by neměla být dostupná široké veřejnosti, byla prozkoumána distribuce mobilní aplikace pro organizace a firmy v obchodech Google Play a Apple Store.

Cíle práce byly splněny a obchodní zástupci společnosti Pebal mohou naplno využívat vytvořenou aplikaci, která jim usnadňuje práci v reálném provozu.

Seznam obrázků

Přehled použitých zkratek a značení

API	Application Programming Interface. Rozhraní pro vývoj aplikací.
APK	Formát souboru používaný pro instalaci software na operační systém Android.
CLI	Command Line Utility. Nástroj pro správu přes příkazovou řádku.
CSS (CSS3)	Cascading Style Sheets. Kaskádové styly jsou jazyk pro popis zobrazení webových stánek.
GUI	Graphical User Interface. Uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků.
HTML	HyperText Markup Language. Jazyk pro tvorbu webových stránek.
HTTP	Hypertext Transfer Protocol. Protokol určený pro výměnu hypertextových dokumentů.
HTTPS	Hypertext Transfer Protocol Secure. Protokol umožňující zabezpečenou komunikaci v počítačové síti.
HW	Hardware. Fyzicky existující technické vybavení nějakého zařízení.
IDE	Integrated Development Environment. Vývojové prostředí. Software usnadňující práci programátorů.

iOS	je mobilní operační systém vytvořený společností Apple Inc.
IP	Internet Protocol. Základní protokol pracující na síťové vrstvě.
JS	JavaScript. Multiplatformní, objektově orientovaný skriptovací jazyk.
JSON	JavaScript Object Notation. Formát určený pro přenos dat nezávislý na počítačové platformě ve formě řetězce.
Mac OS	Macintosh Operating System. Operační systém pro počítače od firmy Apple.
OS	Operační systém. Základní programové vybavení počítače.
PDF	Portable Document Format. Souborový formát pro ukládání dokumentů nezávisle na platformě.
PHP	Hypertext Preprocessor. Skriptovací jazyk.
SSL	Secure Sockets Layer. Vrstva zabezpečených socketů.
SW	Software. Programové vybavení počítače.
TLS	Transport Layer Security. Kryptografický protokol poskytující možnost zabezpečené komunikace. A následovník SSL.
USB	Universal Serial Bus. Způsob připojení periferií k počítači.

Literatura

[1] Blažíček Roman a Basl Josef *Podnikové informační systémy* Grada Publishing, a.s., 2012. ISBN 978-80-247-4307-3.

[2] *K2 atmitec* [online]. [cit. 14.4.2017]. Dostupné z: `http://www.k2.cz/cs/pomahame-ridit-stovky-uspesnych-firem`.

[3] *Programování, implementace a uživatelské nastavení Informačního systému K2 - Bakalářská práce* [online]. [cit. 13.5.2017]. Dostupné z: `https://is.bivs.cz/th/13208/bivs_b/BP_jicha_dusan.pdf`.

[4] *Žebříček nejstahovanějších aplikací Apple Store za 2016* [online]. [cit. 15.4.2017]. Dostupné z: `http://www.businessinsider.com/top-free-apps-of-2016-according-to-apple-2016-12`.

[5] *Prodej Instagramu do rukou společnosti Facebook* [online]. [cit. 15.4.2017]. Dostupné z: `http://www.businessinsider.com/instagram-zuckerbergs-biggest-win-so-far-2016-1`.

[6] *React pro vývoj multiplatformních aplikací společnosti Facebook Inc* [online]. [cit. 15.4.2017]. Dostupné z: `https://facebook.github.io/react/`.

[7] *Aplikace vyvíjené multiplatformním nástrojem React Native* [online]. [cit. 15.4.2017]. Dostupné z: `https://facebook.github.io/react-native/showcase.html`.

[8] *Multiplatformní nástroj Xamarin od firmy Microsoft* [online]. [cit. 15.4.2017]. Dostupné z: `https://www.xamarin.com`.

[9] *GitHub React Native* [online]. [cit. 15.4.2017]. Dostupné z: `https://github.com/facebook/react-native`.

[10] *Dokumentace Apache Cordova* [online]. [cit. 15.4.2017]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview.

[11] *Ionic Framework* [online]. [cit. 15.4.2017]. Dostupné z: http://ionicframework.com.

[12] *Dokumentace Angular* [online]. [cit. 15.4.2017]. Dostupné z: https://angular.io/docs/ts/latest/.

[13] *Framework Nette* [online]. [cit. 15.4.2017]. Dostupné z: https://nette.org.

[14] *Projekty postaevné na frameworku Nette* [online]. [cit. 15.4.2017]. Dostupné z: https://builtwith.nette.org.

[15] *Nette zabezpečení před zranitelnostmi* [online]. [cit. 15.4.2017]. Dostupné z: https://doc.nette.org/cs/2.4/vulnerability-protection.

[16] *Nezávislý test PHP frameworků* [online]. [cit. 15.4.2017]. Dostupné z: https://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/.

[17] *Licence Nette* [online]. [cit. 13.5.2017]. Dostupné z: https://nette.org/cs/license.

[18] *Dokumentace frameworku Ionic* [online]. [cit. 15.4.2017]. Dostupné z: http://ionicframework.com/docs/v1/overview/#license.

[19] *Cordova Apache licence* [online]. [cit. 13.5.2017]. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0.

[20] *Vyšetřování zabití 28 vojáků rakotovým systémem MIM-104* [online]. [cit. 1.5.2017]. Dostupné z: http://www.gao.gov/assets/220/215614.pdf.

[21] *Dokumentace šablonovacího systému Latte* [online]. [cit. 6.5.2017]. Dostupné z: https://latte.nette.org/cs/.

[22] *Dokumentace pre-processoru Less* [online]. [cit. 6.5.2017]. Dostupné z: http://lesscss.org.

[23] *Dokumentace mPDF knihovny* [online]. [cit. 5.5.2017]. Dostupné z: https://mpdf.github.io.

[24] Wargo John M *Apache Cordova 4 Programming* Pearson Addison Wesley Prof, 2015. ISBN 978-01-340-4819-2.

[25] Yakov Fain a Anton Moiseev *Angular 2 Development with Typescript* Manning Publications, 2016. ISBN 978-1617293122.

[26] *Dokumentace komponent frameworku Ionic 2* [online]. [cit. 6.5.2017]. Dostupné z: https://ionicframework.com/docs/components/#overview.

[27] *Dokumentace SASS - rozšíření CSS* [online]. [cit. 6.5.2017]. Dostupné z: http://sass-lang.com/documentation/file.SASS_REFERENCE.html.

[28] *Dokumentace Angular 2 - třída FormGroup* [online]. [cit. 6.5.2017]. Dostupné z: https://angular.io/docs/ts/latest/api/forms/index/FormGroup-class.html.

[29] *Dokumentace Nette - práce s hesly* [online]. [cit. 13.4.2017]. Dostupné z: https://doc.nette.org/cs/2.4/passwords.

[30] *Co je HTTPS - Comodo certifikační autorita* [online]. [cit. 13.5.2017]. Dostupné z: https://www.instantssl.com/ssl-certificate-products/https.html.

[31] *Druhy certifikátů* [online]. [cit. 13.4.2017]. Dostupné z: https://www.globalsign.com/en/ssl-information-center/types-of-ssl-certificate/.

[3] *Symantec chyboval s EV certifikáty, prohlížeče jim přestanou důvěřovat* [online]. [cit. 9.4.2017]. Dostupné z: https://www.root.cz/clanky/symantec-chyboval-s-ev-certifikaty-prohlizece-jim-prestanou-duverovat.

[32] *Úložiště frameworku Cordova* [online]. [cit. 13.5.2017]. Dostupné z: https://cordova.apache.org/docs/en/latest/cordova/storage/storage.html.

[33] *Dokumentace Nette Tester pro použití Unit testů* [online]. [cit. 16.4.2017]. Dostupné z: https://tester.nette.org/cs/.

[34] Matthew Knott *Beginning Xcode* Springer Berlin, 2014. ISBN 978-1430257431.

[35] Nolan Godfrey *Android Best Practices* Springer Berlin, 2013. ISBN 978-1430258575.

[36] *Google Play Private Channel* [online]. [cit. 21.4.2017]. Dostupné z: http://www.techrepublic.com/blog/google-in-the-enterprise/deploy-private-android-apps-on-google-play/.

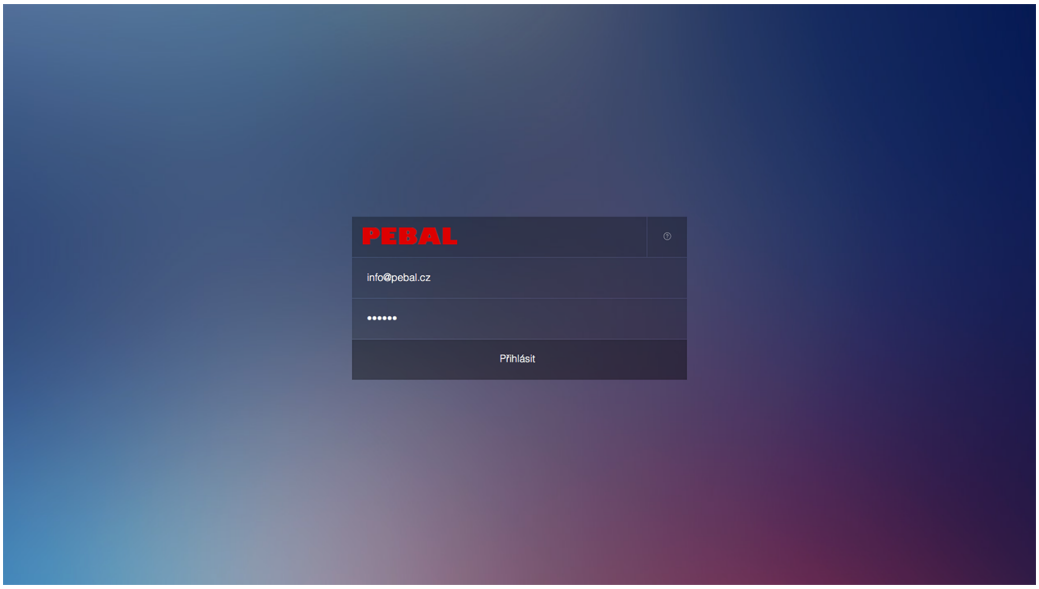
[37] *VPP nákup aplikací pro firmy* [online]. [cit. 24.4.2017]. Dostupné z: http://jankucerik.cz/category/co-delam/business-reseni.

Přílohy

Uživatelská dokumentace - desktopová verze

Přihlášení do aplikace

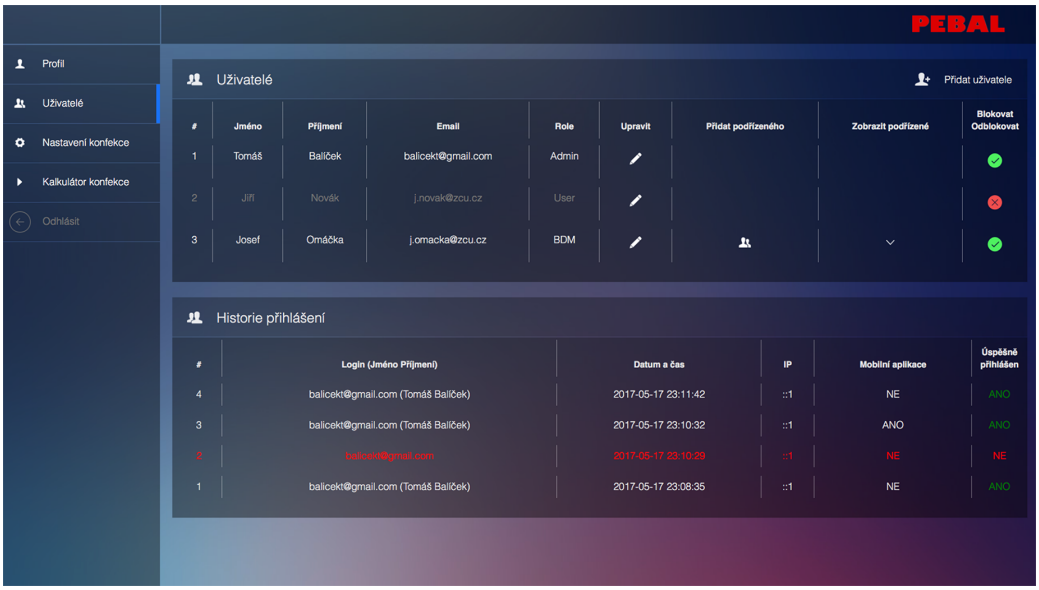
Pro přihlášení do aplikace je potřeba zažádat o přístupové údaje, které přiděluje administrátor systému. Přihlašovací obrazovka je vidět na obrázku 1.



Obrázek 1: Přihlašovací obrazovka desktopové verze systému.

Správa uživatelů

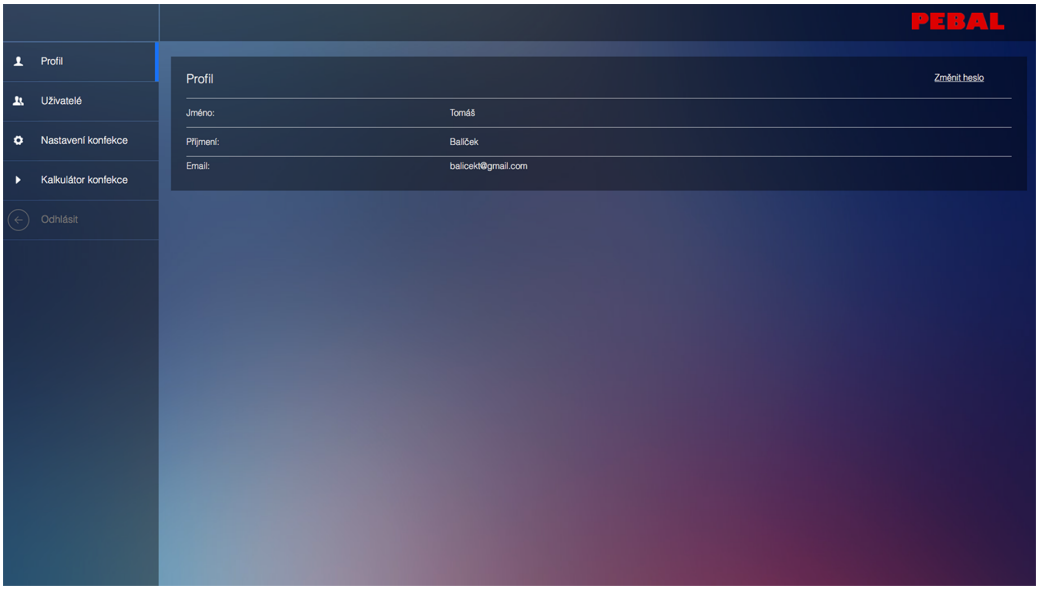
Do záložky uživatelé má přístup pouze uživatel s rolí administrátora. V této záložce je možné vidět historii přihlášení jednotlivých uživatelů včetně jejich IP adresy a jejich neúspěšných pokusů. Dále je zde informace, jestli se uživatel přihlásil z mobilní aplikace nebo z desktopu pomocí prohlížeče. Dále můžeme uživatele přidávat, editovat, přiřazovat jim podřízené a blokovat je. Tuto záložku můžeme vidět na obrázku 2.



Obrázek 2: Přehled uživatelů s historií jejich přihlášení.

Profil uživatele

V záložce *Profil* jsou zobrazeny základní informace přihlášeného uživatele a uživatel si zde může měnit heslo pro přihlášení. Tato obrazovka je vidět na obrázku 3.



Obrázek 3: Profil se základními údaji přihlášeného uživatele.

Nastavení konfekce

Do záložky nastavení konfekce má přístup pouze uživatel s rolí administrátora. Zde je možné nastavit jednotlivé parametry pro kalkulaci. Po kliknutí na název sekce (například *Blokování*) se rozevře tabulka, kde je možné po kliknutí na ikonu tužky položku upravit. Tato obrazovka je vidět na obrázku 4.

Obrázek 4: Nastavení konfekce.

Kalkulátor konfekce

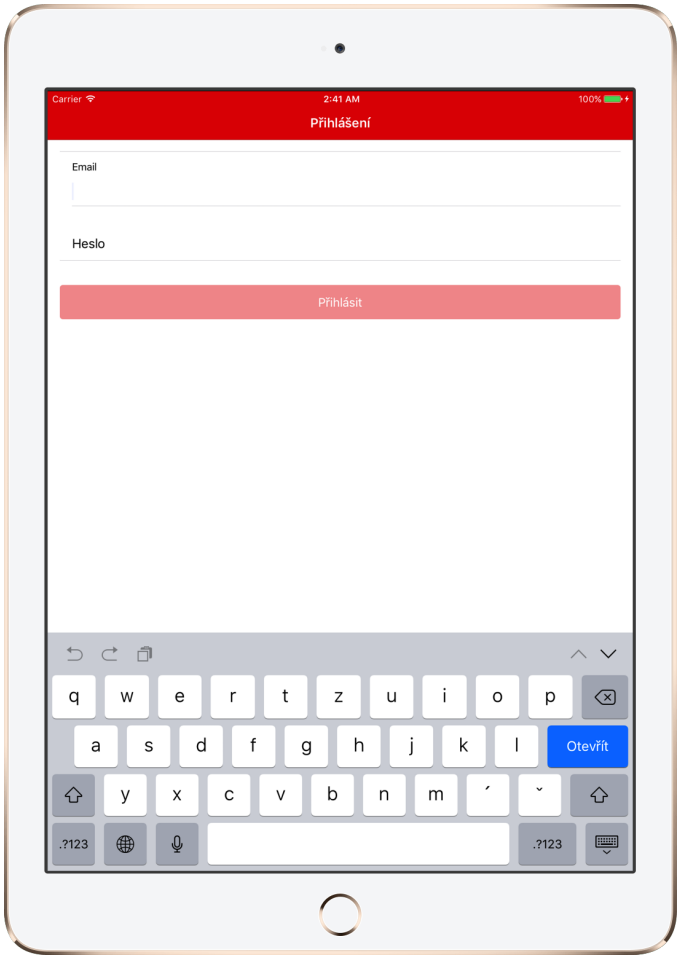
Na obrázku 5 můžeme vidět kalkulátor konfekce, konkrétně pro stroj SCAE. Pokud změníme stroj, pro který chceme kalkulaci vytvořit, dojde k automatickému přegenerování formuláře podle parametrů, které daný stroj potřebuje. Dále je možné vygenerovat PDF soubor s výslednou kalkulací po kliknutí na ikonu tiskárny v pravém horním rohu. Po kliknutí na ikonu dopisu se zobrazí okno, kde uživatel zadá email, na který chce odeslat výslednou kalkulaci ve formátu PDF.

Obrázek 5: Formulář pro výpočet kalkulace.

Uživatelská dokumentace - mobilní verze

Přihlášení do aplikace

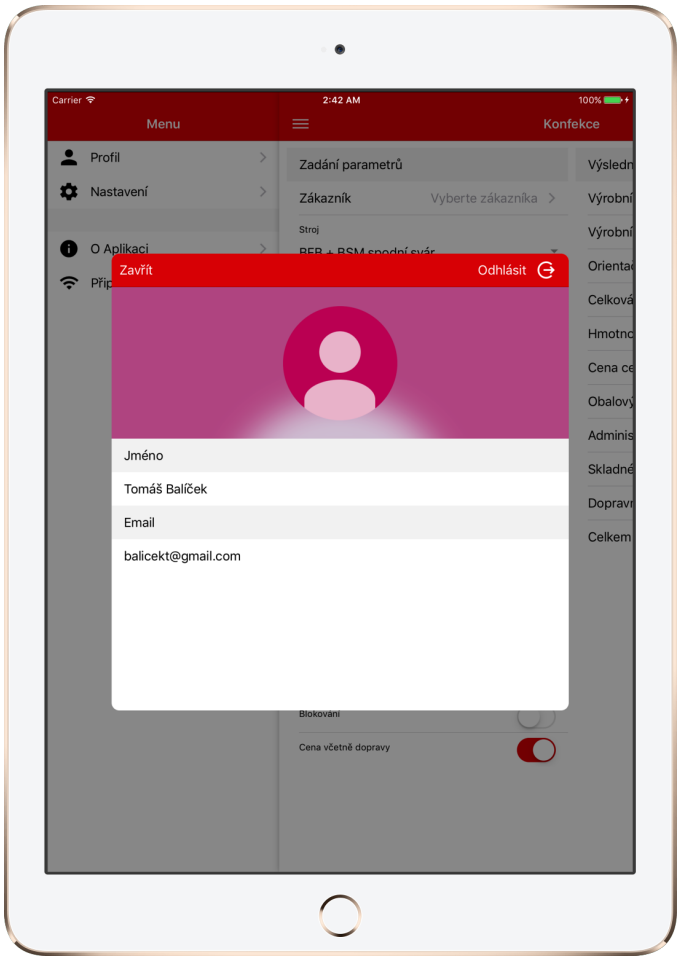
Přihlašovací obrazovku na mobilní verzi aplikace můžeme vidět na obrázku 6.



Obrázek 6: Přihlašovací obrazovka mobilní verze systému na zařízení Apple iPad Air 2.

Profil uživatele

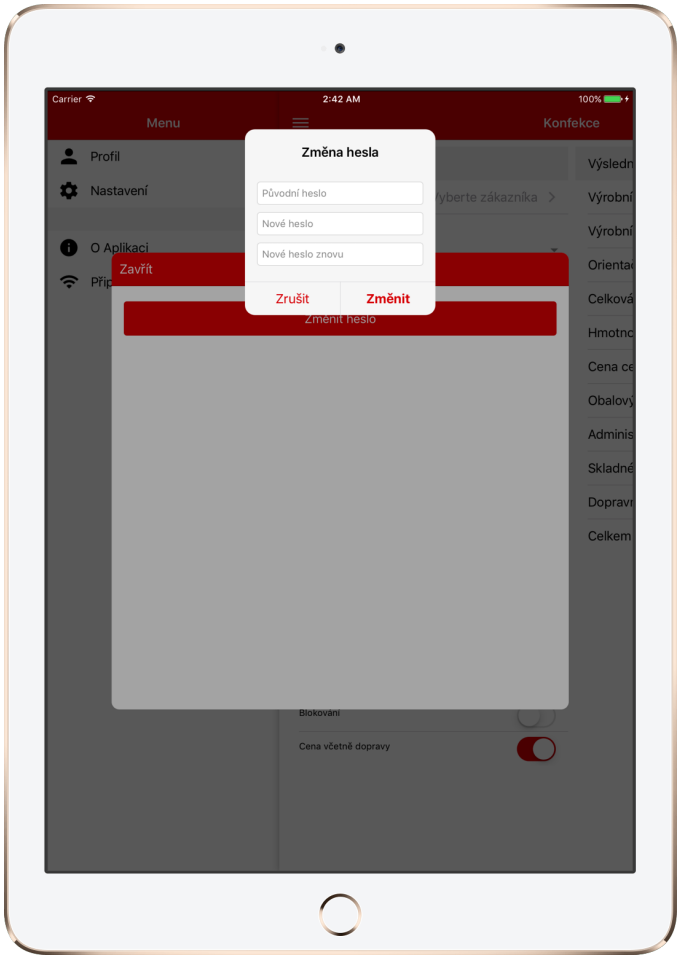
Po přihlášení do aplikace si můžeme zobrazit profil se základními informacemi o přihlášeném uživateli. Na této stránce se také z aplikace můžeme odhlásit. Tato obrazovka je vidět na obrázku 7.



Obrázek 7: Profil se základními údaji přihlášeného uživatele.

Změna hesla

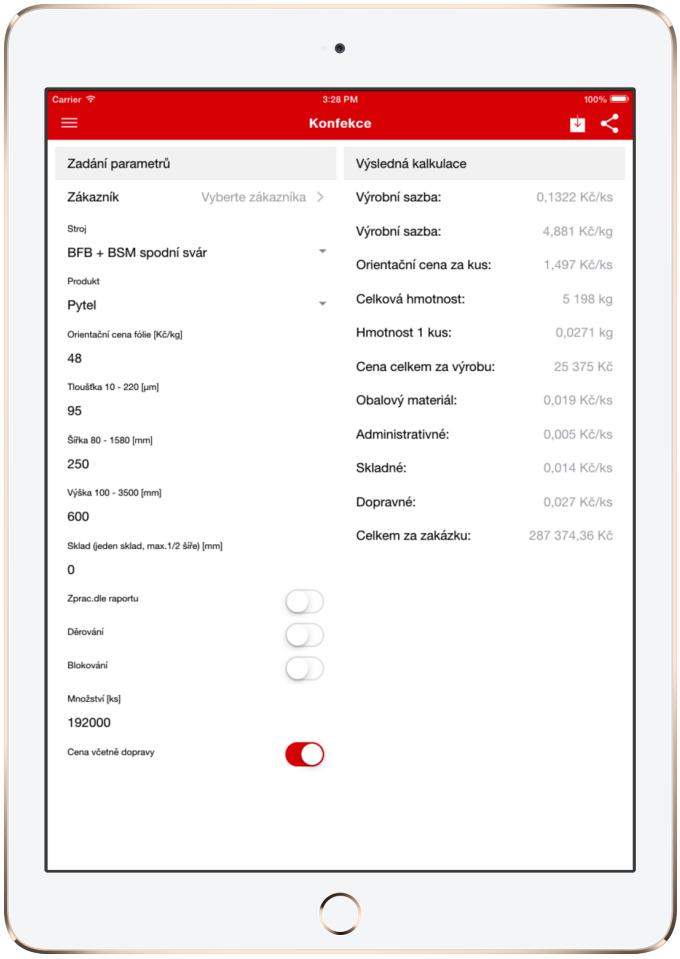
V nastavení aplikace je možné změnit si heslo. Změnu hesla můžeme vidět na obrázku 8. Účet je stejný pro desktopovou i mobilní verzi aplikace, to znamená, že při změně hesla v mobilní aplikaci se změna projeví v obou verzích zároveň.



Obrázek 8: Zobrazení okna se změnou hesla.

Kalkulátor konfekce

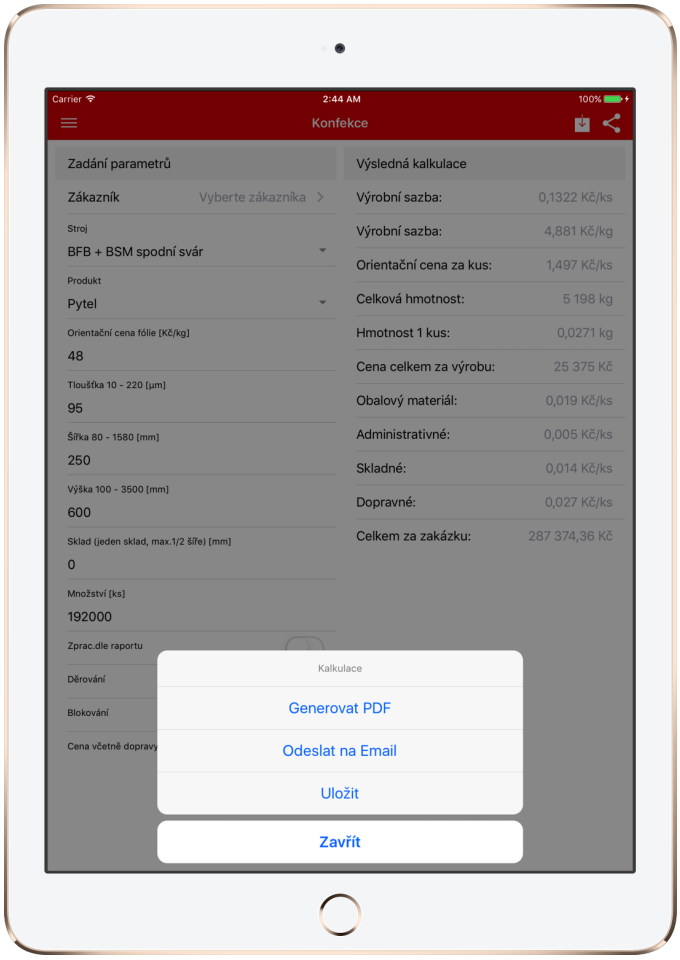
Na obrázku 9 můžeme vidět kalkulátor konfekce, konkrétně pro stroj BFB + BSM. Pokud změníme stroj, pro který chceme kalkulaci vytvořit, dojde k automatickému přegenerování formuláře podle parametrů, které daný stroj potřebuje. Levý sloupec slouží k zadávání parametrů a pravý zobrazuje výsledky kalkulace.



Obrázek 9: Formulář pro výpočet kalkulace.

Kalkulátor konfekce - uložení a export

Pokud chceme vygenerovat PDF soubour s kalkulací, odeslat kalkulaci na email nebo si kalkulaci uložit, stačí kliknout na ikonu *sdílení* v pravém horním rohu aplikace. Zobrazí se nabídka, se které si vybereme požadovanou akci. Pokud chceme načíst uloženou kalkulaci, použijeme druhou ikonku zprava, která nám zobrazí nabídku s kalkulacemi, které jsme si v minulosti uložili (tato funkcionalita je dostupná i v režimu offline).



Obrázek 10: Formulář pro výpočet kalkulace se zobrazeným menu.