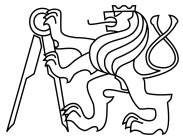
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Platforma pro prezentační vrstvu informačního systému K2
Student:	Bc. Martin Krupka
Vedoucí:	Ing. Tomáš Szkandera
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem diplomové práce je navrhnout univerzální platformu pro definici univerzálních formulářů jednotlivých klientů informačního systému K2. Implementovat a ověřit navržené řešení na mobilním klientu K2 pro systém iOS.

1) Analyzujte požadavky na uživatelské rozhraní klientů IS K2 (plný, mobilní, web klient). Požadavky definují projektoví manažeři oddělení systémové integrace a členové UX týmu K2.
2) Navrhněte řešení pro ukládání definic formulářů (použijte Use Case diagramy, případně ER diagram).
3) Konzultujte navržené řešení s vedoucím diplomové práce.
4) Implementujte univerzální uživatelské rozhraní pro klienta K2 na platformě iOS.
5) Vytvořte ohodnocení nákladů implementace a soupis očekávaných přínosů (klientské i serverové části).

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 9. února 2016

Poděkování

Tímto bych chtěl poděkovat vývojovému týmu informačního systému K2 za podporu při řešení problémů v této diplomové práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 8. května 2016 .....

Odkaz na tuto práci

Krupka, Martin. *Platforma pro prezentační vrstvu informačního systému K2.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Práce se zabývá analýzou a návrhem nové prezentační vrstvy informačního systému K2. Obsahuje porovnání dostupných možností pro prezentační vrstvu programu s ohledem na UX požadavky a použitelnosti navrženého řešení na všech požadovaných platformách. V závěru práce je popsaná implementace základního zobrazení na mobilním systému iOS a zhodnocení nákladů a přínosů celého řešení.

Klíčová slova Prezentační vrstva, IS, K2, Delphi, iOS, VCL, FireMonkey, Xamarin, Formuláře

Abstract

This thesis deal with analysis and design of the new presentation layer of information system K2. It compare available solutions for presentation layer of application with regard to UX requirements and availability of proposed solution on all platforms. Implementation of basic user interface on iOS and evaluation of costs and benefits is described in conclusion.

Keywords Presentation layer, IS, K2, Delphi, iOS, VCL, FireMonkey, Xamarin, Forms

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Práce se věnuje prezentační vrstvě informačního systému K2. Tento systém vyvijí česká společnost K2 atmitec s.r.o. již od roku 1991. Z původně jednoduchého účetního programu se postupem času vytvořil komplexní informační systém pro obchodní i výrobní firmy [7]. Stávající prezentační vrstva informačního systému je tvořena VCL frameworkem, jehož funkčnost a možnosti přestávají stačit. Proto vznikla potřeba navrhnout novou prezentační vrstvu, která bude vyhovovat všem potřebám pro další vývoj a rozšiřování na nové platformy.

Kapitola 1

Cíl práce

Tato práce se věnuje analýze stávajícího řešení prezentační vrstvy IS K2 (informační systém K2), požadavků od projektových manažerů systémové integrace a členů UX týmu. Stávající řešení prezentační vrstvy postavené nad standardními VCL formuláři nedokáže pokrýt potřeby dalšího vývoje. Trend informačních systémů se ubírá k možnosti ovládání nejenom z PC ale především z mobilních zařízení a dalších méně výkonných tenkých klientů. Jak se jednotlivé datové moduly rozšiřují o další pole, které jsou potřeba pro fungování systému, formuláře nad těmito moduly se stávají stále více nepřehledné. Každý uživatel nepotřebuje pro svoji práci všechny pole a stávající formuláře neumožňují uživatelskou úpravu.

Účelem této práce je tedy analyzovat všechny požadavky, vytvořit návrh řešení nové prezentační vrstvy IS K2 a implementovat základní rozhraní na platformě iOS. Pokud se navržené řešení osvědčí, vývojový tým K2 ho použije v nových verzích informačního systému pro desktop, web i mobilní systém Android a Windows Phone.

Kapitola 2

Analýza

2.1 Stávající řešení

Prezentační vrstva je postavena na komponentami, které jsou zděděny ze standardních VCL komponent [8]. Základní funkčnost prezentační vrstvy informačního systému zajišťují komponenty grid (*TxGrid*) a edit (*TxEdit*). Ty zobrazují data z datového modelu reprezentovaného třídou *TDataM*.

Třída *TDataM* poskytuje základní funkčnost pro načítání dat a jejich procházení. Je to hlavní bod pro přístup k datům nejen pomocí formulářů ale i pomocí nevizuálních skriptů.

Tabulka dat připojená k modulu může být bud přímo databázová tabulka nebo view, napojené přes třídu *TxFile*. Ta umí v současné době komunikovat pomocí driverů s databázemi MS SQL a Oracle. Druhým typem je takzvaná paměťová tabulka reprezentovaná třídou *TMemFile*. Ta je vždy uložena pouze dočasně v paměti počítače, kde běží instance informačního systému K2. Používá se často ve speciálních skriptech pro dočasné uložení dat, její výhoda je větší rychlost oproti databázovým tabulkám.

Prakticky všechny hlavičky dokladů a záznamů jsou postaveny nad *TDataM* nebo jeho potomcích. Každý modul má v sobě přidány property pro práci s jednotlivými poli, které můžou být několika typů. Fyzické pole je přímo napojené na pole v databázové tabulce. Takzvané počítané pole není napojeno na databázové pole přímo, ale svojí hodnotu při zápisu nebo čtení musí vypočítat. Toho se využívá například u cenových polí. Hlavičkový modul zakázky obsahuje počítané pole *brutto*, které slouží pouze pro čtení a obsahuje sečtenou celkovou cenu z položek zakázky. Na položkách je pole *brutto* také počítané a nikoliv fyzické. Při zápisu do něj dojde k napočítání hodnoty do pole netto (podle sazby daně) a ta je teprve uložená do fyzického pole v databázové tabulce.

Na pole, ať už fyzické či počítané, je možno připojit vazbu na jiný datový modul. Přes tuto vazbu je možné zobrazovat jiná pole s připojeného DM. Definovaná je pří vytváření instance, kde je určeno do jakého modulu bude

vazba odkazovat, přes jaký klíč a zda bude povinná a kontrolovaná. Například na zakázkách je číselné pole odběratel, které má definovanou vazbu do modulu odběratelů. To nám umožňuje pomocí speciálního formátu zápisu zobrazit hodnoty všech polí v modulu odběratelů.

Krom hlavičkových modulů jsou pro funkčnost informačního systému nutné i položkové datové moduly. Jejich třída vychází z *TDataM* ale jsou přizpůsobeny pro připojení na hlavičkový modul přes určité pole. Je ale možné s nimi pracovat i jako s hlavičkami a například si v modulu položek prodeje zobrazit všechny záznamy napříč všemi hlavičkami.

Jelikož *TDataM* poskytuje pouze základní funkčnost, jsou z něj zděděny další třídy. *TDPraSk* podporuje práci s právy na jednotlivé záznamy a pole. Dále je zde například *TD_Pozn* přidávající funkčnost poznámek nebo *TD_Doklad* pro práci s knihami záznamů.

Každý datový modul má několik stavů zobrazení záznamů. Ty určují jaké záznamy budou zobrazeny. Jedná se o tyto 4 stavy:

- **Kniha**
  Slouží pro zobrazení všech platných záznamů. Datové moduly dokladů, které dědí ze třídy *TD_Doklad*, jsou zařazeny do takzvaných knih. U těchto modulů způsoby tento stav zobrazení pouze záznamů, které náleží k vybrané knize.

- **Filtr** Filtr není definovaný pro všechny datové moduly. Pro jeho fungování je nutná přítomnost filtrovací tabulky v databázi. Každý podporovaný modul má vytvořenou tabulku, která v názvu obsahuje jeho zkratku, například FILTRZAK pro modul zakázek. V této tabulce je několik klíčových polí, různých pro každý modul, nutných pro identifikaci originálního řádku. Filtrovací tabulka umožňuje například transakční zpracování více záznamů.

- **Výběr** Umožňuje vyfiltrovat záznamy podle všech dostupných polí v datovém modulu i v jeho položkových modulech. Při definování se vybere příslušné pole, nastaví se filtrovací operátor (=, <, >, v seznamu, obsahuje, mezi, atd.) a určí se typ hodnoty:

  - **Konstanta** : přímo zapsaná hodnota k porovnávání
  - **Výraz**: zapsaný pomocí jazyku Pascal, je vyhodnocen při spuštění výběru
  - **Datové pole**: dojde k porovnání 2 polí pro každý záznam

  Po vložení všech filtrovaných polí a spuštění dojde nejprve k sestavení SQL dotazu. V něm jsou zahrnuty podmínky, které lze vyhodnotit přímo na SQL databázi. Výsledek je předán do datového modulu, kde se popřípadě projdou všechny vrácené záznamy a dofiltrují se podle podmínek, které nemohly být vyhodnoceny na SQL serveru.

- **Vše** Tento stav umožňuje zobrazení všech záznamů. Na rozdíl od režimu kniha zde jsou zobrazeny i zneplatněné záznamy (vizuálně oddělené) a u dokladů záznamy napříč všemi knihami.

Nad všemi stavy modulu je navíc možno označovat záznamy hvězdičkami. To lze využít pro rychlé akce nad skupinou záznamů. Dostupné je v hierarchii od třídy *TDPraSk*. Klasické procházení záznamů je možno pomocí metod datového modulu *DoNextDM*/*DoPriorDM* nebo pro označené položky *DoNextSelected*/*DoPriorSelected*. Tím dojde k naplnění všech polí datového modulu dalším nebo předchozím záznamem.

Záznam datového modulu je také možno potvrdit. Tím se do pole potvrzení zapíše aktuální datum a záznam je považován za vyřízený. Dále nemůže být měněn a je možno ho pouze prohlížet. Existuje i funkce pro odpotvrzení, která je přístupná pouze uživatelům s příslušným právem.

Krom funkcionalit přímo nad datovými moduly je možno v IS K2 využívat skripty a sestavy. Dělí se na standardní, které jsou vytvořeny přímo výrobcem IS K2 a na speciální, které tvoří sami uživatelé nebo programátoři servisních a implementačních firem.

- **Skript** – přímo v IS K2 se nachází editor skriptů. Ty slouží k práci s daty informačního systému. Editor vychází z IDE pro Delphi od společnosti Embarcadero [9]. Krom standardních Delphi tříd přidává editor vizuální i nevizuální třídy informačního systému, které jsou zveřejněny do skriptu. Je možno zde vytvořit skripty nebo formuláře, které se pak spouštějí manuálně, při určité akci nebo po uplynutí časového intervalu.

  Pokud nechceme ve skriptu vytvářet vlastní instance datových modulů, je v proměnné *AktDM* dostupná instance potomka *TDataM* nad kterou je aktuálně postaven uživatel v K2. Pro práci s formulářem, nad kterým je skript spuštěn, jsou dostupné proměnné *AktForm* s instancí formuláře a pro formuláře se záložkami je naplněna i proměnná *AktStr* s aktuální otevřenou stranou.

- **Sestava** – slouží k zobrazení dokladů a jiných informací z IS K2 s možností exportu do formátu PDF nebo vytisknutí na tiskárně. Typicky slouží pro zobrazení stránky zakázky, faktury objednávky, inventury, výrobního příkazu a jiných dokladů. V informačním systému je také obsažen editor pro sestavy. Je možno upravovat stávající nebo vytvářet nové, čisté. Do sestavy se naplní pole, přidá se k ním programový kód a na vlastním listu „papíru“ se definuje jejich zobrazení, seskupování a další vizuální věci.

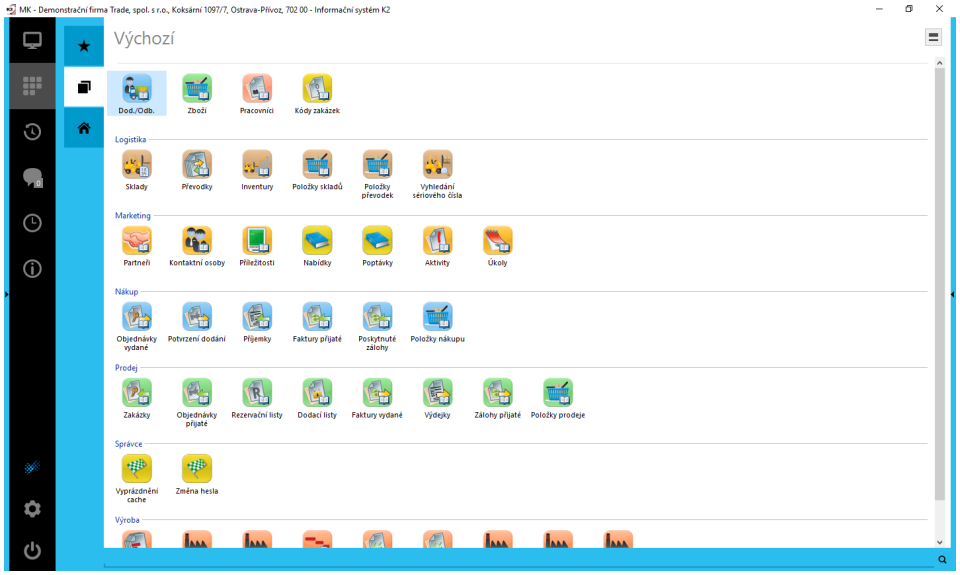
Datová vrstva je pro všechny klienty IS K2 stejná. Na desktopu v operačním systému Windows je nad ní postavena prezentační vrstva popsaná níže v sekci 2.1.1. Ta společně s datovou vrstvou tvoří hlavní část informačního

systému K2. V posledních několika letech vznikla potřeba pro přístup do IS K2 nejenom z desktopu s OS Windows ale i pomocí dalších platforem (sekce 2.1.2).

Přístup klientů je zajištěn pomocí AS (aplikačního serveru). Jedná se o Windows službu běžící na serveru, které využívá zdroje IS K2 a poskytuje prostředky pro paralelní zpracování požadavků více uživatelů z více platforem. Přímo s AS komunikuje webový klient a server webových služeb. Webové služby otevírají IS K2 pro aplikace třetích stran pomocí architektury REST. Pomocí nich komunikují i oficiální aplikace K2 pro platformu Android a iOS. V současné době je také dostupný e-shop K2 komunikující přes stejné rozhraní. Webové služby zprostředkovává autentizaci uživatelů, získávání parametrů a další služby nezbytné pro webové rozhraní. Data mohou být ve formátu JSON nebo XML. Pomocích webových služeb jsou dostupné pouze datové moduly, skripty a sestavy, které jsou ošetřené pro vícevláknové prostředí a které nevyužívají grafické komponenty.

2.1.1 Desktop

Nejpoužívanější přístup do IS K2 je pomocí desktopové verze. Ta je nainstalována na serveru a spouští se na jednotlivých uživatelských stanicích. Při přihlášení do informačního systému se zobrazí takzvaná plocha. V horní části se nacházejí záložky právě otevřených formulářů. V levé části jsou tlačítka pro zobrazení hlavního menu, naposledy spuštěných funkcí a formulářů, seznamu notifikací, historie záznamů a informačního panelu K2.



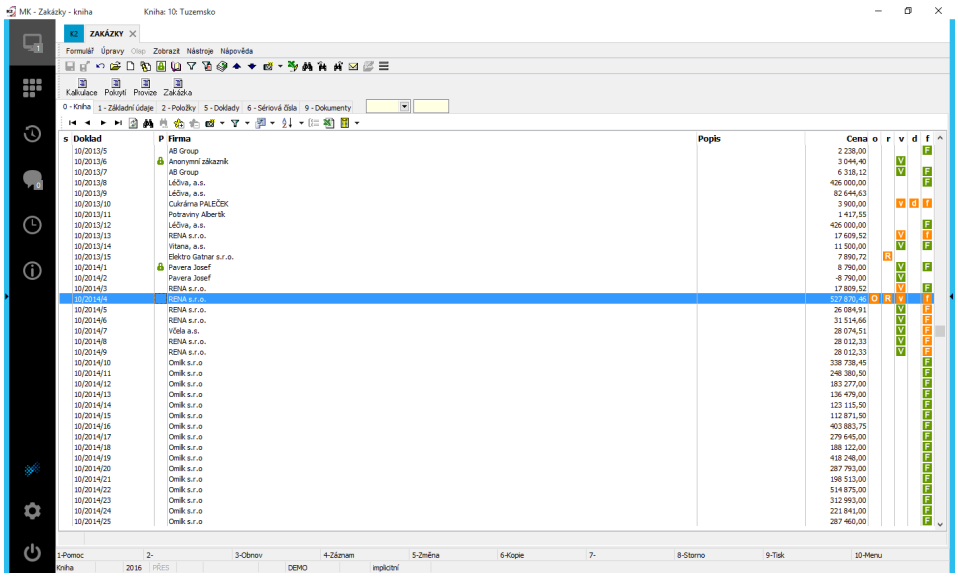
Obrázek 2.1: Výchozí menu

Menu obsahuje 3 záložky.

- **„Oblíbené“** – umožňuje každému uživateli přidat si libovolné zástupce, podobně jako v operačním systému Windows, pro spouštění datových modulů, funkcí, skriptů nebo sestav.
- **„Výchozí“** – ikony pro základní práci v IS K2 definované přímo v instalaci. Jsou rozděleny podle jednotlivých funkčních modulů.
- **„Hlavní menu“** – stromové menu, ve kterém jsou odkazy na všechny standardní moduly dostupné v K2. Obsahuje také uzel „Správce“ s odkazy do uživatelů, parametrů, číselníků i systémového nastavení K2.

Do menu je také možné přidat jednu nebo více vlastních záložek. V nich si můžeme definovat například funkce pro jednotlivé oddělení firmy. Celá plocha na všech místech podporuje drag&drop. Ze stromového menu můžeme zkopírovat odkaz na libovolnou knihu pomocí Ctrl+C nebo z kontextového menu. Tím dojde na pozadí k serializaci objektu ikony do XML a vložení do systémové schránky. Na jiné záložce menu je možno použít Ctrl+V nebo kontextové menu a vložit položku. Plocha pozná že se jedná o serializovaný objekt odkazu do modulu a vytvoří příslušnou ikonu.

Většina ze standardních modulů má vytvořen formulář ze třídy *TK2Grid*, ta je specifická gridem záznamů na 0. straně a dále několika stranami s detaily. Tato třída může být spuštěna nemodálně do záložky na hlavní ploše. Ostatní třídy formulářů jsou zobrazeny modálně nad základní plochou K2.



Obrázek 2.2: Formulář zakázky - 0. strana

Ve formuláři zakázky vidíme nahoře lištu pro hromadné akce formuláře a možnosti zobrazení. Pod ní jsou ikony pro obsluhu stavů a posunu mezi záznamy. Jsou zde tlačítka pro nový záznam, uložení, potvrzení, přechod mezi

záznamy a další. Tyto akce je možné obsluhovat i pomocí klávesových zkratek F1-10 s kombinací kláves Ctrl, Alt a Shift, které jsou vyznačeny ve spodní části formuláře. Pod touto lištou se nachází panel uživatelských funkcí. Zde si mohou uživatelé přidat vlastní tlačítka na spuštění skriptů, sestav nebo dávek. Dále vidíme jednotlivé záložky formuláře. Nultá strana obsahuje grid se seznamem záznamů. Na dalších stranách jsou zobrazování detailní informace pro záznam, na kterém je nastaveno pravítko na nulté straně. Zde jsou zobrazeny buď přímo pole datového modulu, nebo grid pro zobrazení položek, ať už vlastněných nebo nevlastněných.

Komponenta gridu zajišťuje zobrazení všech položek aktuálně načtených v datovém modulu. Vychází ze standatdní VCL třídy *TCustomGrid* a k ní přidává property DataM jako zdroj dat a další funkce a procedury pro práci s daty.

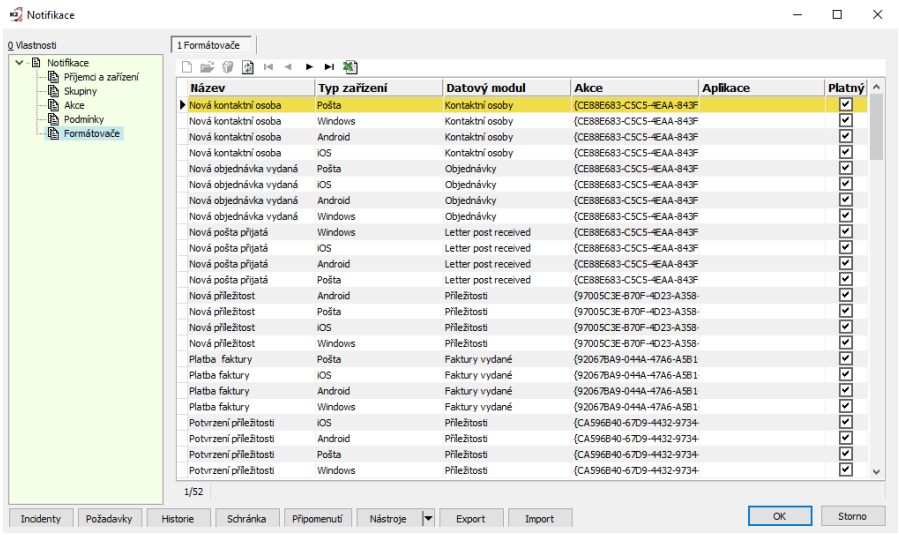
Aktuální záznam je vždy zvýrazněn takzvaným „pravítkem“. Uživatel si může definovat vlastní zobrazení sloupců. Je možno zobrazit jak fyzická pole tak i počítaná. Pole může mít v datovém modulu definovanou funkci pro převod hodnoty na ikonu, potom je ji možné zobrazit přímo ve sloupci gridu. Pro textové nebo číselné hodnoty se nastavuje formát zobrazení. Určuje délku zobrazeného textu nebo formát čísla. Pro každý datový modul jsou také definované implicitní sloupce. Grid umožňuje uživatelské řazení, nezávislé na indexu datového modulu. Pokud jsou ale data seřazena podle pole na kterém není index, dochází ke zpomalení některých funkcí. Grid také umožňuje definovat podmínky na sloupcích a tím obarvovat jednotlivé řádky.

Na dalších stranách formuláře jsou zobrazeny detailní informace o většině polích z datového modulu. Data jsou čtena ze záznamu, na kterém je nastaveno „pravítko“ hlavního gridu na 0. straně. Každé textové nebo číselné pole je možné zobrazit v komponentě *TxEdit*. Na datový modul jsou navázané pomocí property *ProgField*. Ta obsahuje název pole v datovém modulu, se kterým bude komponenta propojena. Pole, které mají definovanou vazbu do datového modulu, mají v komponentě zobrazenu ikonu šipky. Při kliknutí na ní je zobrazen buďto rozbalovací seznam s hodnotami, nebo dojde k otevření celého formuláře datového modulu na který vazba ukazuje. Text pole je možné ručně zadat do *TxEditu* i bez otevírání lookupu. Pokud je vazba definovaná jako kontrolovaná, je po dopsání textu a opuštění pole daná hodnota vyhledána v odkazovém datovém modulu. Uložení je možné pouze pokud je zapsaná hodnota nalezena. Ikona šipky je zobrazena i u datumových polí. U nich ale slouží k zobrazení komponenty kalendáře pro výběr hodnoty.

IS K2 umožňuje uživatelskou modifikaci standardních formulářů ve smyslu přidávání nových komponent. Tato vlastnost musí být zachována a pokud možno vylepšena. Pokud máme spuštěn formulář, v horním menu je dostupná položka „Editace formuláře“. Po její spuštění dojde k otevření formuláře v editoru skriptu s omezenými možnostmi. V tomto režimu není možné jakkoliv manipulovat se standardními komponentami, pouze je možné přidat nové. Je možno vložit popisek, edit, grid, a několik další komponent z nich odvozených.

Při přidání editu na standardní formulář stačí uvést do *ProgName* jméno pole z datového modulu a hned je možno s ním pracovat. U gridu je nutné využít property *ScriptName*. Do ní se zadá název speciální skriptu K2, který vytvoří novou instanci datového modulu a přiřadí ji do property *DataM* formuláře.

V informačním systému K2 je i možnost editace objektů bez vytvoření formuláře. K tomu slouží generované formuláře. Pomocí funkce *EditObject* je možné nechat vygenerovat formulář podle published property objektu. Ve skriptu jsou podporovány objektu *TScriptCollection*, *TScriptItem* a *TScriptPersistant*. Z objektu jsou přečteny RTTI informace a podle nich jsou pod sebe vygenerované pole typu *TxEdit*, popřípadě je zobrazen grid. Jednotlivé objekty mohou být do sebe vnořené a například můžeme mít kolekci objektů, které obsahuje svojí vlastní kolekci a tak dále. Generované formuláře se především používají pro nastavování vlastností a chování systému nebo skriptů. Výhodou je jejich rychlý vývoj. Stačí nadefinovat nového potomka některé z podporovaných tříd a bez vytváření formuláře ho můžeme ihned používat. Kód třídy obsahuje pouze nezbytné minimum informací pro práci s objektem. Vlastnosti jednotlivých property se definují pomocí metody *DefineClassProperties*. Je možné nastavit vazby na pole, styl zobrazení, popisky a další vlastnosti. Krom funkce *EditObject* je možné využít *CollectionLookup*. Ta pro potomky *TScriptCollection* dokáže zobrazit grid jako u *EditObject*, který slouží pouze pro čtení a místo přidání nové položky je dostupná funkce výběru položky.



Obrázek 2.3: Generovaný formulář

2.1.2 Klienti

Pro obsloužení všech klientů IS K2 slouží aplikační server K2. Pro AS je dostupný upravený editor skriptů a sestav. Ten neobsahuje žádné vizuální

komponenty a v třídách zveřejněných pro skript nejsou dostupné property a metody, které mají cokoliv společného s vizuálním rozhraním.

Na přímo s AS komunikují pouze SWS (Server webových služeb) a webový klient. Webový klient udržuje s aplikačním serverem persistentní spojení, tudíž může s datovými moduly pracovat stejně jako plný klient. Stejně tak je to i s licencováním. Při přihlášení se spotřebuje klasická konkurenční licence. Oproti tomu aplikace komunikující pomocí SWS a jeho REST rozhraní nemohou využívat klasické persistentní spojení. Při načtení dat se přenáší i časové razítko. Při následném ukládání se na serveru kontroluje, jestli během doby editace nedošlo ke změně dat jiným uživatelem. Pokud ano, k uložení dat nedojde. SWS ze své podstaty tedy dokáže v jednu chvílí obsloužit několik uživatelů webových služeb pomocí jedné licence do IS K2. U každého uživatele není přímo definovaný časový úsek od načtení do uložení dat. Každé zavolání URL webových služek je tedy jeden samostatný požadavek. Z tohoto důvodu není licencování přímo na každého uživatele, ale licencuje se spojení mezi SWS a aplikačním serverem. Tato takzvaná sdílená licence dokáže v jednu chvíli obsloužit desítky uživatelů.

- **Silverlight**

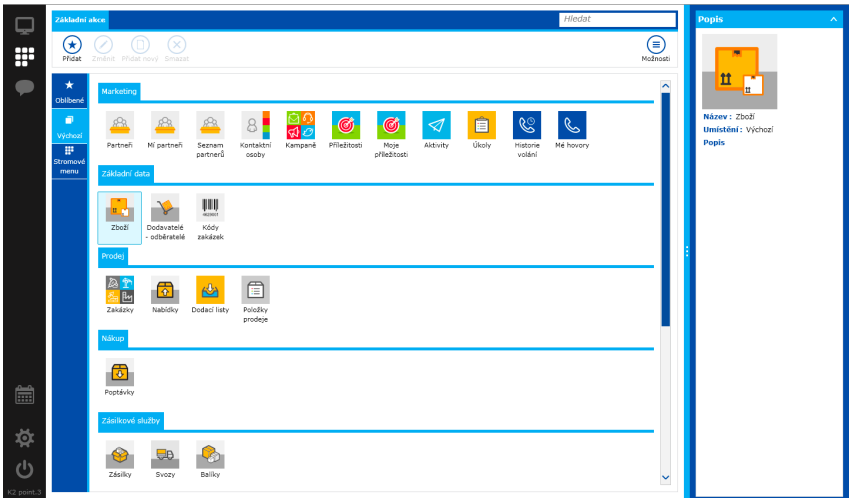
  IS K2 lze na počítačích využívat nejen přímo v počítačové síti firmy ale také pomocí webového klienta. Přihlášení do IS K2 probíhá pomocí stejných údajů a je stejně licencované jako přístup do plného klienta K2. Ten je realizován pomocí technologie Silverlight. Pro spuštění v prohlížeči je nutné mít nainstalovaný zásuvný plugin, jeho kompatibilita je zobrazena níže v tabulce.

OS	IE	Firefox	Safari	Chrome	Edge
Windows 7 – 10	X	X		X*	
Windows Vista	X	X		X*	
Windows Server 2012	X	X		X*	
Windows XP		X	X	X*	
OS X 10.6 – 10.11		X	X	X*	

** - pouze do verze 41*

Tabulka 2.1: Kompatibilita pluginu Silverlight [5]

  Webový klient neobsahuje všechny datové moduly a funkčnosti plného klienta K2. V současné době funkčnost pokrývá oblasti CRM, Dodavatelé / Odběratelé, Zboží, Prodej, Callcentrum a Zásilkové služby. Jednotlivé dostupné moduly musely být upraveny pro běh na aplikačním serveru. Standardní moduly mají část logiky definovanou v VCL formulářích plného klienta, například v událostech komponent *OnExit*. Tato logika musela být u každého formuláře přepsána do komponent frameworku Silverlight.



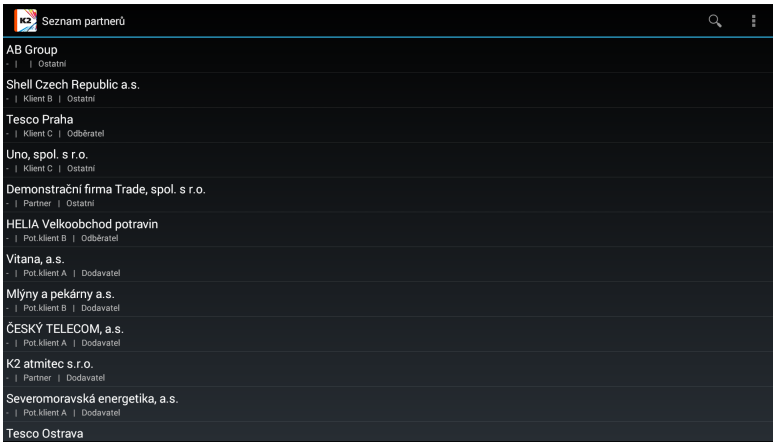
Obrázek 2.4: K2 4WEB

Současný klient používá statickou definici formulářů pomocí XAML jazyka. Jediným rozdílem oproti standardu je možnost vnořovat do sebe jednotlivé XAML souboru. Je tím urychlen vývoj nových formulářů pro tuto platformu, kdy není nutné znovu definovat již vytvořené části formulářů.

- **Android** Mobilní operační systém Android je jeden z nejpoužívanějších a proto se klientu na tuto platformu věnovalo více času. Základní funkčnost je podobná jako ve webovém klientu Silverlight. Základem tedy jsou moduly CRM systému, notifikace, workflow a analytické vyhodnocení pomocí dashboardů. Každý odkaz na modul v sobě obsahuje navíc ikony pro hledání a aplikování filtrů na záznamy. Po otevření knihy je dostupný seznam záznamů. Kvůli úspoře přenášených dat a rychlosti odezvy se vždy načítá jen určitý počet záznamů v jednom spojení se serverem. Výchozí hodnota je 30 záznamů. Při srolování stránky na konec klient vyšle požadavek na stáhnutí dalších 30 záznamů a takto jsou podle potřeby načteny všechny záznamy. Konkrétně jsou zde dostupné knihy:

  - Partneři
  - Kontaktní osoby
  - Úkoly
  - Příležitosti
  - Kampaně
  - Aktivity

- Faktury
- Notifikace
- Dashboardy
- Workflow
- Zakázky

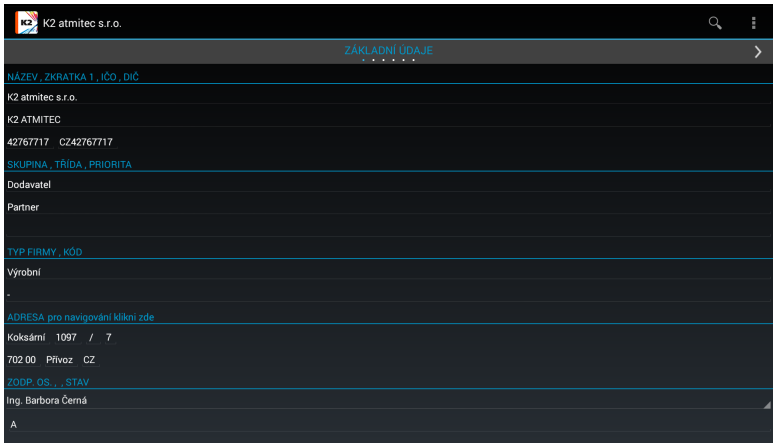


Obrázek 2.5: Android klient - seznam Partnerů

Grid pro zobrazení seznamu záznamů je vesměs stejný napříč všemi knihami. Zvýrazněn je název položky (popř. číslo dokladu) a pod ním je v seznamu zobrazeno několik důležitých polí. Po kliknutí na konkrétní záznam je zobrazen formulář všech dostupných detailních informací. Jednotlivá pole a položkové moduly jsou zobrazeny na formuláři rozděleného na jednotlivé stránky. Stejně jako na všech klientech i zde je problém dopracovat kompletně všechnu funkčnost a dostat informace na menší obrazovku zařízení se systémem Android. Oproti webovému klientu nabízí Android klient vetší integraci do operačního systému. Přístupné jsou funkce:

- Ukládání hovoru – zaznamená se číslo a čas zahájení hovoru
- Vyhledávat neznámé kontakty – neznámé telefonní číslo je vyhledáno v databázi partnerů
- Aktivita z hovoru – předvyplnění informací o hovoru do aktivity
- Vytvoření kontaktní osoby – vytvoření kontaktní osoby v IS K2 z kontaktu v telefonu

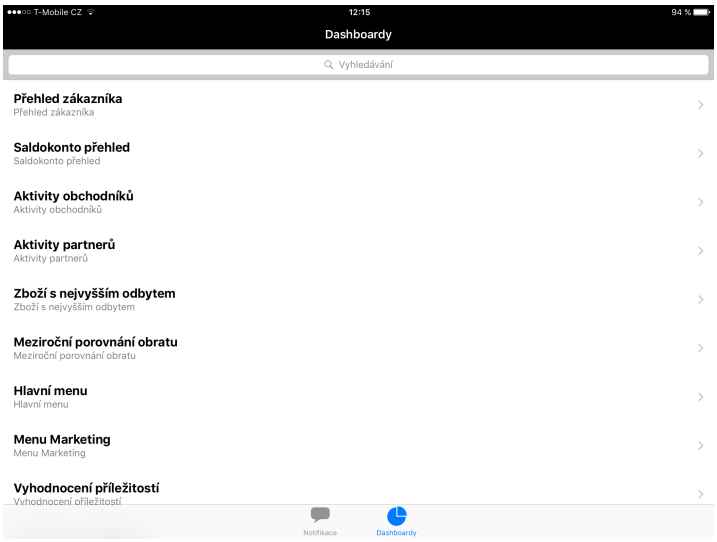
V klientu je také dostupný widget na plochu. Umožňuje zobrazit jeden vybraný dashboard přímo bez otevírání aplikace.



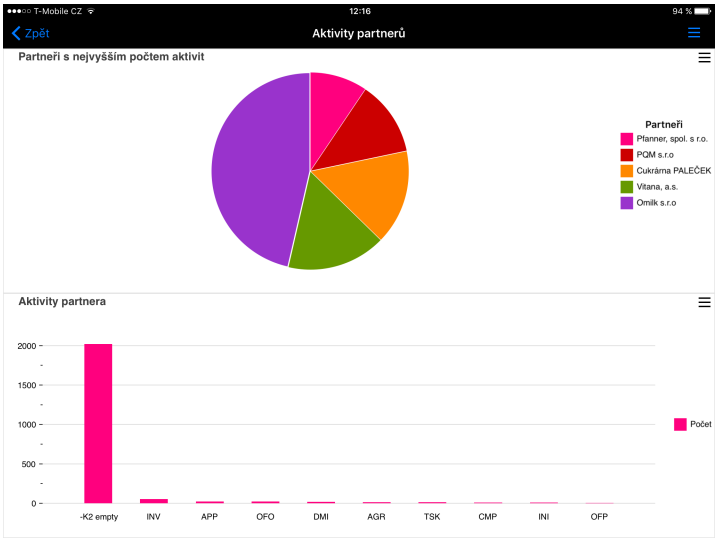
Obrázek 2.6: Android klient - detail Partnerů

- **iOS**

  Další aplikací využívající SWS pro přístup do IS K2 je iOS klient. Ten v současné době má pouze 2 funkčnosti: příjem notifikací a zobrazování dashboardů. Pro notifikace je nakreslen formulář zobrazující seznamu a následně i detail záznamu. Vzhled i chování je přizpůsobeno platformě iOS a je jiné než v Android a Webovém klientu. Druhý dostupný modul jsou dashboardy. Jejich vzhled a logika se také mírně liší od ostatních klientů.



Obrázek 2.7: iOS klient - seznam



Obrázek 2.8: iOS klient - detail

2.2 Problémové části

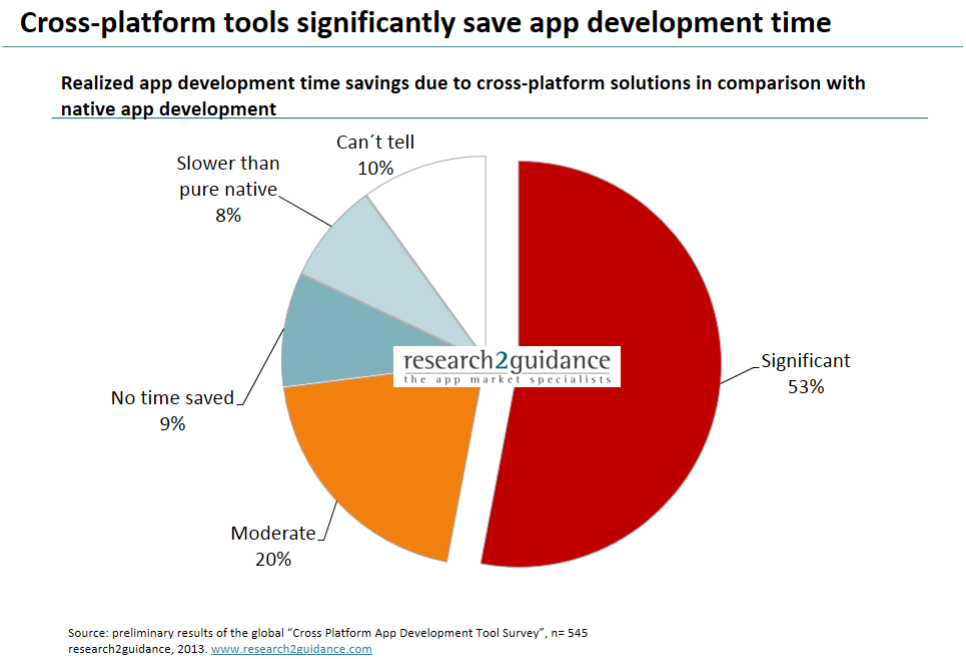
Aby se celý informační systém mohl posouvat dopředu a dále vyvíjet, je nutné odstranit chyby a vlastnosti stávajícího systému, které další vývoj zpomalují nebo přímo neumožňují.

Největší problém způsobuje roztříštěné vývojové prostředí a jazyky jednotlivých platforem.

Plný klient je vyvíjen v IDE Delphi 10 v jazyku Object Pascal, webový klient v Silverlight frameworku pomocí C# a mobilní klienti v Android SDK(Java) a iOS SDK(Objective-C).

Tento způsob sice zaručuje nativní, plně funkční aplikace bez omezení ale drasticky zvětšuje čas pro opravy chyb a vývoj nových funkčností. Výhoda nativních aplikací je především rychlost a uživatelský zážitek z aplikace. Existují zde cross–platform vývojová prostředí pro tvoření aplikací v jednom vývojovém prostředí, které následně vygenerují nativní kód na obě nejpopulárnější mobilní platformy [1]. V době vzniku mobilních klientů ale neposkytovaly dostatečnou funkčnost a výkon, aby mohly být využity. Chyběla například podpora pro push notifikace, které tvoří základní kámen všech mobilních platforem. V současnosti cross–platform vývojová prostředí obsahují podporu pro většinu API na všech platformách a tudíž se i více hodí pro vývoj mobilních aplikací.

Dalším nevýhodou stávajících VCL formulářů v plném klientu je pevná definice komponent. Formuláře jsou pevně nadefinované od vývojového týmu K2 u standardních, nebo konzultantů u speciálních a lze na ně pouze přidávat komponenty. Při editaci se ve složce K2 vytvoří textový soubor, který definuje



Obrázek 2.9: Cross–platform vývoj – úspora času [1]

přidané komponenty s vlastnostmi. Tento systém je ale dostupný pouze v plném klientu a nikoli ve webovém a mobilních klientech.

Jelikož v IS K2 pracují nad jednotlivými datovými moduly uživatelé z různých oddělení, pozic a s různými právy, není vždy vhodné, aby jednotlivé formuláře vypadaly vždy stejně. Ve stávajícím řešení chybí jakákoliv podpora pro uživatelskou modifikaci standardních formulářů.

Současný webový klient, postavený nad technologií Silverlight, se pomalu dostává do fáze, kdy bude muset být něčím nahrazen. Společnost Microsoft bude podporu pro tuto svojí technologii držet nejdéle do konce roku 2021 [10]. Bude tedy nutné najít jinou technologii pro prezentační vrstvu ve webovém prohlížeči. Současný webový klient také není kvůli nepřítomnosti pluginu dostupný v operačním systému Linux a na mobilních platformách.

Spoustu kódu logické vrstvy aplikace je v plném klientovi integrováno v kódu formulářů. To způsobuje problémy při rozšiřování jednotlivých modulů na ostatní klienty. Kód se musí znovu přepisovat a tím vzniká prostor pro chyby a roztříštěnost mezi verzemi a platformami.

V současnosti je na každé platformě každý formulář zobrazen jinak. Není dodržen žádný standard co se týče rozložení komponent. Na každé platformě jsou komponenty v jednotlivých modulech přidávány podle potřeby bez toho, aby se změna promítla do více zobrazení.

2.3 Požadavky

Výsledný produkt musí splňovat několik základních požadavků, aby umožnil rychlejší vývoj nových funkčností a pokryl požadavky vývojového týmu i zákazníků IS K2.

Zásadní požadavek je zrychlení vývoje a dostupnost všech datových modulů na všech dostupných operačních systémech a platformách. Výhodou kompletní prezentační platformy IS K2 pro všechny zásadní OS bude dostupnost nejen standardních ale i speciálních skriptů, sestav i celých datových modulů s formuláři.

Formuláře by měli na všech platformách zobrazovat stejné informace i ve stejném zobrazení s optimalizací pro velké obrazovky s velkým rozlišení i malé obrazovky mobilních telefonů a tabletů. Systém by měl umět definici různých zobrazení pro jednotlivé velikosti obrazovek.

Celý informační systém je velice komplikovaná aplikace s mnoha formuláři a velkou funkčností, proto je potřeba zachovat funkční stávající prezentační vrstvu a vedle ní paralelně vytvořit novou. Celý přechod tak bude muset být postupný.

V současných formulářích je možno najít více společných částí. Upravovat takto společné části pak vyžaduje více času a také zvyšuje chybovost při opomenutí některé z nich. Tuto vlastnost je nutné vzít v potaz pří návrhu nové vrstvy.

Z tohoto vycházejí tyto základní cíle:

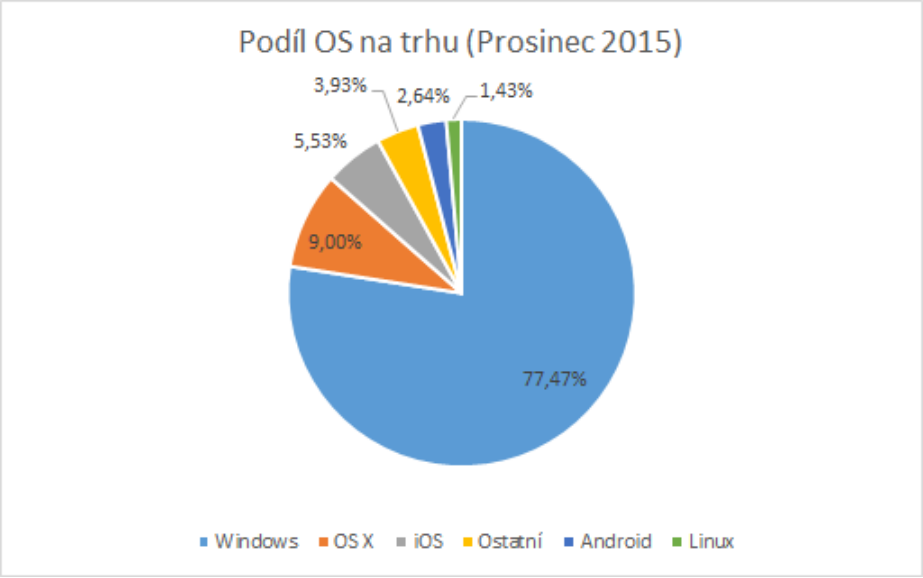
- Přesun klientské logiky (vazby, závislosti, výpočty) z prezentační vrstvy plného klienta do aplikační vrstvy datových modulů.
- Jednotná definice formulářů pro všechny klienty K2 na všech platformách
- Možnost měnit styl formuláře
- Definice uživatelských pohledů na formulář pro různé uživatelské skupiny
- Souběžné fungování starého a nového UI
- Možnost definovat společné části formulářů na jednom místě
- Zjednodušení vytváření formulářů
- Přehlednější zobrazení všech informací
- Bezstavový přístup k celé funkcionalitě pomocí AS

V současné době je plná funkčnost IS K2 dostupná pouze na operačním systému Windows jako 32b nebo 64b aplikace. Omezená funkčnost v podobě

CRM a balíkových služeb je dostupná pomocí pluginu webového prohlížeče na Windows a OS X, a jako nativní aplikace pro systém Android. Systém iOS zpřístupňuje pouze analytické vyhodnocení a notifikace. Cílem je dostat plnou verzi IS K2 na platformy:

- **Microsoft** – desktopová Windows aplikace
- **Linux** – Android klient
- **Apple** – iOS klient

Tím bude plný klient IS K2 dostupný pro všechny hlavní mobilní a desktopové platformy. Z obrázku 2.10 je ale vidět, že další rozšiřování plného klienta by mělo směrovat na operační systém OS X. Jeho 9% podíl je nezanedbatelný, tudíž počítače s tímto systémem se mohou ve společnostech využívající IS K2 objevit.



Obrázek 2.10: Podíl OS na trhu – desktop a tablety [2]

Operační systém Linux je využíván především na serverech a proto není potřeba rozšiřovat plného klienta i na tento desktopový systém. Mobilní systém Windows Phone není podle průzkumu mezi uživateli IS K2 významný, není tedy nutné mu věnovat velkou pozornost.

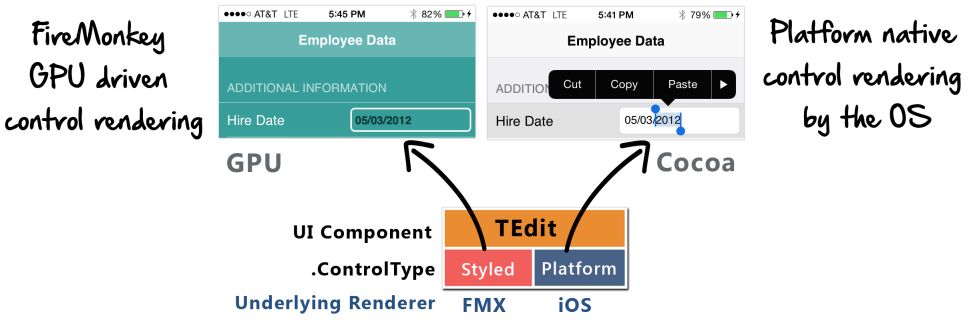
2.4 Dostupné varianty

2.4.1 FireMonkey

Ze současných řešení na trhu se jako první nabízí řešení přímo od společnosti Embarcadero a jeho FMX (FireMonkey) knihovna. Je to vizuální knihovna, která je dostupná spolu s VCL knihovnou od Delphi verze XE2. Oproti starší VCL má několik výhod:

- **Vizuální stylizace komponent**
  Všechny obrázky a vizuální komponenty jsou ve FMX vektorové, tudíž nedochází k degradaci grafiky na obrazovkách s vysokým rozlišením. U jednotlivých komponent je možno pomocí vlastností „StyleBook“ a „StyleManager“ integrovat různé styly a přepínat mezi nimi.
- **Podpora platform MS Windows, Mac OS X, iOS a Android**
  Přibližně 90% kódu aplikací vytvořených pomocí FMX je nezávislých na cílovém OS. Aplikace vytvořené pomocí této knihovny využívají nativní prvky cílového OS a nevyžadují další mezi vrstvu pro běh. Krom nativních komponent jednotlivých OS lze použít i vizuální komponenty přímo z FMX knihovny, které se vykreslují přímo na GPU daného zařízení. Zbylá část kódu se určí pro konkrétní platformu pomocí podmíněného překladu.
- **Podpora grafických efektů na GPU**
  Framework podporuje technologii ImageFX, které doposud využívali pouze specifické grafické editory jako třeba Adobe Photoshop. Je možno používat filtry obrázků, transformace a další grafické efekty. Všechny výpočty probíhají přímo na GPU v samostatném vlákně a tudíž nevyužívají CPU. Výkon procesoru je tedy plně dostupný pro aplikační logiku.
- **Možnost používat LiveBindings**
  Tato technologie slouží pro jednoduché propojení více objektů a interakci mezi sebou. Může se například připnout kontrola *TEdit* na *TLabel* a při změně hodnoty v editačním poli dojde automaticky k přizpůsobení textu v popisku.
- **Prototypování aplikace**
  Přímo ve vývojovém prostředí je možné vytvářet prototypy aplikací pro různé platformy. Je zde možnost vytvoření jak horizontálního tak vertikálního prototypy. Horizontální nám umožňuje definovat rozsah aplikace z pohledu funkcionalit nebo uspořádání vizuálních prvků. Naproti tomu vertikální se zaměřuje na jednu funkcionalitu a prototyp zasahuje od UI přes aplikační logiku až k datové vrstvě.
- **FireUI**
  Koncept FireUI umožňuje univerzální vývoj uživatelského rozhraní pro

různé operační systémy a velikosti zobrazovací plochy. Pro formulář se navrhne základní formulář, který je následně přizpůsobován pro různé platformy a zařízení. Pomocí vlastnosti *ControlType* lze například na komponentě *TEdit* nastavit, zda se bude vykreslovat na iOS zařízeních pomocí nativního ovládacího prvku *UITextField*, nebo je vykreslen upravený ovládací prvek FMX knihovny.



Obrázek 2.11: FMX – nativní ovládací prvky v iOS [3]

Nevýhoda knihovny FMX je nekompatibilita s knihovnou VCL. Kvůli rozdílnosti obou knihoven nemohou být obě zároveň použity ve stejném modulu.

FMX obsahuje kompletní sady drag&drop ovládacích prvků, stejně jako VCL. Základní třída z které dědí všechny vizuální komponenty FMX je *TFmxObject*. Ta poskytuje funkcionalitu pro vytváření kopírování a uvolňování objektů. Umožňuje také serializaci do streamu dat a následně deserializaci z něj. Všem komponentám odvozeným z této třídy umožňuje využívat styly a animace.

Dostupné jsou například třídy *TEdit* nebo *TCustomGrid*, z kterých dědí základní komponenty současné prezentační vrstvy IS K2.

Obrázek 2.12: FMX – TEdit

Obrázek 2.13: VCL – TEdit

Třída *TPresentedControl* poskytuje oddělení dat a zobrazení komponenty. Dále umožňuje nastavit styl zobrazení na jednotlivých platformách. Pokud nelze použít nativní styl pro danou platformu, je komponenta vykreslena pomocí univerzálního FireMonkey stylu [11].

2.4.1.1 Výhody

- **Multiplatformní vývoj aplikací**
  Pomocí FMX frameworku je možno v jednom prostředí vytvořit multiplatformní aplikace fungující jak na desktopových systémech MS Windows a Mac OS X tak na mobilní iOS a Android. Díky tomu že je valná většina kódu platformě nezávislá, není potřeba tolik využívat podmíněný překlad pomocí *{$IFDEF}*.

  Pro vývoj multiplatformní aplikace s podporou systému Android je nutné nainstalovat do počítače Android SDK. V RAD Studiu je následně nutné přidat cílovou platformu Android a přidat cestu k SDK. Přímo z RAD Studia je možné ladit aplikaci na fyzickém telefonu připojeném přes USB.

  Vývoj pro platformu iOS není tak pohodlný. Vedle počítačem se systémem MS Windows je nutné mít i počítač s Mac OS X. Ty musejí být propojené počítačovou sítí. Doporučená je i varianta pouze jednoho počítače s OS X, na kterém běží virtualizovaný systém Windows s RAD Studiem. Na OS X musí být také nainstalován nástroj Platform Assistant. Ten zajišťuje propojení vývojového prostředí s laděním iOS zařízení připojeného k počítači s OS X systémem.

- **Jednotné zdroje**
  Všechny standardní formuláře IS K2 pro všechny platformy by bylo možné nadefinovat pomocích jednoho zdrojového kódu. Na každé platformě by ale aplikace běžela nativně a nebylo by potřeba využívat mezivrstvu. Tím by byl zajištěn rychlý běh na všech platformách.

  Po předělání editoru skriptů a sestav by bylo možné i tomto novém rozhraní využívat speciální úpravy, které jsou nezbytné pro fungování IS K2 ve firmách.

- **Webový klient**
  Při používání FMX knihovny lze použít nástroj WebFMX. Ten umožňuje publikovat FMX aplikaci na web a přistupovat k ní z jakékoliv platformy s HTML5 webovým prohlížečem. Jakákoliv aplikace kompilovaná s pomocí WebFMX má duální kompatibilitu a může být využita jak jako Windows tak jako HTML5. Tento nástroj v sobě integruje vlastní webový server, který je využíván pro zpracování požadavků z webového prohlížeče. Vzhled aplikace publikované přes webové rozhraní je velmi podobný Windows aplikaci a nabízí i podobný uživatelský zážitek [12]. Webový klient vygenerovaný pomocí tohoto nástroje má tyto vlastnosti:

  - Multiplatformní aplikace spustitelná na jakémkoliv zařízení s HTML5 prohlížečem.
  - Možnost generovat a tisknout PDF
  - Přesměrování souborů pro možnost nahrávání dat z prohlížeče do aplikace
  - Metody pro stahování souborů
  - Dostupnost informací o prohlížeči pomocí *RemoteInfo* objektu
  - Správa připojení jednotlivých klientů. Umožňuje znovu navázat spojení při ztrátě.
  - Implementace *MessageDialog* a *InputQuery*
  - 2 módy zobrazení písem: webové fonty nebo bitmapové
  - Události aplikace oznamující změnu velikosti okna nebo zavření panelu
  - Podpora JavaScript SDK knihovny

2.4.1.2 Nevýhody

- **Datové typy**
  Některé datové typy nejsou podporované v Delphi Mobile Compilers. Musí být odstraněny nebo nahrazeny alternativami použitelnými na mobilních platformách. Popsané jsou níže v tabulce 2.2

Datové typy v desktopových aplikací	Datové typy v mobilních aplikací
System.WideString	System.String
System.AnsiString, System.ShortString, System.RawByteString, System.UTF8String	Přímá náhrada není, pouze „array of byte“
System.AnsiChar	System.Char, System.Byte, System.UInt8
System.PAnsiChar, System.PWideChar	System.SysUtils.TStringBuilder, System.String, System.MarshaledString
System.Openstring	Přímá náhrada není, pouze „array of byte“

Tabulka 2.2: FMX Datové typy

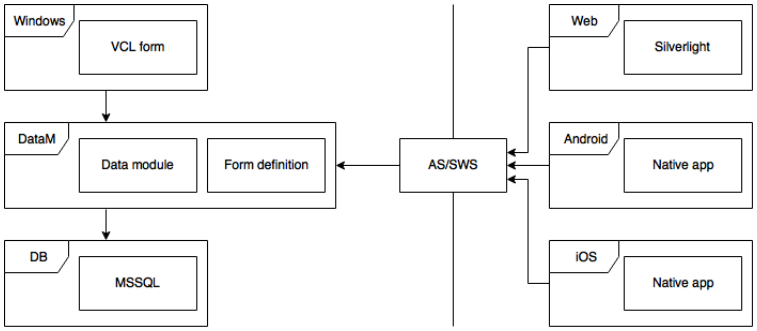
- **Nekompatibilita**
  Při přechodu z VCL na FMX by bylo nutné zahodit všechny stávající zdroje prezentační vrstvy K2 a kompletně vše naprogramovat znovu. Krom zdrojů prezentační vrstvy plného desktopového klienta by se nepoužili kompletní zdroje stávajícího webového Silverlight klienta ani mobilních klientů pro Android a iOS. Musel by se taký upravit editor skriptu a sestav v IS K2 a všechny speciální skripty postavené na formulářích.

- **Jednorázový přechod**
  Při přepsání prezentační vrstvy IS K2 do frameworku FMX by bylo nutné celý program předělat najednou. FMX neumožňuje souběh s VCL knihovnou. Bylo by tedy nutné paralelně s údržbou současného řešení odděleně vytvářet novou podobu IS K2.

[13]

2.4.2 VCL

Pro dalším rozvoji IS K2 lze postupovat i se současnou knihovnou VCL pro prezentační vrstvu aplikace. Musí se ale odstranit všechny stávající nevýhody a překážky, které brání rozvíjení funkčnosti na všechny desktopové a mobilní platformy. Problém VCL knihovny je, že je jí možné použít pouze v operačním systému MS Windows. Zachováním této knihovny prezentační vrstvy se tedy nevyřeší problém s více zdroji pro všechny dostupné platformy. Pokud by se jednalo pouze o zdroje pro desktopové systémy, VCL neumožní ani rozšíření na platformy Mac OS X ani Linux. Webový a mobilní klienti by se také museli řešit jinými knihovnami. Toto řešení by vyžadovalo vytvořit novou vrstvu aplikace mezi aplikační logikou a samotnou prezentační vrstvou.



Obrázek 2.14: Definice formulářů – struktura

Stávající definice VCL formulářů plného klienta by se z větší části přesunula do této nové nižší vrstvy, která by byla přístupná i pomocí aplikačního serveru a mohli by jí využívat všichni klienti. Tím by se zajistila jednotná definice pro všechny klienty a zachovala by se valná většina stávajícího kódu.

Do definice formulářů by se přesunulo rozmístění jednotlivých vizuálních komponent, struktura vnoření do sebe a přiřazení komponent k datům. Bylo by nutné vyřešit uživatelskou definici formulářů. Musel by být vytvořen nový editor formulářů, který by pracoval pouze s vrstvou definice. V novém editoru by se uživatel nedostal ke komponentám VCL knihovny, pouze k abstraktním třídám. Odstínění od nejvyšší prezentační vrstvy umožní jednodušší definici. Tu by měl zvládnout i běžný uživatel K2, ne pouze konzultant nebo programátor.

2.4.2.1 Výhody

- **Zachování většiny kódu**
  Většina stávajících vizuálních komponent IS K2 je zděděna ze standardních VCL komponent a přidává kód pro specifické fungování a obsluhu dat. Zachování knihovny by umožnilo využít tento stávající kód a pouze ho pozměnit pro běh nad touto novou vrstvou pro definici formulářů. Webový klient i mobilní klienti by definici rozložení formulářů také získávali z této nové vrstvy. Současně s definicí nového rozložení formulářů by muselo dojít k přesunu aplikační logiky z prezentační vrstvy formulářů do datových modulů. Tím bude přístupná i pro tenké klienty pomocí rozhraní aplikačního serveru. Jedná se hlavně o navrhnutí nové funkčnosti pro obsloužení událostí formulářů jako *OnClick*, *OnExit* a *OnKeyPress*, které slouží pro práci nad datovými poli a dále volání funkčností pro přechod mezi záznamy, a uživatelské akce.

- **Vlastní řešení**
  Při vytvoření nové vrstvy pro definici formulářů je možné od začátku navrhnout strukturu přesně podle požadavků pro další vývoj a poznatků

od oddělení softwarové integrace. Celý koncept se může připravit na míru tenkým klientů. Vlastní řešení také umožní uživatelskou editaci rozložení, které by odpovídalo UX zvyklostem v K2.

- **Šablony**
  Ve vlastním řešení může být vymyšlen systém „recyklace“ komponent, které jsou shodné přes několik formulářů. Systém by umožňoval využití i celých bloků komponent standardních formulářů nadefinovaných přímo vývojovým týmem. Tím by se pro uživatele zjednodušila tvorba nových formulářů.

- **Souběh nového rozhraní**
  Nová prezentační vrstva nad definicí formulářů by mohla koexistovat se stávajícími VCL formuláři. Stávající funkčnost by se musela upravit, aby se přizpůsobila přepisu obsluhy událostí do datových modulů. Definice formulářů nebudu zasahovat do stávajícího systému formulářů a proto umožní souběh obou prezentačních vrstev. Bude pouze sloužit pro generování současných VCL komponent do požadovaného rozložení.

- **Publikování přes API** Výhodou tohoto řešení je možnost publikovat rozložení formulářů pomocí stávajícího API webových služeb. Pokud by byla definice postavena nad třídou *TDataM*, mohla by se publikovat bez dalších úprav na AS a SWS.

2.4.2.2 Nevýhody

- **Webový klient**
  Definice formulářů popsaná daty řeší problém s rozložení ale už neřeší budoucnost webového klienta. Tato varianta umožní v současném webovém klientu číst rozložení formulářů přímo z IS K2. Bude nutné upravit vykreslování formulářů, které se změní z pevné definice na generování komponent podle načteného rozložení z K2. S využitím stávajících VCL komponent nelze vygenerovat webovou aplikaci.

  Existuje například uniGUI framework [14], který umožňuje z jednoho zdrojového kódu a vizuálních komponent vytvořit VCL aplikaci i webového klienta. Webový klient je zde ale jen integrovaný webový server přímo v aplikaci, který obsluhuje požadavky z webu. Pro funkčnost webového klienta je tedy nutná spuštěná desktopová aplikace. Ta nemůže běžet jako služba systému Windows ale pouze jako klasická Windows aplikace. Problém s webovým klientem IS K2 tedy neřeší.

- **Podpora**
  VCL knihovna není v současné době hlavní priorita u tvůrce Delphi. Společnost Embarcadero se spíše zaměřuje na vývoj multiplatformního frameworku FireMonkey. Proto zde existuje možnost, že při odladění

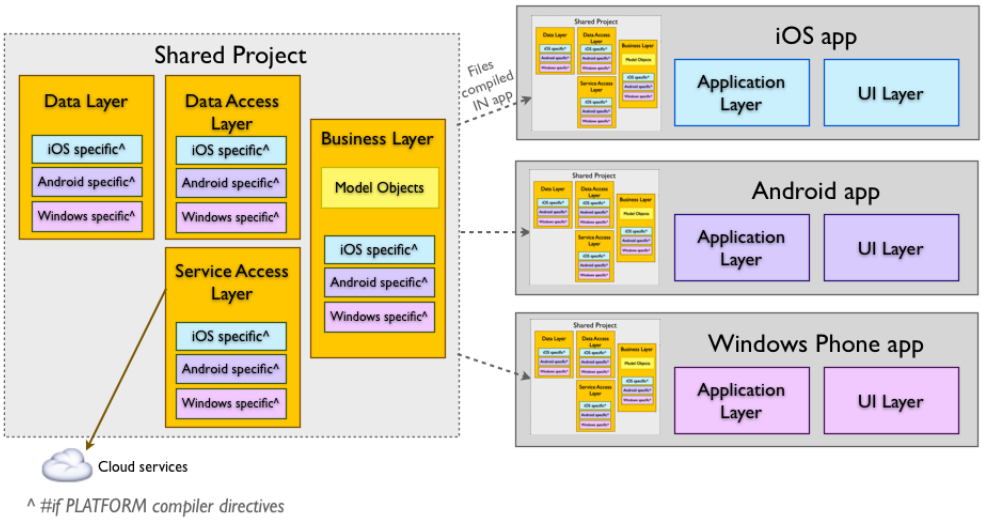
FMX knihovny dojde k ukončení podpory VCL. Pokud by ale všechny formuláře v desktopové verzi K2 byly definované pomocí mezivrstvy definice formulářů, výměna vizuální knihovny poslední prezentační vrstvy by nemělo vliv ani na programátory, ani na uživatele. Všechny standardní i speciální formuláře by byly definované o vrstvu níže, pouze by se vykreslily pomocí jiné knihovny.

- **Pouze Windows**
  Zachování VCL knihovny neumožní mít jedny zdrojové kódy prezentační vrstvy pro všechny desktopové systémy. Pokud by v budoucnu byly vyžadovány klienti pro systémy Mac OS X a Linux, bylo by nutné vytvořit jednoho nebo dva tenké klienty s komunikací pomocí REST API webových služeb K2. V současné době ale není podpora systémů Mac OS X a Linux nutná podmínka nové prezentační vrstvy.

2.4.3 Xamarin

Xamarin je jeden z nejznámějších a nejrozšířenějších multiplatformních vývojových prostředí. V současné době ho využívá více jak 1 400 000 vývojářů. Na sklonku roku 2016 celou společnost odkoupila firma Microsoft a zakomponovala Xamarin do svého vývojového prostředí Visual Studia [15].

Použitím stejného programovacího jazyka, API a datových struktur v programu ušetří podle společnosti Xamarin v průměru 75% aplikačního kódu napříč platformami. Pokud využijeme Xamarin.Forms využití jednoho kódu mezi všemi platformami je 100%.



Obrázek 2.15: Xamarin – Shared Project [4]

Shared Project slouží pro sdílení kódu mezi platformami. Společný zdrojový kód, obrázky a další multimediální soubory jsou automaticky vloženy do všech cílových platforem.

Krom sdílení zdrojového kódu a souborů je možno používat celé knihovny. Podporovány jsou PCL knihovny jako například SQLite, Json.NET nebo ReactiveUI [16].

Xamarin podporuje vývoj Windows aplikací pro projekty [17]:

- **Windows 8.1**
  Umožňuje vytvořit aplikace pro desktopy a tablety pomocí WinRT ovládacích prvků
- Windows Phone 8.1
  Starší řešení pomocí Silverlight je v nové verzi Xamarinu předělané na WinRT.
- Universal Windows Platform (UWP)
  Nejnovější typ Windows aplikací umožňuje běh na různých zařízení. Jedna aplikace může své vizuální rozhraní přizpůsobovat pro běh na telefonu, tabletu i počítači. Tento projekt je ve strany Xamarinu pouze ve fázi vývoje.

Při přechodu z VCL na Xamarin by se musel stávající projekt v Delphi zachovat pro část aplikačního serveru. Všichni klienti by byli generováni z jednoho zdrojového kódu a komunikace by byla zajištěna pouze pomocí aplikačního serveru. Musel by být ale vytvořen jednoduchý Delphi klient pro tvorbu speciálních skriptů a sestav. Tvorba speciálních úprav v jazyce C# a linkování Delphi funkcí z jádra IS K2 je příliš složitá.

2.4.3.1 Výhody

- **Multiplatformní vývoj**
  Aplikace napsané v Xamarinu je možno zkompilovat pro platformy Android, iOS i Windows. Na všech využívají přímo nativní prvky uživatelského rozhraní. Je tím zajištěno stejné chování i uživatelský požitek. Možnosti vývoje pro všechny platformy jsou popsány v tabulce 2.3.

  Aplikace je možné vyvíjet 2 způsoby:

  - **Xamarin.Forms**
    Slouží pro vytvoření univerzálních aplikací, které nepotřebují využívat specifické funkcionality jednotlivých platforem. Například jednoduché aplikace pro zadávání a čtení dat. U těchto aplikací je sdílení kódu nadřazeno specifickému uživatelskému rozhraní. Hodí se také například při vytváření prototypů aplikací a „proofs-of-concept“. K dispozici je více než 40 ovládacích prvků a rozložení.

– **Xamarin.iOS**, **Xamarin.Android** a **Xamarin.Mac**
Tyto 3 části Xamarinu slouží pro vytvoření aplikací vyžadujících platformě specifické interakce. Poskytují pohodlnější vývoj při využití více specifických API. Zde je nadřazeno uživatelské rozhraní oproti sdílení kódu aplikace.

	Mac OS X	**Windows**	
Vývojové prostředí	Xamarin Studio	Visula Studio	Xamarin Studio
Xamarin.iOS	Ano	Ano (s počítačem Mac)	Ne
Xamarin.Android	Ano	Ano	Ano
Xamarin.Forms	iOS, Android	Android, Windows (iOS pouze s počítačem Mac)	Android
Xamarin.Mac	Ano	Ne	Ne

Tabulka 2.3: Xamarin – Vývojová prostředí

- **Přístup k API**
Přístup k API jednotlivých platforem není omezen. Aplikace mají přístup ke všem rozhraní a dokáží využít i specifické funkčnosti jednotlivých platforem, jako například technologii iBeacon na iOS nebo fragmenty na Androidu.

- **Výkon**
Překlad aplikace do nativního kódu probíhá při kompilaci. Oproti řešení s interpretováním kódu za běhu vykazuje mnohem lepší výkon a plynulost i náročnějších aplikací.

- **Souběh nového rozhraní**
Projekt všech klientů by mohl běžet paralelně se stávajícím rozhraní. Podmínkou by bylo přesunutí veškeré aplikační logiky z VCL formulářů do datových modulů. Tato funkčnost by pak mohla být volána přes AS na přímo, tak pomocí REST rozhraní serveru webových služeb. Pokud by se využil pouze přístup pomocí SWS, byl by kód jednotný jak pro desktopového Windows 10 klienta, tak pro všechny mobilní klienty.

2.4.3.2 Nevýhody

- **Nekompatibilita**
Stejně jako při přechodu na FireMonkey 2.4.1, ale i na jakoukoliv jinou technologii, i zde je problém se stávajícím aplikačním kódem obsaženým ve formulářích. Dále zde vzniká problém s roztříštěností informačního

systému na 2 části. Serverová část napsaná v Delphi a k ní nástroj na tvorbu speciálních skriptů, formulářů a sestav. Druhá část by byla napsaná pomocí C# v Xamarinu a sloužila by pouze pro prezentační vrstvu na všech platformách. Běžný uživatel ani konzultant by nemohl do této vrstvy přistupovat a cokoliv v ní měnit.

- **Desktopové platformy**
  Xamarin také nemá dostatečnou podporu pro desktopové Windows aplikace. Uživatelé IS K2 stále používají systémy Windows 7, někteří dokonce i Windows XP. Vytvoření UWP aplikace pro systém Windows 10 je dobré řešení pro budoucí vývoj IS K2, ale v současnosti není tato část Xamarinu dokončena.

  Rozšiřování na další desktopové platformy není podmínkou ale výhodou do budoucna. Aplikace pro systém Mac OS X a iOS je možno v Xamarin Studiu vyvíjet pouze na počítači s Mac OS X, nebo pokud je z počítač s tímto systémem dostupný po síti. Pak je možné z Visual Studia vzdáleně kompilovat a ladit kód pro iOS. Aplikace pro systém Linux není možné jednoduše vytvořit pomocí Xamarinu.

- **Jazyk C#** V současné době je většina vývojářů IS K2 programuje v jazyce Delphi. Pouze část starající se o současného Silverlight klienta ovládá jazyk C#. Při přechodu prezentační vrstvy na Xamarin by část stávajících zaměstnanců musela jazyk C# a související knihovny naučit, nebo by muselo dojít k náborů nových programátorů.

- **Budoucnost**
  Společnost Xamarin vznikla v roce 2011. Její nástroje využívá mnoho programátorů [15], ale do budoucna zde existuje možnost zaniknutí celé platformy. Takový osud postihlo více nových IT technologií. Jistou důvěru do pokračování projektu vnesl Microsoft odkoupením celé firmy.

2.4.4 HTML5

Značkovací jazyk HTML se v roce 2014 dočkal finální specifikace verze 5. Oproti verzi 4 z roku 1997 přináší spoustu vylepšení pro práci s multimédii, podporu aplikací běžící v prohlížeči bez připojení k internetu nebo lepší práci s asynchronními událostmi na pozadí webové stránky [18].

Technologie HTML5 je dnes podporovaná ve všech moderních webových prohlížečích na všech platformách. Odpadá tedy nutnost vyvíjet klienty pro všechny platformy a zařízení. Jako HTML5 se označuje celý balík i s technologií CSS3, umožňující tvorbu responzivních webů a JavaScriptu pro tvorbu aplikační logiky na straně klienta. Responzivní weby se dokáží dynamicky přizpůsobovat zařízení podle velikosti obrazovky, jejího rozlišení a také orientace na mobilních zařízení. Pro zajištění zobrazení správného obsahu pro dané za-

řízení se používá „Media Queries“. Ty vnášení do CSS jazyka podmínky na základě vlastností zařízení. Zápis probíhá pomocí tagu *@media* a podmínky:

```
@media not | only mediatype and (media feature) {
CSS–Code;
}
```

Tento zápis umožňuje definovat různé vlastnosti pro mobilní zobrazení (základní styl) a pro tablety a desktopy (pomocí *@media*)

```
.menu{
        width: 100%
}

@media only screen and (min-width: 600px) {
        .menu {
                width: 20%;
        }
}
```

Technologie HTML5 poskytuje rozhraní pro takzvané SSE (Server-Send Events). Standardní komunikace mezi klientem (např. ve formě webového prohlížeče) a serverem je vždy inicializovaná na straně klienta. Ten otevře HTTP spojení se serverem a vyžádá si vygenerování požadované stránky a připojených souborů. Po načtení je spojení ukončeno a server sám od sebe nemůže jakkoliv komunikovat s klientem. Pokud potřebujeme při změně obsahu na serveru aktualizovat data všech klientů, využívá se SSE. V kódu webové aplikace je vytvořen objekt *EventSource*, který odchytává události souboru zadaném pomocí URL jako parametr. Vždy když server odešle data obsahující v hlavičce typ *text/event-stream*, je v klientu zavolána událost *onMessage* pro zpracování dat [19].

Pro komplexní webové aplikace přináší HTML5 podporu pro úložiště na straně webového prohlížeče. V předchozích verzích bylo nutné uživatelská data ukládat do cookies na straně serveru a ty při každém požadavku odesílat klientovy. Nové úložiště na straně webového prohlížeče umožní ukládat jak persistentní tak dočasné relační data. Jsou uložena do asociativního pole s maximální velikostí 5MB [20]. Krom jednoduchého asociativního pole je možné využít i SQLite databázi pro složitější struktura dat. Ta podporuje vykonávání SQL dotazů i transakční zpracování.

Webového klienta na bázi HTML5 nabízí například i Microsoft se svým Dynamics GP. Nová verze bude představena v květnu 2016 a poskytne plnohodnotného klienta tohoto ERP systému. Konkurenční společnost Microsoft tím řeší problém s dostupností svého řešení na systémech Mac OS X a Linux. HTML5 klient Dynamics GP bude sloužit pouze jako prezentační vrstva formulářů, náročné výpočty a práce s daty bude stále probíhat pouze na serveru [21].

Vytvoření prezentační vrstvy pomocí technologie HTML5 by stejně jako u ostatních řešení vyžadovalo přesunu aplikačního kódu z VCL formulářů do datových modulů. Celé řešení by komunikovalo pomocí REST rozhraní SWS. Pro implementaci se nabízí 2 řešení.

První varianta by znamenala kompletní přechod prezentační vrstvy na HTML5. Z desktopu i mobilních telefonů by uživatelé využívali pouze tohoto webového klienta. Ve stávajících Delphi zdrojích by se stejně jako u ostatních řešení, které nejsou postaveny na tomto jazyce, musela vytvořit aplikace pro vytváření speciálních skriptů a sestav. Vytváření datových modulů pomocí HTML klienta není možné. Definice formulářů by mohla být uložena v databázi a načítala by se stejně jako datové moduly. HTML5 technologie poskytuje drag&drop technologii, která by se mohla použít v novém editorů formulářů.

Druhá varianta by znamenala zachování zachování současného plného klienta pro desktop, do kterého by byla integrovaná nová vrstva pro definici formulářů pomocí dat. Všechny ostatní klienti by přistupovali pomocí plnohodnotné HTML5 aplikace. Ta by byla vytvořena samostatně jako oddělený projekt. Poskytovala by pouze prezentační vrstvu pro data. Aplikace by umožňovala funkčnost nezbytnou pro běžné uživatele IS K2. Správa a další systémové věci by byly dostupné pouze v plném Delphi klientu.

Při vývoji tohoto klienta by mohly být využity dostupné knihovny. Pro celkový vzhled webu a jeho responzivní vlastnosti lze využít populární Bootstrap framework od společnosti Twitter. Ten využívá pro zobrazení mřížkový systém. Celá stránka je rozdělena na řádky, a každý řádek obsahuje 12 sloupců. Tento systém je dostupný pomocí specifických CSS tříd přiřazených do * * elementů.

Základní ovládací prvek IS K2 je bezesporu datový grid. Pro HTML5 je dostupno několik připravených knihoven. Například Data Grid od DevExtreme. Poskytuje optimalizované ovládání jak pro normální tak i dotykové obrazovky. Umožňuje „nekonečné“ rolování obsahem s postupným načítáním dat. Ve sloupcích je možné filtrovat a vyhledávat informace a při editaci jednotlivých záznamů umožňuje validaci hodnot [22].

2.4.4.1 Výhody

- **Společný zdrojový kód**
  Pro vývoj celé prezentační vrstvy všech klientů by byl použit pouze jeden programovací jazyk v jednom zdrojovém projektu. Oprava chyb, vývoj nových vlastností a vylepšení by byl mnohem rychlejší. Při vykreslování HTML5 ve webovém prohlížeči se všechny komponenty chovají stejně, nebylo by tedy nutné dělat výjimky pro různé platformy a zařízení. Dynamické zobrazení podle velikosti obrazovky a dalších vlastností zařízení je možné definovat na úrovní CSS a do vlastního HTML kódu nezasahuje.

- **Offline režim**
  Pokud by se IS K2 dokázal přizpůsobit offline využívání klienta, umožnil by zpřístupnit jednodušší logiku části systému i bez připojení k serveru.

  Spoustu standardních skriptů i speciálních úprav by mohlo fungovat z části i jako samostatný celek. Například skladník by mohl spustit webového klienta na mobilním telefonu nebo čtečce čárových kódů, ve skladu načíst informace o zboží pomocí čárového kódu, zadat množství a následně při opětovném připojení k internetu, potažmo serveru K2 vložit všechny položky do modulu inventur.

- **Bez instalace**
  U značkovacího jazyka HTML a jeho interpretované aplikační logiky v JavaScriptu odpadá nutnost jakékoliv instalace aplikace, nebo její aktualizace. Dnes už prakticky všechny zařízení umožňující připojení k internetu obsahují webový prohlížeč. Nasazení informačního systému na zařízení by tedy pouze obnášelo zprovoznění serveru webových služeb K2 a samotné HTML5 aplikace na webovém serveru.

2.4.4.2 Nevýhody

- **Rychlost**
  HTML jakožto značkovací jazyk je při požadavku na zobrazení stránky na serveru vygenerován a následně celý vykreslen ve webovém prohlížeči. Samotné vykreslení elementů a zpracování JavaScriptového kódu má na starosti jádro webového prohlížeče. To se rychlostí nemůže vyrovnat nativním kompilovaným aplikacím. Další zpomalení způsobuje samotné stahování všech částí webové aplikace. Současné protokoly fungují na principu postupného stahování. Při zavolání požadavku na stažení webové stránky do prohlížeče je odeslán soubor hlavní stránky a až následně jsou podle potřeby zavolány požadavky pro připojené CSS a JS soubory. To samotné načítání prodlužuje.

- **Kompatibilita**
  Nová verze jazyka HTML s číslem 5 byla ve finální verzi schválena teprve v roce 2014. Proto stále ještě nejsou ve všech majoritních prohlížečích všechny nové atributy a funkce podporované. Tabulka 2.4 ukazuje procentní podporu HTML5 prvků v prohlížečích.

  Nejlepší podporu všech technologií má webový prohlížeč Google Chrome. Ten je dostupný na všech platformách. Při použití většiny nových vlastností HTML5 v klientu IS K2 by bylo nutné používat webové prohlížeče Chrome nebo Firefox pro maximální využitelnost.

Prohlížeč	Podpora HTML5 prvků
Safari	62%
Chrome	93%
Firefox	89%
Internet Explorer	35%
Microsoft Edge	40%

Tabulka 2.4: HTML5 – Podpora v prohlížečích [6]

Kapitola 3

Návrh

Zhodnocením všech výhod a nevýhod vychází jako nejlepší řešení zachování současné VCL knihovny pro prezentační vrstvu a vytvoření nové vrstvy pro definici formulářů. Z ní se budou generovat formuláře pro všechny platformy. Velkou výhodou je především zachování většiny dosavadního kódu prezentační vrstvy. Toto řešení bude využívat současné prvky uživatelského rozhraní jako je například editační pole *TxEdit* a grid *TxGrid* pro zobrazení seznamu položek datového modulu.

Velkou výhodou při zachování stávající VCL knihovny je možnost souběhu obou nových prezentačních vrstev. Datové moduly informačního systému mohou využívat stávající, pevně definované, formuláře v .DFM souborech i novou definici a z nich vygenerované formuláře. Bude to ale vyžadovat úpravu i ve stávajících formulářích. Z nich bude nutné přesunout aplikační logiku níže do datových modulů, které budou první společnou částí obou prezentačních vrstev viz. obrázek 2.14. Toto řešení zajistí postupný přechod na novou prezentační vrstvu jak z pohledu vývojového oddělení, které bude moci postupně předělávat jednotlivé staré formuláře do nového systému podle dostupných kapacit, tak z pohledů uživatelů, kteří nebudou muset ihned využívat novou prezentační vrstvu.

Místo použití některé dostupné knihovny a její následné přizpůsobení pro složitou strukturu informačního systému nabízí vlastní řešení jednodušší implementaci. Od začátku budou nové, datově popsané formuláře na míru přizpůsobené všem specifickým vlastnostem IS K2.

Do vrstvy definice formulářů bude možné implementovat mechanismus rozdílné definice pro různé úrovně a skupiny uživatelů. Tím se například zobrazí formulář zakázek jinak pro obchodní zástupce a jinak pro účetní oddělení. Bude zde možnost vytvoření buď zcela odlišného formuláře pro skupinu uživatelé, nebo pouze skrytí určitých částí již nadefinovaného formuláře.

Definice formulářů bude vytvořena a uložena ve standardním datovém modulu K2. Tím se zjednoduší publikování definice pomocí SWS. Současný webový klient, který používá pro čtení dat aplikační server, bude umět číst

definici bez úpravy v komunikaci. Změna proběhne pouze pro vizuální komponenty, které budou svoji strukturu načítat z tohoto datového modulu místo současných XAML souborů. Pro mobilní klienty budou muset být definovány jiné formuláře oproti desktopu. Velikost obrazovek není tak velká a je nutné myslet na pohodlnost celého ovládání na malé dotykové obrazovce.

Zachování VCL knihovny a vytvoření definiční vrstvy formulářů neřeší problém s webovým klientem. Podpora ze strany Microsoftu nebude dlouho trvat a i současní výrobci prohlížečů přestávají Silverlight plugin podporovat. Do současného klienta se tedy implementuje funkčnost pro čtení definice z IS K2, aby se zajistila stejná funkčnost napříč všemi klienty. Do budoucna by ale webový klient přešel na jazyk HTML5 a komunikaci pomocí webových služeb. Z analýzy 2.4.4 se jako řešení jeví využití například Bootstrap knihovny. Naprogramování responzivního webového HTML5 klienta by do budoucnosti mohlo nahradit jak současného webového klienta tak i mobilní klienty pro Android i iOS.

Nové formuláře budou univerzální napříč platformami a umožní jednoduše uživatelsky přizpůsobovat vizuální podobu informačního systému K2. Dále je budeme pojmenovávat jako **UniForm**.

3.1 Úprava současného řešení

Současné VCL formuláře musejí projít odstraněním aplikačního kódu z .DFM souborů a nahrazení alternativou přímo v datových modulech. Pokud datový modul projde touto úpravou, bude možné nad ním definovat nový formulář s identickou funkčností. Bude tedy možné zobrazit například modul Zboží jak pomocí současného formuláře, tak pomocí nověh, a pracovat s daty v něm stejným způsobem. Předělání se týká 2 hlavních funkčností, které jsou volány z VCL formulářů popsaných níže.

3.1.1 Akce

Datový modul, respektive formulář nad ním, musí poskytovat minimálně několik základních funkčností pro práci se záznamy. Jedná se o pohyb mezi záznamy, vytváření nového, vstup do editace a uložení. K těmto základním akcím si každý datový modul dodefinuje svoje vlastní akce pro specifickou funkčnost, kterou poskytuje.

Tyto akce budou součástí přímo datových modulů. Pokud v UniFormu budeme chtít zobrazit danou akci, použijeme pole **Command**, kde bude textový název akce datového modulu.

Mezi základní příkazy bude patřit:

- **ListCommand**
  Dojde k zobrazení první strany formuláře na které je zobrazen grid se záznamy.

- **NextCommand**
  Přepnutí na následující záznam datového modulu.
- **PreviousCommand**
  Přepnutí na předchozí záznam datového modulu.
- **:List\NewHeaderCommand**
  Příkaz pro inicializaci nového prázdného záznamu a přepnutí do editačního režimu.
- **CopyHeaderCommand**
  Vytvoření kopie aktuálního záznamu a přepnutí do editačního režimu.
- **:List\BrowseHeaderCommand**
  Přepnutí formuláře na první stranu detailu aktuálního záznamu
- **ChangeHeaderCommand**
  Přepnutí do editačního režimu.
- **RefreshCommand**
  Aktualizace všech zobrazených dat.

Význam zástupných slov :List a :Detail v příkazech je vysvětlen v sekci 3.2.3.

IS K2 umožňuje vytváření vlastních datových modulů, které se následně začlení do systému a chovají se stejně jako standardní. Pokud si bude uživatel definovat akci na vlastním modulu, budou mu přístupny následující proměnné a vlastnosti.

- Name – Název akce pro datový modul
- Caption – Název akce zobrazovaný pro uživatele
- Shortcut – Klávesová zkratka
- Icon – Ikona pro tlačítko
- Visibility – Volba zobrazení akce podle stavu formuláře (Vždy, Pouze prohlížení, Pouze změna)
- Context – Volba dostupnosti akce z pohledu kontextu dané komponenty (Vždy, :List, :Detail, konkrétní pole)
- Umístění – Povolení na kterých místech formuláře se bude akce zobrazovat (Panel nástrojů, Panel tlačítek, Hlavní nabídka, Místní nabídka)
- Execute – Vlastní aplikační kód akce

Tím zajistíme základní funkčnost akcí a vložíme aplikační kód, který má daná akce provést. Z těchto vlastností můžeme následně generovat tlačítka do starých formulářů, která budou identická jak vizuálně tak funkčně s původními akcemi přímo ve formulářích.

3.1.2 Přiřazení hodnoty

Informační systém K2 ve velké míře využívá událost *OnExit* na komponentě editačního pole. V této události se volá aplikační kód, který reaguje na změnu hodnoty. Při změně bývají například načteny informace do ostatních polí datového modulu. Tato funkčnost musí být přesunuta do datového modulu, aby byla přístupné i pro novou prezentační vrstvu. Pro uživatele současných formulářů musí být funkčnost stále stejná.

Při zapsání hodnoty do editačního pole formuláře (*TxEdit*) se při opuštění této vizuální komponenty kopíruje hodnota do pole *TDataField* v datovém modulu. Zapisování je prováděno pomocí property *AsString*, *AsLong*, *AsBool* atd.

Pro možnost kontrolovat hodnotu a reagovat na ní vznikne nová property **AsControlValue**. Ta bude typu *Variant*, aby šla zapisovat jakákoliv hodnota. Při zapisování hodnoty se zavolá metoda datového modulu *ControlValueChanged*, ta se v současné době volá při každém zmáčknutí klávesy v editačním poli. Této metodě bude předána aktuální instance *TDataField* a stará hodnota pole v *AOldValue*. Na základě datového pole se zavolá příslušná kontrola nebo jiný aplikační kód.

Pro stávající VCL formuláře dojde jen k přesunutí z události formuláře *OnExit* do metody *ControlValueChanged* datového modulu. Z pohledu uživatele se funkčnost nezmění. Z pohledu programátora je nutné pro nové moduly využívat pouze tuto metodu reakce na změnu. Tím bude možné vytvořit nové UniForm formuláře se stejnou funkčností.

3.2 UniForm formuláře

Nová prezentační vrstva IS K2 se dá rozdělit na 2 hlavní části. První se bude věnovat nové mezivrstvě v podobě datové definice rozložení jednotlivých komponent ve formuláři a druhá bude následně z této definice vykreslovat samotné komponenty uživatelského rozhraní desktopového klienta K2.

3.2.1 Definice rozložení

Pro ukládání definic nových UniForm formulářů se využije standardní datový modul *TDataM*, respektive zděděná třída *TD_Fragment*. Jeden záznam bude odpovídat jedné části formuláře kterou budeme pojmenovávat fragment. Bude možné využít tyto typy:

- **Label**
  Textový popisek, umožňuje zobrazit jak název, tak needitovatelnou hodnotu pole v jedné komponentě

- **Input**
  Editační pole pro zadávání a zobrazování hodnoty datového pole. Umožní zobrazit odkaz do jiného datového modulu pomocí vazby, nebo například komponentu kalendáře.

- **CheckBox**
  Zatrhávající komponenta pro boolean pole datového modulu s popiskem

- **Grid**
  Zobrazení seznamu záznamů datového modulu. Slouží pro výběr, filtrování a úpravu jednotlivých záznamů.

- **Panel**
  Vizuální panel sloužící jako kontejner pro další panely, nebo koncové komponenty typu *Label*, *Input* nebo například *Grid*

- **Expander**
  Komponenta vycházející z klasického panelu. Přidává ale horní lištu, která zobrazuje popisek a umožňuje celý panel zabalit do této lišty

- **Tab Control**
  Přepínač jednotlivých stránek formuláře. Do něj se vnořují jednotlivé panely, pro které jsou v levé části na liště zobrazeny tlačítka pro přepínání. Na tlačítku se zobrazuje popisek a ikona přiřazená k panelu

- **Splitter**
  Posuvník pro rozdělení panelu nebo hlavního formuláře na 2 části. Do obou částí se vloží panely nebo například tab control. Posuvníkem v mezilehlé části je možné upravovat poměr velikostí obou polovin.

- **Toolbar**
  Panel pro zobrazení tlačítek akcí. Bude nahrazovat záhlaví klasických VCL formulářů, kde jsou umístěny tlačítka ovládání datového modulu.

Toto je seznam základních komponent, ke kterému se následně mohou vytvořit specializovanější komponenty pro zobrazování adresy, mapy, obrázku nebo dashboardu. Základ každého formuláře bude sloužit komponenta *TFrgtForm* která nepůjde vložit do žádného dalšího fragmentu.

Fragment bude následně mít definovaný stav formuláře, pro který je uložen.

- Prohlížení

- Nový

- Změna

Pokud chceme definovat UniForm pro zobrazení, musíme vytvořit komponentu alespoň pro stav Prohlížení. Ostatní stavy jsou volitelné a více popsány v sekci 3.2.5.

Jelikož bude nové rozhraní podporovat jak desktop tak mobilní platformy, bude základním rozlišovacím parametrem počet sloupců. To nám umožní definovat rozložení pro různé velikosti obrazovek. Pro desktop bude standardní počet sloupců na šířku 15. Pro tablet v zobrazení na šířku se definuje počet sloupců 10. Tím se zajistí dostatečná velikost ovládacích prvků v poměru k velikosti obrazovky.

Formulář také v sobě musí mít určeno, pro jaký datový modul je určen. Standardní datové moduly a uživatelské moduly vytvořené pomocí návrháře objektů jsou zaregistrovány v interní knihovně K2 a každý má svoje unikátní číslo. Přes to se budou jednotlivé fragmenty odkazovat na modul.

Nastavení konkrétního fragmentu bude serializované z objektů K2 do XML a uloženo do databáze. Předek všech fragmentů bude mít tuto strukturu:

- **Actions**
  Seznam akcí dostupných pro daný fragment. Tyto akce jsou například na gridu dostupné v kontextovém menu. Popsány jsou v sekci 3.2.4

- **Align**
  Zarovnání prvku v rámci nadřízeného fragmentu. Na výběr bude těchto 5 konstant, které vycházejí ze stylu zarovnání ve VCL komponentách:

  - ufaLeft
  - ufaRight
  - ufaTop
  - ufaBottom
  - ufaClient

- **Bounds**
  Struktura obsahující pozici daného fragmentu v rámci nadřazeného. Obsahuje počet sloupců od horního a levého okraje a podle hodnoty šířky dopočítá hodnoty od pravého a spodního okraje.

- **DataContext**
  Obsahuje název pole, popřípadě zástupnou zkratku, ze které se budou číst data pro daný fragment. Více vysvětleno v sekci 3.2.3.

- **Fragment type**
  Identifikace typu fragmentu, jednotlivé typy jsou popsány výše.

- **Height**
  Určuje výšku fragmentu. Při nevyplnění je výška sama určena podle minimální výšky u koncových fragmentů, popřípadě podle vnitřního obsahu fragmentu u panelů.

- **Icon**
  Zkratka ikony, která je zobrazena na tlačítku ve fragmentu tab control nebo na tlačítku akce. K2 obsahuje seznam svg ikon, které jsou obsaženy v instalaci a ze kterých je možno vybírat.

- **Name**
  Slouží pro pojmenování fragmentu pro uživatele.

- **StringId**
  Textový identifikátor fragmentu. Uživatelsky definované fragmenty budou mít tento identifikátor složen z názvu třídy fragmenty a unikátního identifikátoru záznamu v databázi. Pře redefinování standardního formuláře K2 zůstane tato hodnota stejná. Využije se pro sestavení formuláře popsaném v sekci 3.2.5

- **Title**
  Popisek daného fragmentu. Je využíván u tlačítek akcí, názvů panelů v tab control, nebo jako popisek expanderu.

- **Unique**
  Jedná se o unikátní náhodný identifikátor každého vytvořeného fragmentu uloženého pomocí 128-bitového GUID čísla.

- **Width**
  Počet sloupců, které má komponenta obsadit do šířky.

Do XML formátu se serializují pouze vyplněné property objektů v K2. Zde je ukázka, jak bude vypadat uživatelský input pro datové pole CDo (číslo dodavatele) s šířkou přes 3 sloupce:

```
<object class="TFrgtInput">
  <Unique>{E30339E1-81C5-4982-A070-AE860B75DF5A}</Unique>
  <DataContext>CDo</DataContext>
  <Width>3</Width>
</object>
```

Ukázka uživatelského panelu:

```
<object class="TFrgtPanel">
  <Unique>{ED2AA8D1-77FF-48CD-AF90-CAC987CD3DAE}</Unique>
  <Title>Detail</Title>
  <Icon>List</Icon>
  <DataContext>:Detail</DataContext>
```

```
  <Width>12</Width>
  <Name>ZakOtherPanel</Name>
  <Align>ufaClient</Align>
</object>
```

Jako požadavek prezentační vrstvy byla možnost uživatelské definice formulářů. Proto je nutné odlišit, které fragmenty jsou vytvořené výrobcem IS K2, a které uživatelsky definované. Pro rozlišení jednotlivých úrovní vyjdeme ze současné funkčnosti serializace objektů v K2, které také rozlišuje uložení do několika úrovní.

Budou definovány tyto úrovně:

- **Tovární**
  Fragment definovaný výrobcem IS K2 obsažený v každé instalaci programu.

- **Instalace**
  Uživatelsky definovaný fragment, který je možné použít napříč všemi firmami v konkrétní instalaci.

- **Mandant**
  Uživatelsky definovaný fragment, určený pro konkrétní firmu v dané instalaci programu. Firma je identifikovaná podle své zkratky.

- **Skupina**
  Fragment pro skupinu kontaktních osob. Ta se přiřazuje přímo na kontaktní osobě. Je zde uložené číslo z datového modulu *TD_GroupContPerson*

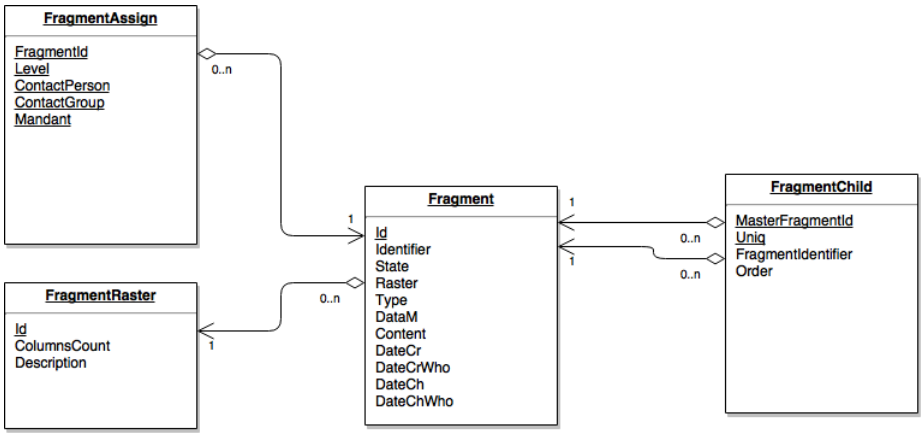
- **Uživatel**
  Fragment, který je přiřazený konkrétnímu uživateli aplikace IS K2.

Jeden fragment může být přiřazen více úrovním, proto je toto přiřazení uložené v samostatném datovém modulu a připojeno jako položky modul na fragment. Datová struktura je popsaná níže.

Krom samotného vytváření bude uživateli umožněno skrývat fragmenty na formulářích. To bude využíváno zejména nad formuláři, které budou definované od výrobce IS K2. Tato funkčnost bude oddělena od modifikace a tvorby formulářů. Každý uživatel bude mít nad UniForm formulářem možnost spustit režim potlačení fragmentů. Tento režim nebude na rozdíl od modifikace záviset na uživatelském právu „Modifikace Formuláře“. Při vstupu do tohoto režimu se fragment při kliknutí označí pro skrytí. Takto označené fragmenty se podle unikátního identifikátoru uloží do nového objektu *TSuppresItem* a serializované se uloží do stávajícího uživatelského nastavení K2. Následně se při sestavování formuláře (popsané v sekci 3.2.5) toto potlačení fragmentu zohlední a uživateli se nevykreslí.

3.2.2 Datová struktura

Datová struktura byla naznačena v předchozích odstavcích. Základ tvoří tabulka *Fragment*, kde jsou uloženy hlavní informace o jednotlivých částech formuláře. Přiřazení fragmentů do jednotlivých úrovní zajišťuje tabulka *FragmentAssign*. Vazby mezi jednotlivými fragmenty ve stromu formuláře jsou uloženy v *FragmentChild*. Poslední tabulka je *FragmentRaster*, kde jsou definované počty sloupců pro různé zařízení a velikosti obrazovek.



Obrázek 3.1: ER–diagram

Tabulka *Fragment*, nad kterou je postaven datový modul *TD_Fragment*, obsahuje tyto pole:

- Id
  Číselní primární klíč tabulky
- Identifier
  Textový identifikátor prvku, není unikátní. Jeho význam je dále popsán v sekci 3.2.5
- State
  Číselná hodnota označující stav formuláře (1 – Prohlížení, 2 – Nový, 3 – Změna)
- Raster
  Odkaz do tabulky *FragmentRaster*, kde je definován počet sloupců formuláře
- Type
  Odkaz na typ použitého fragmentu. Názvy fragmentů popsané v sekci 3.2.1 budou uloženy v universální tabulce *Texty* a zde bude odkaz na jejich číselné ID

- DataM
  Číslo datového modulu K2, nad kterým je fragment definován
- Content
  Instance konkrétního fragmentu serializovaná do XML streamu.
- DateCr
  Datum vytvoření
- DateCrWho
  Uživatel K2, který záznam vytvořil
- DateCh
  Datum změny
- DateChWho
  Uživatel, který provedl poslední změnu

Poslední 4 pole jsou určeny pro potřeby K2 a obsahuje je každá hlavičková tabulka K2 nad datovým modulem.

3.2.3 DataContext

DataContext určuje namapování fragmentu na pole nebo část datového modulu. Krom názvu polí datového modulu je možno zadat i 2 speciální identifikátory.

- **:List**
  Slouží jako identifikátor pro seznam položek datového modulu, nad kterým je formulář spuštěn. Využívat se bude především nad hlavním gridem pro pohyb mezi záznamy.
- **:Detail**
  Přiřazení tohoto identifikátoru definujeme pro fragment, aby pracoval nad aktuální položkou datového modulu. Pokud zadáme tento identifikátor na fragment panelu, pole do něj vnořená budou číst hodnoty přímo z aktuálního záznamu datového modulu. Pro prvky vložené do fragmentu s identifikátorem kontextu :Detail jsou přístupná všechna pole datového modulu. Například pro panel ve formuláři zakázek bude pole s hodnotou CDo v DataContextu namapované na *TD_Zak.CDo*.

Krom těchto dvou identifikátorů se do tohoto pole především zapisují programové názvy polí datového modulu. Podle textového názvu je následně možné vyhledat příslušnou hodnotu *Progname* v meta informacích datového modulu. Zde jsou obsaženy informace o datovém typu pole a jeho popisu. Po položkové pole je zde uvedena i třída datového modulu položek. Stejně

tak je tomu i u vazebních polí. Ty ještě navíc obsahují název výchozího pole vazebního modulu.

Pokud se do input fragmentu zadá vazební pole, prezentační vrstva si z meta informací zjistí přesný typ a podle toho může dynamicky zobrazit ikonu pro lookup. Ve výchozím stavu je u vazebních polí zobrazena hodnota výchozího pole připojeného datového modulu. Například datový modul odběratelů je se zakázkou svázané pomocí číselného identifikátoru (CDo), ale podle meta dat je zobrazeno výchozí pole se zkratkou firmy.

Pokud budeme chtít ve vazebním poli zobrazit jiné než výchozí pole, využijeme spojování názvu polí pomocí tečkové notace. Tím se odkážeme na jiní pole výstupního datového modulu. Pokud tedy chceme na zakázce pro vazební pole odběratele zobrazit místo zkratky firmy její název, použijeme zápis:

```
CDo.Firm
```

3.2.4 Akce

Kromě standardních akcí obecného datového modulu a UniFormů, budou také realizovány uživatelské a programátorské akce specifických zděděných modulů.

Datový modul podporuje základní akce pro práci s daty pomocí příkazů popsaných v sekci 3.1.1. Zde je i popsán systém přidávání akcí do vlastních datových modulů.

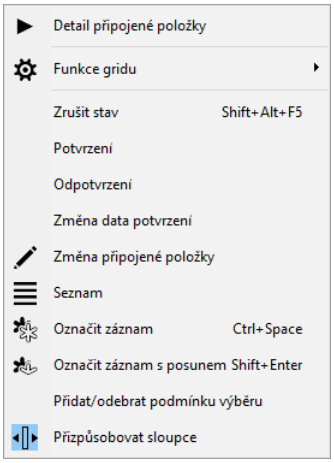
Krom datových akcí budou přítomné i akce jednotlivých komponent. Grid bude pomocí akcí definovat výběr, ukládání a resetování zobrazených sloupců, exportování do Excelu a přepínání mezi řádkovou editací. Jedná se o hlavní akce ze současného uživatelského rozhraní gridu.

Jednotlivé akce mají definované, za jakých okolností se budou zobrazovat. Je určeno jestli vždy, pouze v prohlížení, pouze ve změně a v jakém datovém kontextu. Nastavení ve vlastním datovém modulu je popsané v 3.1.1.

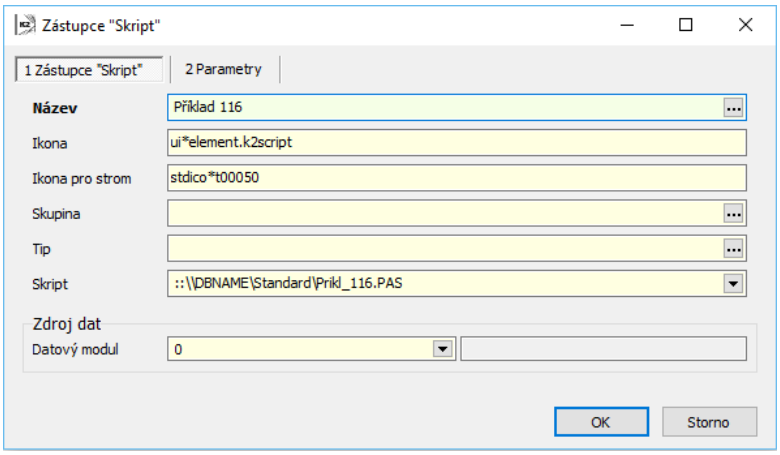
Všechny datové akce se budou podle svých vlastností zobrazovat u jednotlivých komponent společně s jejich formulářovými akcemi. Vyvolat je bude možné buď pravím tlačítkem myši, nebo kliknutím na ikonu menu v záhlaví komponenty. Nad gridem tedy budou dostupné akce jak pro ovládání gridu a jeho sloupců, tak i datové umožňující například potvrdit, připojit nebo odpojit položku.

Uživatel také může do akcí přiřadit svoji vlastní. Buď nadefinuje spuštění nadefinovaného příkazu a nebo přidá akci na spuštění skriptu popřípadě sestavy.

Definování spuštění skriptu nebo sestavy proběhne podobně jakou současné přidávání zástupců na plochu K2 nebo do automatických úloh. U akce se nadefinuje její název a ikona a přiřadí se cesta k standardnímu nebo speciálnímu skriptu jak ze souborového úložiště tak z databáze. Po uložení do fragmentu se následně zobrazí tlačítko s textem a ikonou v kontextovém menu nebo přímo záhlaví komponenty.

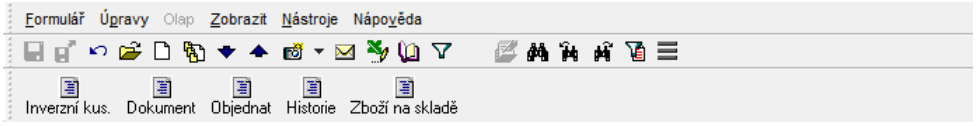


Obrázek 3.2: Zobrazení akcí v kontextovém menu



Obrázek 3.3: Odkaz na skript

Akce bude také možné vkládat do fragmentu *Toolbar*. Ten bude sloužit výhradně pro jejich zobrazení. Bude v přehlednější variantě nahrazovat současný panel akcí. Strukturou bude fragment *Toolbar* vycházet z panelu použitém ve webovém klientovy.



Obrázek 3.4: Toolbar současného plného klienta

Jak je vidět na panelu akcí webového klienta, převzetím jeho rozložení

zobrazení dojde k velkému zpřehlednění celého rozhraní. Méně používané akce nebudou mít nastavenou viditelnost na panelu ale pouze v menu pod tlačítkem „Možnosti“.

Obrázek 3.5: Toolbar webového klienta

3.2.5 Sestavení formuláře

Jelikož budou jednotlivé fragmenty definované na různých úrovních a pro různé stavy, je nutné před jejich zobrazením najít správné rozložení pro konkrétní situaci.

Funkce pro sestavení dostane 3 parametry:

- Datový modul – Název třídy nebo alias datového modulu (např. TD_Zak, Zak)
- Požadovaná velikost – Počet sloupců pro optimální zobrazení na dané velikosti obrazovky
- Stav formuláře – Jestli se jedná o prohlížení, změnu nebo vytváření dat

Při zavolání funkce dojde k vyhledání všech záznamů v tabulce *Fragment*, které mají:

- Identifier = %datamodule+'HeaderViewModel'
- Raster.ColumnsCount <= %size

Funkce má dostupné hodnoty aktuálního mandanta a pro uživatele jeho číslo, popřípadě zařazení do skupiny. Pokud je tedy vrácen více než jeden záznam pro hlavní fragment formuláře, dojde k průchodu a vyhledání záznamu. Ty jsou seřazeny podle počtu sloupců sestupně a podle pole *Level* (z připojené tabulky *FragmentAssign*) v tomto pořadí:

1. Konkrétní uživatel
2. Skupina uživatelů
3. Mandant
4. Instalace
5. Tovární

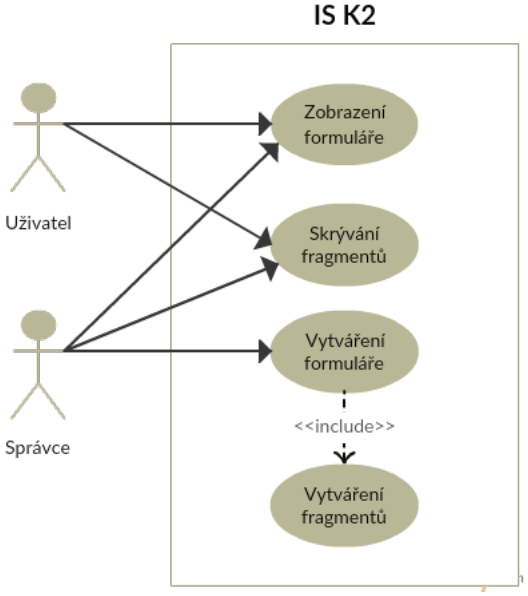
Tím je nalezen nejvhodnější formulář pro daného uživatele a velikost obrazovky. Přečteme Id fragmentu *TFrgtForm* a může začít sestavování podřízených fragmentů.

Z tabulky *FragmentChild* se vyberou záznamy s *MasterFragmentId* rovnému získanému Id a pro všechny nalezené položky se pomocí textového *FragmentIdentifier* vyhledá záznam v tabulce fragmentů. Pokud je nalezeno více fragmentů s tímto textovým identifikátorem, postupuje se stejně jako při hledání hlavního fragmentu formuláře výše. Ze získaného Id nejvhodnějšího fragmentu se rekurzivně naleznou další jeho potomci. Vyhledávání skončí pokud není nalezen žádný záznam v tabulce *FragmentChild*.

Takto funguje vyhledávání pro formulář ve stavu prohlížení. Pokud je požadován formulář pro změnu nebo nový záznam, je zohledněné pole *State* v tabulce *Fragment*. Při hledání formuláře pro stav „Nový“ se hledá v tomto pořadí:

3. Nový
2. Změna
1. Prohlížení

Z takto získaného stromu fragmentů se přečtou XML data z polí *Content* a sestaví se z nich strom objektů, podle kterého se buďto vygeneruje formulář plného desktopového klienta, nebo se pomocí aplikačního serveru serializuje a předá některému z tenkých klientů.



Obrázek 3.6: Use Case

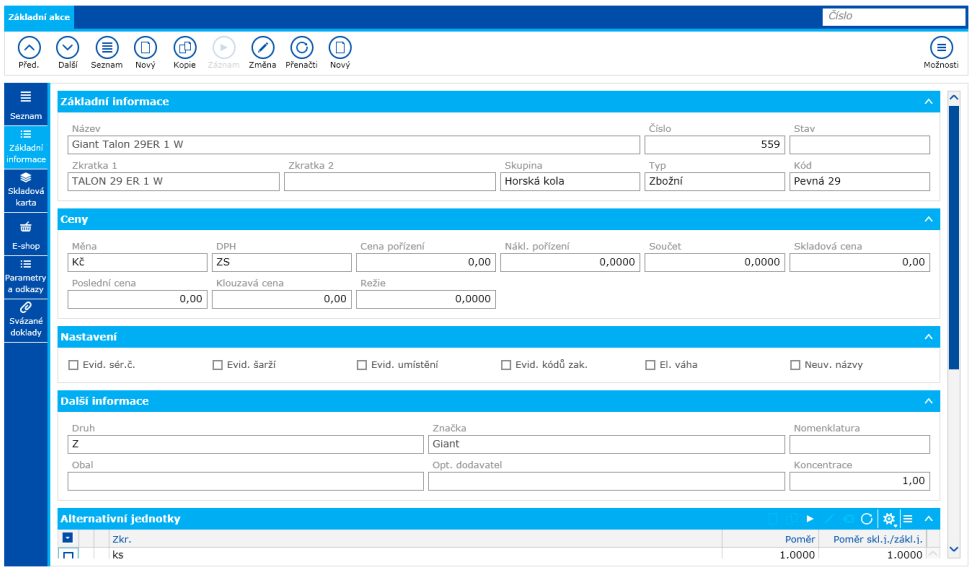
3.3 Nové UI

Nové uživatelské rozhraní plného klienta K2 převezme vzhled z webového klienta. Celkově se zmenší počet zobrazených vizuálních komponent jako jsou například různá ovládací tlačítka, a zbylé komponenty se zvětší. Méně používané funkce se přesunou do nabídek pod jednu ikonu menu.

Přepínání záložek, jakožto základní prvek většiny formulářů, se přesune z horní horizontální pozice do levé vertikální. To lépe využívá prostor dnešních širokoúhlých obrazovek. Přesunem na stranu se může zvětšit velikost jednotlivých „oušek“ záložek a zlepšit tím uživatelskou přívětivost.

Nový typ expander umožní shlukovat komponenty na panelech do logických celků a tím zvětší přehlednost celého rozhraní. Pokud uživatel nebude chtít daný celek vidět, sbalí expander a ušetří tím místo pro ostatní komponenty.

Celkově se všechny komponenty zvětší, aby umožňovali pohodlnější ovládání i na dotykových obrazovkách. Osvědčený grafický vzhled s bílým pozadím a modrými prvky z webového klienta je přehledný a dobře se ovládá. Proto na něj naváže desktopový klient IS K2.



Obrázek 3.7: Webový klient – formulář

Kapitola 4

Realizace iOS klienta

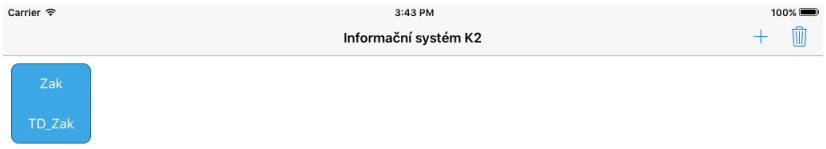
Klient pro platformu iOS implementuje část funkčnosti pro zobrazování formulářů. Celkově bude pro UniForm formuláře definováno několik typů fragmentů. Klient pro iOS bude implementovat pouze základní mód zobrazení dat, několik fragmentů a omezenou funkčnost. V informačním systému postupně dochází k implementaci navrženého řešení, tudíž není dostupná všechna popsaná funkcionalita pro jeho klienty.

Tento klient slouží pro ověření použitelnosti navrženého řešení. Do budoucnosti se spíše počítá v responzivním HTML5 klientem pro všechny mobilní platformy. Aplikace bude dostupná pouze pro zařízení iPad, na mobilní aplikaci pro iPhone by se špatně vyzkoušelo ovládání a funkčnost UniFormů.

4.1 Menu

Po spuštění aplikace se objeví jednoduché menu s ikonami datových modulů. Pro zobrazení se využije komponenta *UICollectionView*. Tím se jednotlivé ikony zarovnají horizontálně vedle sebe do mřížky. Ikona prozatím zobrazuje pouze uživatelský popis a název datového modulu. Tyto informace se ukládají pomocí knihovny Core Data do persistentního úložiště dat. Pro lepší práci s komplexním systémem ukládání pomocí Core Data se používá knihovna Magical Record. Horní menu obsahuje 2 ikony pro přidání nového záznamu do menu a výmaz všech uložených dat. Na ikonách je také definované gesto pro dlouhé kliknutí pro vymazání konkrétních záznamů.

Při otevření datového modulu je z položky menu přečten název modulu, ten je předán konstruktoru třídy *UFViewController*, která zajistí stažení dat a následně je vykreslen formulář.



Obrázek 4.1: iOS klient – menu 1

4.2 Datové třídy

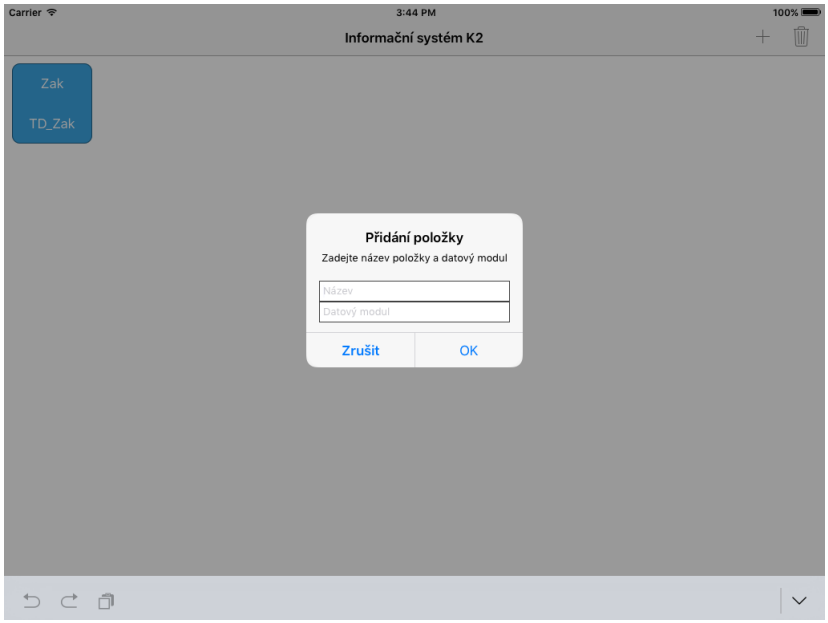
Třídy datových modulů byly vygenerování ze stáhnutých JSON dat z webových služeb pomocí nástroje JSON Accelerator. Rozdělené jsou na 2 části, které jsou stahovány ze serveru samostatně.

4.2.1 Data

Jedná se o 3 třídy, *DataModule*, *DataObject* a *DataField*. Data jsou stahovány ze zdroje *NazevWS*/Data/*NazevModulu*. Hlavní třída *DataModule* obsahuje pole:

- RequestedFields – Seznam vrácených polí datového modulu
- LastPageURL – Odkaz na poslední stránku dat
- NextPageURL – Odkaz na další stránku dat
- PrevPageURL – Odkaz na předchozí stránku dat
- Items – Pole tříd *DataObject* s jednotlivými záznamy

Po stažení dat ve formátu JSON je objekt předán této třídě, která z něho přečte hodnoty. Pro úsporu dat je ve webových službách implementováno stránkování. Zavoláním URL se tedy ze serveru vrátí pouze výchozích 30



Obrázek 4.2: iOS klient – menu 2

záznamů. Jiná velikost stránky se určuje parametrem *pageSize*. Pro načtení dalších záznamů slouží URL odkazy vypsané výše. Tato třída také obsahuje ukazatel na aktuálně vybraný záznam datového modulu.

Třída *DataObject* obsahuje pole:

- DOClassName – název datového modulu
- FieldValues – názvy a hodnoty jednotlivých polí
- ItemURL – odkaz na konkrétní záznam modulu

FieldValues je pole objektů DataField s touto strukturou:

- Name – jméno pole
- Value – hodnota pole
- Link – vazba na jiný *DataObject*
- Child – položky reprezentované pomocí *DataModule*

Do objektu *DataObject* byla implementovaná metoda pro získávání hodnoty datového pole podle jeho jména. Ta umí pracovat i s vazbami na jiné záznamy, pro ně se používá „tečková“ notace.

4.2.2 Meta

Obsahuje popis jednotlivých datových polí, vazeb a připojených položkových modulů. Data z webové služky */Meta* jsou deserializovány do třídy *MetaData* a jejich potomků. V ní jsou obsaženy informace o názvu a vlastnostech modulu. Obsahuje popis datových polí, vazeb, položkových modulů a dalších informace.

U klasických datových polí je obsažen název, popis, typ hodnoty a zda je povinné, nebo pouze pro čtení. Položkové pole obsahuje navíc informace o třídě položkového modulu. Vazební pole mají také obsažen název datového modulu a výchozí pole pro zobrazení hodnoty.

4.3 Formuláře

Datová reprezentace rozložení formuláře je stažena z webových služeb ze zdroje *NazevWS*/SystemModule/FormDefinition/*PocetSloupcu*/*NazevModulu*. Navrácená data ve formátu JSON obsahují v hodnoty:

- fragment – hlavní fragment formuláře
- requestedFields – seznam všech použitých datových polí ve formuláři

Seznam všech použitých datových polí je nezbytný pro jejich načtení ze zdroje */Data*. Ten vrací pro každý modul pouze několik základních polí a pokud chceme jiné, musíme jejich seznam předat parametrem *fields*.

Stažení dat probíhá v metodě *loadData* v *UFViewController*. Ta jsou předána třídě *FrgFragment*, která je deserializuje a vytvoří strom podřízených fragmentů. Struktura je popsaná v sekci 3.2.1. Následně se zavolá metoda *createUI:*, která rekurzivně vytvoří celé uživatelské rozhraní. Každá třída, která se podílí na vizuálním rozhraní formuláře, implementuje protokol *UIFragment*. Ten má následující strukturu:

```
@property (nonatomic, strong) id<UIFragment> parent;
@property (nonatomic, strong) UFViewController* mainController;
@property (nonatomic, strong) FrgFragment* definition;

-(void)createUI:(FrgFragment*)fragment
        in:(id<UIFragment>)parentFragment;
-(void)refreshUI;
-(CGRect)getFrame;
```

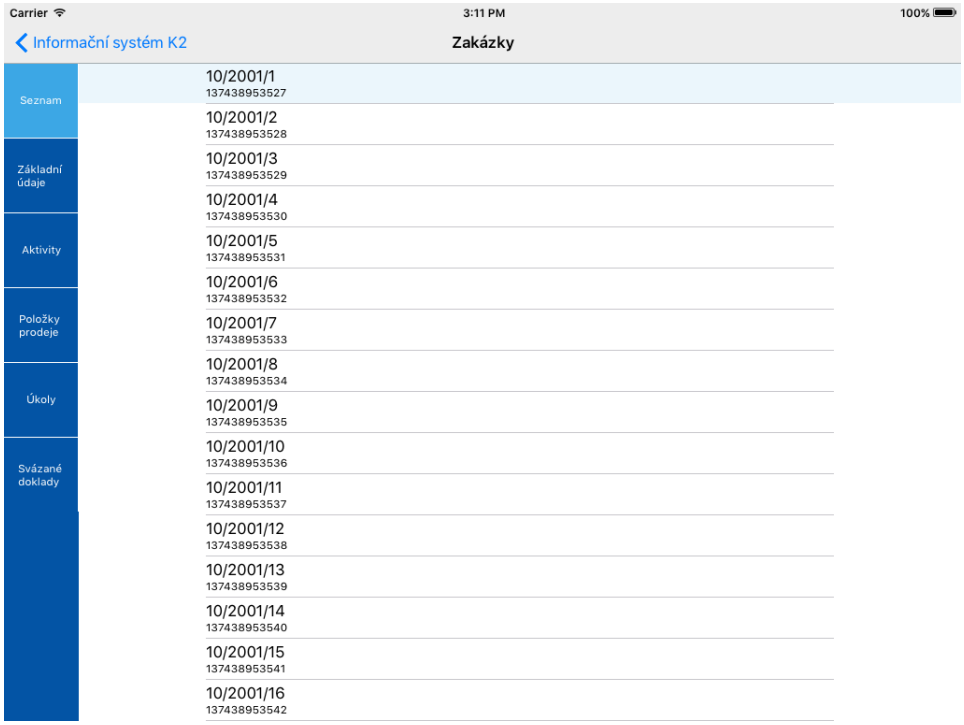
V hlavní třídě *UFViewController* je v metodě *createUI* procházeno pole *fragments* z definice fragmentu předaného jako parametr. Pro každý vnořený fragment je přečtena hodnota *type*, podle které je vytvořena instance příslušné třídy. Ta je připojena jako subview do aktuálního fragmentu a je na ní opět zavolána metoda *createIU*, které jsou předány její data.

Metoda refreshUI je volána pro načtení dat aktuálního záznamu datového modulu do vizuálních komponent UniFormu. Poslední metoda protokolu *UIFragment* je *getFrame*. Ta vrací velikost části komponenty, do které se vnořují další fragmenty.

Ze všech typů fragmentů, které budou implementované v plném klientu IS K2, budou v mobilním iOS klientu implementováno těchto 6 pro základní funkčnost.

4.3.1 TabControl

Hlavní vizuální prvek pro zobrazování jednotlivých panelů. V levé části se nacházejí tlačítka pro přepínání, právě aktivní záložka je zvýrazněna světlejší barvou. Celou pravou hlavní část tvoří kontext daného panelu. Všechny panely jsou vykresleny v jeden okamžik, ale pouze právě aktivní je nastaven jako viditelný. Ostatní jsou skryté na pozadí. Chováním se komponenta podobá *UITabBarController*, která ale může fungovat pouze jako hlavní kontrolér a nemůže být vložena do jiných objektů. Proto je třída *UITabControl* vytvořena samostatně se svojí vlastní funkčností přepínání panelů.

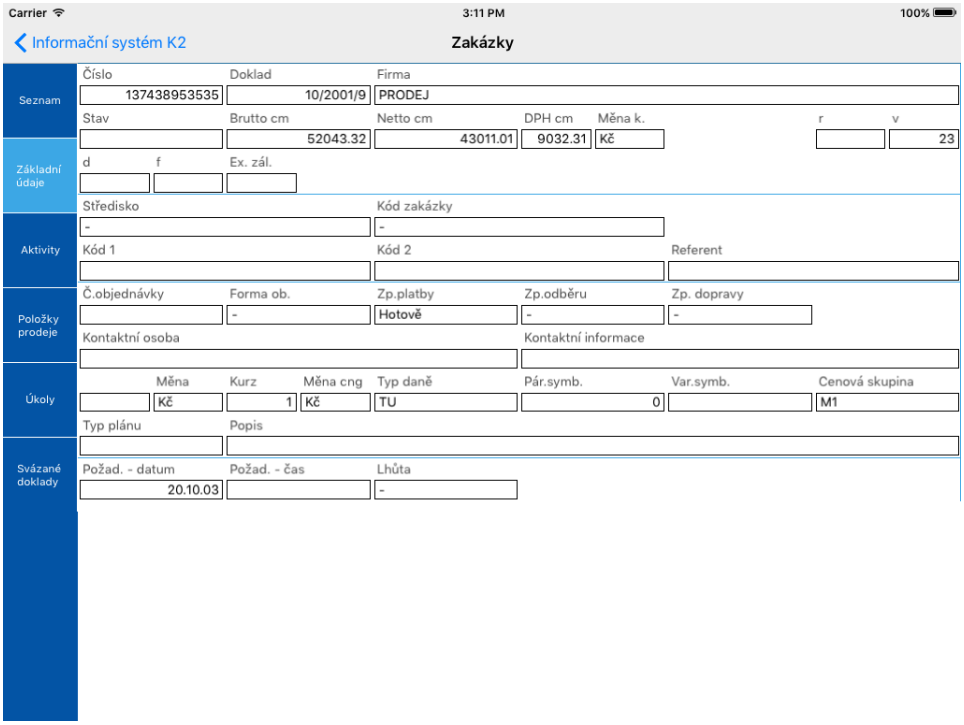


Obrázek 4.3: iOS klient – UniForm 1

4.3.2 Grid

Komponenta *UIGrid* slouží pro zobrazení seznamu položek. Pro hlavní grid na 1. straně formuláře je čten přímo seznam záznamů datového modulu.

Buňky seznamu zobrazují identifikátor položky (pokud byl stažen) a její fyzické číslo. V současné době není pomocí webových služeb dostupný nástroj pro čtení nastavení sloupců z plného klienta K2. Po zpřístupnění této funkčnosti bude vytvořen takový formát buňky, který bude dané sloupce respektovat. Jelikož datový modul obsahuje pouze určitý omezený počet položek, při posunu seznamu na první nebo poslední položku dojde k zavolání metod datového modulu pro načtení předchozích, respektive následujících položek. Pokud je datový kontext pro položku *:List*, je při kliknutí na ni nastaven aktuální záznam datového modulu na tuto položku a dojde k přepnutí strany formuláře. Rekurzivně jsou procházeni rodiče fragmentů od aktuálního to kořenový a je hledána třída *UITabControl*. Pokud je nalezena, je zavolána metoda pro přepnutí aktuální strany formuláře.



Obrázek 4.4: iOS klient – UniForm 2

4.3.3 UIPanel

Slouží pouze jako obálka pro další fragmenty. Pouze vygeneruje podřízené fragmenty, přidá je jako své *subview* a spustí na nich vytvoření komponent.

Panel se na samotném formuláři vizuálně nezobrazuje.

4.3.4 UIExpander

Chová se podobně jako *UIPanel*. Navíc oproti němu přidává horní lištu s názvem a možností srolovat celý panel. Tyto 2 funkčnosti jsou v současné době potlačené. Budou zprovozněny až bude dopracován systém jiného vypočítávání pozice fragmentů. Nyní pouze zobrazuje rámeček okolo svého kontextu.

4.3.5 UIEmptyPanel

Prázdný panel pro vyplnění místa mezi fragmenty. Jedná se pouze o potomka třídy *UIView*, který pouze nastaví svou velikost.

4.3.6 UIInput

Slouží pro zobrazování informací z polí datového modulu. Jedná se o kombinaci komponent *UITextView* pro zobrazení hodnoty a *UILabel* pro popisek pole. Podle hodnoty DataContext jsou přečteny meta informace o poli pro zobrazení popisku a určení případné vazby na poli. Pokud je zobrazovaná hodnota pole typu datum, je překonvertována z časového razítka do klasické hodnoty.

Kapitola 5

Zhodnocení nákladů a přínosů

Navržené řešení odpovídá nejvíce požadavků společnosti K2 na další rozvoj informačního systému. Nová prezentační vrstva přinese pohodlnější ovládání pro uživatele, rozšíření funkčností mimo plného klienta a jednodušší přizpůsobení K2 potřebám jednotlivých firem.

Informační systém K2 se tímto posune dále ve své konkurenceschopnosti a umožní to rychlejší implementaci nových funkčností, levnější vytváření speciálních funkčností pro zákazníky a pohodlnější práci lidem ze systémové integrace.

Jelikož se formulář nadefinuje na jednom místě a následně se automaticky správně zobrazení jak na webu, tak popřípadě s optimalizací na mobilních klientech, dojde k ušetření 1/2 až 1/3 času tvorby formulářů. Nehledě na to, že současný webový ani mobilní klienti neumožňují jednoduché vytváření uživatelských úprav.

Náklady i přínosy jsou rozdělené na klientskou a serverovou část. Obě dvě části potřebují jiné úpravy stávajícího řešení a tvorbu nové funkčnosti.

5.1 Klientská část

5.1.1 Plný klient

Klientská část se dále dělí mezi plného klienta K2, webového klienta a mobilní aplikace. Největší a nejdůležitější část je desktopový plný klient. Výhodou navrženého řešení je především možnost paralelního běhu obou uživatelských rozhraní. Díky tomu se může nejprve naprogramovat základ nové prezentační vrstvy, otestovat toto řešení interně ve firmě a vybranými zákazníky a funkčnost dále zpřesňovat podle návrhu a připomínek.

Grafický vzhled nového UI bude lépe odpovídat současným standardů a požadavkům UX. Tím, že se vyjde ze současného webového klienta, nebude tato změna pro koncové uživatele tak velká. Do formulářů se vzhledově pouze přesune větší množství funkčnosti především pro gridy a editační pole. Vzhled

formulářů v plném klientu je tedy pouze evoluce a tím je zajištěno, že bude nové rozhraní zákazníky přijato.

Jelikož se prezentační vrstva rozdělí na 2 oddělené celky, budou se z pohledu vývojového týmu lépe implementovat nové funkčnosti a úprava vzhledu. Administrátoři, konzultanti ani další lidé krom vývojového týmu nebudou mít přístup k poslední vizuální vrstvě. Její úpravy se tedy budou moci realizovat, aniž by uživatelé museli cokoliv měnit ve svých nadefinovaných formulářích.

Současné řešení, kdy se na speciálních formulářích používají přímo zděděné komponenty z VCL knihovny, trpí tímto problémem. Při změně funkčnosti v komponentě je občas nutné, aby i konzultanti upravili své speciální formuláře, které mají na starosti.

Do budoucna tedy bude možné například uživatelskou změnu barev, stylu a zobrazení, nezávisle na vlastní definici rozložení fragmentů.

Tvorba uživatelských UniForm formulářů také zabere zlomek času oproti tvorbě ve staré prezentační vrstvě. Bude možné zkopírovat celý fragment, ve kterém budou podřízené fragmenty, definované akce a rozložení, a vložit si ho do vlastního formuláře, popřípadě bude možné použít knihovní fragment.

V klientské části bude nutné nadefinovat nové třídy pro vykreslování formulářů, vytvořit kompletně nový editor rozložení fragmentů a nové definování a provádění akcí.

Název	**Počet hodin**
Nové třídy pro zobrazování	300
Výpočet pozic fragmentů	200
Napojení na DataM	150
Definice akcí v DataM	50
Definice akcí na fragmentech	50
Zobrazení a volání akcí	20
Editor rozložení	800
Vytvoření lookup	50
Přepis std. modulů	500
Celkem	2120

Tabulka 5.1: Náklady na plného klienta

5.1.2 Webový klient

Webový klient v současné době umí zobrazit většinu vizuálních komponent UniFormů. Bude tedy nutné dodělat načítání definice rozložení z aplikačního serveru K2 a formuláře generovat z tohoto datového popisu namísto statického ze souborů. Bude nutné upravit volání akcí na formulářích a datových modulech. Z klienta se odstraní všechen aplikační kód pro data, který bude nyní

volán na serveru. Bude nutné definovat všechny nové standardní fragmentové akce, respektive příkazy. Jedná se o procházení dat v gridu, přepínání stran záložek, vytváření nových záznamů atd.

Název	**Počet hodin**
Čtení rozložení UniFormů	50
Výpočet pozic fragmentů	40
Definice akcí fragmentů	60
Čtení akcí DataM	20
Zobrazení a volání akcí	20
Celkem	190

Tabulka 5.2: Náklady na webového klienta

5.1.3 Mobilní klienti

Mobilní iOS klient do současné doby nezobrazoval žádná formuláře z IS K2. Celá aplikace se tedy nově vytvořila v rámci této práce. Jelikož se navržené řešení teprve implementuje na serverové straně K2 a v jeho plném klientovy, není všechna nadefinovaná funkčnost přístupná pomocí webových služeb.

iOS aplikace tedy slouží jako ukázka toho, jak se mohou a budou mobilní klienti dále rozvíjet. Implementovaná funkčnost pokrývá načtení definice rozložení a čtení dat a popisu polí a modulů. Klient umí zobrazit komponenty *Tab control*, *Grid*, *Input* a *Panel*. Bylo implementované vygenerování rozložení a vnořování fragmentů, čtení jak seznamu záznamů pro grid, tak přímo polí aktuálního záznamu pro input.

Přes webové služby není zatím možné volat datové akce, číst informace o sloupcích gridu a další funkčnost, která bude po zapracování na straně AS postupně přidávána i do iOS klienta.

Název	**Počet hodin**
Definice tříd fragmentů	10
Čtení rozložení UniFormů	10
Čtení dat a jejich popisů	15
Generování rozložení	25
Celkem	60

Tabulka 5.3: Náklady na iOS klienta

Klient pro platformu Android obsahuje některé formuláře z modulů CRM, které jsou ale naprosto odlišné od formulářů v plném klientovy. Pro UniFormy bude možné využít stávající komponenty pro implementaci fragmentů *Grid*, *Input* a *Tab control*. Pro vytvoření alespoň základní funkčnosti pro jednoduché

zobrazení a editaci dat bude nutné implementovat čtení fragmentů a jejich zobrazení a připravit práci s akcemi. Oproti iOS klientu je zde již vyřešení čtení dat a jejich popisů pomocí webových služeb.

Název	**Počet hodin**
Definice tříd fragmentů	10
Čtení rozložení UniFormů	10
Práce s akcemi	25
Generování rozložení	25
Celkem	70

Tabulka 5.4: Náklady na Android klienta

5.2 Serverová část

Na serverové straně IS K2 je nutné implementovat uložení fragmentů a upravit práci s daty v třídě *TDataM*. Všechny úpravy se musí dělat kompatibilní s aplikačním serverem, aby byly přístupné pro webového i mobilní klienty.

Pro datový modul je nutné implementovat mechanismu AsControlValue pro reakci na změnu hodnoty. Dále je nutné naprogramovat objekt pro definování akcí a jeho obsluhu. Tím se otevře prostor pro přepis funkčností ze stávajících formulářů do datových modulů. Informační systém K2 obsahuje několik stovek datových modulů, u všech bude nutné přepracovat datové akce a reakce na změnu hodnoty.

Pro fragmenty UniFormů bude nutné vytvořit nový datový modul s pomocnými tabulkami a zapracovat podporu pro ukládání a načítání. Algoritmus musí respektovat zařazení na konkrétní úroveň, přiřazení počtu sloupců a zohlednit již vytvořené fragmenty.

Název	**Počet hodin**
Datový modul fragmentů	50
Třídy reprezentující fragmenty	40
Akce datových modulů	40
Reakce na změnu pole	50
Zpřístupnění pro klienty	10
Přepis std. DataM	600
Celkem	790

Tabulka 5.5: Náklady na serverovou část

Díky serializaci vlastních tříd do XML je možné bez úpravy databáze přidávat další fragmenty se specifickou funkčností.

Celý systém fragmentů a systému zpracování událostí v datovém modulu otevře cestu pro další rozšíření IS K2. Bude možné vytvářet specifické aplikace na míru každé firmě, které budou komunikovat pomocí webových služeb, ale definice funkčnosti a vzhledu zůstane na jednom místě v IS K2.

Závěr

Cílem práce bylo analyzovat dostupné varianty a navrhnout řešení prezentační vrstvy, která umožní další rozvoj informačního systému. Měl jsem možnost seznámit se s několika dostupnými řešení na trhu a více proniknou do vývoje velkého informačního systému. Navržené řešení bylo přijato a v době odevzdávání práce se postupně implementovalo na straně výrobce IS K2. Klient pro platformu iOS zatím poskytuje pouze základní funkčnost ale s přibývající funkčností na straně serveru se bude dále rozvíjet. Při vývoji klienta jsem využil nasbírané zkušenosti z bakalářské práce, kde jsem se také věnoval čtení dat z IS K2.

Literatura

[1] Research 2 Guidance: Performance of cross platform app development tools [online]. 2016, [cit. 2016-03-23]. Dostupné z: http://research2guidance.com/2013/05/27/cross-platform-tools-can-save-more-than-30-of-app-development-time/

[2] Statista Inc.: Computer operating systems market share (desktop/tablet) 2012-2015 [online]. 2016, [cit. 2016-03-29]. Dostupné z: http://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/

[3] Embarcadero: FireUI: Revolutionary Multi-Device Development [online]. 2016, [cit. 2016-03-30]. Dostupné z: https://www.embarcadero.com/products/rad-studio/fireui

[4] Xamarin: Shared Projects [online]. 2016, [cit. 2016-04-04]. Dostupné z: https://developer.xamarin.com/guides/cross-platform/application_fundamentals/shared_projects/

[5] Microsoft: Silverlight 5 System Requirements [online]. 2016, [cit. 2016-03-24]. Dostupné z: https://www.microsoft.com/getsilverlight/locale/en-us/html/installation-win-SL5.html

[6] The Paciello Group: HTML5 Accessibility [online]. 2016, [cit. 2016-04-10]. Dostupné z: http://www.html5accessibility.com

[7] K2 atmitec s.r.o.: K2 atmitec s.r.o. [online]. 2016, [cit. 2016-02-24]. Dostupné z: http://www.k2.cz/cz/o-nas/profil-spolecnosti.html

[8] Embarcadero: VCL Overview [online]. 2016, [cit. 2016-02-24]. Dostupné z: http://docwiki.embarcadero.com/RADStudio/Seattle/en/VCL_Overview

[9] Embarcadero: Delphi [online]. 2016, [cit. 2016-03-25]. Dostupné z: https://www.embarcadero.com/products/delphi

[10] Microsoft: Silverlight Support Lifecycle [online]. 2016, [cit. 2016-03-24]. Dostupné z: https://support.microsoft.com/en-us/lifecycle?C2=12905

[11] Embarcadero: TPresentedControl [online]. 2016, [cit. 2016-03-30]. Dostupné z: http://docwiki.embarcadero.com/Libraries/XE8/en/FMX.Controls.Presentation.TPresentedControl

[12] Embarcadero: Publish FireMonkey apps to the Web with RAD [online]. 2016, [cit. 2016-04-01]. Dostupné z: http://edn.embarcadero.com/en/article/43158

[13] Embarcadero: Multi-Device Applications Index [online]. 2016, [cit. 2016-04-01]. Dostupné z: http://docwiki.embarcadero.com/RADStudio/XE8/en/Multi-Device_Applications_Index

[14] FMSoft Co. Ltd.: uniGUI Web Application Framework [online]. 2016, [cit. 2016-04-04]. Dostupné z: http://www.unigui.com

[15] Xamarin: The Xamarin Story prvn[online]. 2016, [cit. 2016-04-04]. Dostupné z: https://www.xamarin.com/about

[16] Xamarin: Mobile Application development [online]. 2016, [cit. 2016-04-04]. Dostupné z: https://www.xamarin.com/platform

[17] Xamarin: Windows Platform Features [online]. 2016, [cit. 2016-04-04]. Dostupné z: https://developer.xamarin.com/guides/xamarin-forms/platform-features/windows/

[18] Wikipedia: HTML5 [online]. 2016, [cit. 2016-04-06]. Dostupné z: https://cs.wikipedia.org/wiki/HTML5

[19] W3Schools: HTML5 Server-Send Events [online]. 2016, [cit. 2016-04-06]. Dostupné z: http://www.w3schools.com/html/html5_serversentevents.asp

[20] W3Schools: HTML5 Local Storage [online]. 2016, [cit. 2016-04-06]. Dostupné z: http://www.w3schools.com/html/html5_webstorage.asp

[21] ERP Software Blog: Microsoft Dynamics GP 2016 New Features [online]. 2016, [cit. 2016-04-06]. Dostupné z: http://www.erpsoftwareblog.com/2015/10/microsoft-dynamics-gp-2016-new-features-full-control-from-any-device-anywhere-with-html5/

[22] DevExtreme: DataGrid HTML5/JS [online]. 2016, [cit. 2016-04-06]. Dostupné z: http://js.devexpress.com/WebDevelopment/DataGrid/

Příloha A

Seznam použitých zkratek

VCL Visual Component Library

IS K2 Informační systém K2

UX User Experience

UI User Interface

AS Aplikační server K2

REST Representational state transfer

PDF Portable Document Format

SWS Server webových služeb

API Application Programming Interface

RTTI Run-time type information

UWP Universal Windows Platform

GPU Graphic processing unit

PCL Portable Class Libraries

SSE Server-Send Events

JS JavaScript

XAML Extensible Application Markup Language

GUID Globally unique identifier

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF