# Designer formulářů pro TESCOSW a.s.

Bc. David Kolář

Diplomová práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Bc. David Kolář** |
| Osobní číslo: | **A21496** |
| Studijní program: | **N0613A140022 Informační technologie** |
| Specializace: | **Softwarové inženýrství** |
| Forma studia: | **Kombinovaná** |
| Téma práce: | **Designer formulářů pro TESCOSW a.s.** |
| Téma práce anglicky: | **Form Designer for TESCOSW a.s.** |

## Zásady pro vypracování

1. Seznamte se s technologickým frameworkem společnosti TESCOSW a.s.
2. Seznamte se s aktuálním procesem tvorby definic formulářů a jejich životním cyklem.
3. Analyzujte strukturu definic formulářů, zjistěte množinu možných komponent a jejich vlastností.
4. Naprogramujte v požadovaných technologiích klientskou a serverovou část aplikace.
5. Vyhodnoťte řešení aplikace.
6. Vypracujte závěr a navrhněte směry budoucího vývoje aplikace.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:


1. BANKS, Alex a Eve PORCELLO. Learning React: functional web development with React and Redux. Beijing: O'Reilly Media, 2017. ISBN 978-1-491-95462-1.
2. TROELSEN, Andrew W. a Philip JAPIKSE. Pro C# 7: with .net and .net core. Eighth edition. New York, NY: Apress, [2017]. ISBN 978-1-4842-3017-6.
3. HOTEK, Mike. Microsoft SQL Server 2008: krok za krokem [online]. Brno: Computer Press, 2009 [cit. 2022-10-21]. ISBN 978-80-251-2466-6.
4. FENTON, Steve. Pro TypeScript: application-scale JavaScript development. [New York]: Appress, [2014]. The expert's voice in TypeScript. ISBN 978-1-4302-6791-1.
5. LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Brno : Computer Press, 2011. ISBN 978-80-251-3539-6.


Vedoucí diplomové práce: **Ing. Bc. Pavel Vařacha, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

**doc. Ing. Jiří Vojtěšek, Ph.D.**
děkan

**prof. Mgr. Roman Jašek, Ph.D., DBA**
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 25. 05. 2023

David Kolář, v. r.
…….
podpis studenta

## ABSTRAKT

Cílem diplomové práce je navrhnout a uvést do provozu aplikaci pro tvorbu webových formulářů v rámci technologické platformy společnosti TESCOSW a.s. Tato aplikace má nahradit a rozvinout funkčnost nástroje zvaného XGEN, který se doposud v této firmě využíval a který již technicky zastaral natolik, že se jeho údržba stala neúnosnou. V teoretické části práce jsou definovány pojmy spojené s procesem tvorby formulářů, popis dosavadního řešení a požadavky na novou aplikaci. V praktické části je uveden analytický návrh systému a jeho integrace.



Klíčová slova: TESCOSW a.s., Designer formulářů, React, Typescript, C#



## ABSTRACT

The goal of this thesis is to design and put into operation an application for creating web forms within the technological platforms of TESCOSW a.s. This application is supposed to replace and develop the functionality of a tool called XGEN, which has been used in this company until now and which has become so technically outdated that its maintenance has become unbearable. In the theoretical part of the thesis, the terms associated with the process of creating forms, the description of the existing solution and the requirements for the new application are defined. The practical part presents the analytical design of the system and its integration.



Keywords: TESCOSW a.s., Form designer, React, Typescript, C#

Tímto bych chtěl poděkovat mému vedoucímu diplomové práce za asistenci, mé přítelkyni za trpělivost, kterou se mnou musela mít, a nakonec i mému novorozenému synovi za to, že byl mimořádně hodný, když tatínek pracoval na následujících stránkách.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

# OBSAH


**ÚVOD** ..... **9**

**I TEORETICKÁ ČÁST** ..... **10**

**1 SPOLEČNOST TESCOSW A.S.** ..... **11**

1.1 ZADÁNÍ K DESIGNERU FORMULÁŘŮ ..... 12

**2 FORMULÁŘ A JEHO DEFINICE** ..... **13**

2.1 STRUKTURA DEFINICE FORMULÁŘE ..... 13

2.2 XML ..... 14

2.2.1 XML Deklarace ..... 14
2.2.2 Značka ..... 14
2.2.3 Element ..... 14
2.2.4 Atribut ..... 15

2.3 PODOBA DEFINICE FORMULÁŘE ..... 15

2.4 KLIENTSKÁ PROSTŘEDÍ VYUŽÍVAJÍCÍ FORMULÁŘE ..... 16

**3 NÁSTROJ XGEN** ..... **18**

3.1 DŮVODY NEVHODNOSTI DALŠÍHO POUŽÍVÁNÍ ..... 18

3.2 HLAVNÍ UŽIVATELSKÉ PRVKY ..... 19

3.2.1 Okno schématického náhledu ..... 19
3.2.2 Okno s nastavením vlastností ..... 20
3.2.3 Okno s datovým modelem ..... 21
3.2.4 Okno s navigačním menu ..... 22

**4 SELECT ARCHITECT** ..... **23**

**5 ANALÝZA KOMPONENT A JEJICH VLASTNOSTÍ** ..... **25**

5.1 REVIZE KOMPONENT ..... 25

5.2 REVIZE VLASTNOSTÍ ..... 26

**6 APLIKACE PORTÁLU VÝVOJÁŘE** ..... **27**

**7 POPIS POUŽITÝCH TECHNOLOGIÍ A KNIHOVEN** ..... **29**

7.1 TECHNOLOGIE VYVINUTÉ SPOLEČNOSTÍ TESCOSW A.S. ..... 29

7.1.1 MultiWeb ..... 29
7.1.2 TEAF Server ..... 30

7.2 VEŘEJNĚ DOSTUPNÉ VYUŽITÉ TECHNOLOGIE ..... 30

7.2.1 TypeScript ..... 31
7.2.2 React ..... 31
7.2.3 Redux ..... 31
7.2.4 Redux-Saga ..... 32
7.2.5 ASP.NET ..... 32

**II PRAKTICKÁ ČÁST** ..... **33**

**8 ARCHITEKTURA DESIGNERU FORMULÁŘŮ** ..... **34**

8.1 Popis architektury frontend části ..... 34
8.1.1 Integrace do MultiWeb klienta ..... 34
8.1.2 Struktura stromu React komponent ..... 35
8.1.3 Modul picasso.ts ..... 35
8.1.4 Modul reducer.ts ..... 36
8.1.5 Modul saga.ts ..... 38
8.1.6 Modul konfigurace ..... 39
8.2 Popis architektury backend části ..... 41
8.2.1 Integrace do TEAF ..... 42
8.2.2 Projekt DevPortal_FormDesigner ..... 42
8.2.3 Projekt ExpertDataProvider ..... 43
**9 AUTOMATIZACE TVORBY DOKUMENTACE ..... 44**
9.1 Azure DevOps Server ..... 44
9.2 Skript pro převod z JS na JSON ..... 45
9.3 Transformace z JSON na tabulku ve Wiki formátu ..... 45
9.4 Upload na firemní Wikipedii ..... 46
**10 POPIS OVLÁDACÍCH PRVKŮ ..... 49**
10.1 Záložka Model ..... 49
10.2 Záložka Hierarchie ..... 50
10.3 Panel nástrojů ..... 52
10.4 Panel schématického zobrazení ..... 52
10.4.1 ListFrame ..... 54
10.4.2 Pages ..... 54
10.4.3 GridPanel ..... 54
10.4.4 PutFormInPanel ..... 55
10.5 Panel definičního zobrazení ..... 55
10.6 Panel náhledového zobrazení ..... 56
10.7 Záložka vlastností ..... 56
10.8 Záložka Komponenty ..... 57
10.9 Editor lokalizací ..... 59
10.10 Průvodce vytvoření formuláře ..... 60
**11 ZHODNOCENÍ APLIKACE ..... 62**
**ZÁVĚR ..... 63**
**SEZNAM POUŽITÉ LITERATURY ..... 65**
**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK ..... 67**
**SEZNAM OBRÁZKŮ ..... 68**
**SEZNAM UKÁZEK KÓDŮ ..... 69**
**SEZNAM PŘÍLOH ..... 70**

# ÚVOD

Společnost TESCOSW a.s. je na trhu již přes 30 let, a tak je přirozené, že po této době některé její vnitřní systémy zastarají natolik, že vznikne potřeba jejich aktualizace či nahrazení. Jedním z těchto systémů je interně využívaný nástroj zvaný XGEN, který slouží pro tvorbu webových formulářů v uživatelsky přívětivé formě, která nevyžaduje znalost technologií jako je HTML, CSS nebo JavaScript a zároveň umožňuje použít stejný formulář na různých aplikacích psaných v různých technologiích. Mezi hlavní důvody, proč se firma rozhodla tento nástroj nahradit, patří jeho technologická zastaralost a absence důkladné znalosti všech konstrukčních prvků, ze kterých se formuláře skládají.

Technologická zastaralost vyplývá z použití programovacího jazyka Delphi, který využívá syntaxe Object Pascalu a v aktuální době neposkytuje potřebnou technickou podporu.

# I. TEORETICKÁ ČÁST

# 1 SPOLEČNOST TESCOSW A.S.

Společnost TESCOSW a.s. byla založena roku 1991 a jedná se o jednu z největších českých IT firem na trhu [1]. Primárně se specializuje se na tvorbu webových aplikací zabývajících se managementem majetku, nemovitostí či organizací zdravotních středisek. Její hlavní sídlo je v Olomouci na adrese tř. Kosmonautů 1 [2].


Obrázek 1 Sídlo společnosti TESCOSW a.s.

## 1.1 Zadání k Designeru formulářů

V následujícím odstavci je uvedeno detailní zadání, které bylo společností TESCOSW a.s. dodáno při zahájení prací na této diplomové práci.

*„Je potřeba vytvořit náhradu za dosavadní aplikaci pro úpravu formulářů zvanou XGEN. Tato aplikace je již technicky a morálně zastaralá a současně je obtížné s ní pracovat. Rovněž je takřka nemožné do ní provést jakýkoliv větší rozvoj kvůli použitému programovacímu jazyku, ve kterém byla původně vyvíjena.*

*Naší potřebou je vytvoření náhrady za tuto aplikaci, která bude pojmenovaná jako Designer formulářů a která bude využívat námi vyvinuté technologie. Zároveň bude součástí již existující aplikace zvané Portál vývojáře a mohla by potenciálně být využívána našimi zákazníky, kteří by si mohli upravovat vzhled formulářů sami a nemuseli by kontaktovat technickou podporu kvůli relativně malým úpravám. Designer formulářů bude využívat datového modelu aplikace Portál vývojáře, ze kterého bude čerpat informace o dostupných třídách se kterými se každý formulář váže.*

*Hlavními požadavky na Designer formulářů jsou rychlost zobrazování, kdy aplikaci XGEN vykreslování složitějších formulářů dělá již očividné problémy a zjednodušení ovládání pro zvýšení efektivity při tvorbě nových formulářů. Dále je to zachování zpětné kompatibility definic formulářů se starým nástrojem, ukládání definic formulářů do nástroje Select Architect (stejně, jako to dělá XGEN) a rešerše všech formulářových komponent a jejich vlastností, jejichž přesný výčet aktuálně neexistuje a s ním ani řádná dokumentace.“*

# 2 FORMULÁŘ A JEHO DEFINICE

Pod pojmem formulář si lze představit znovupoužitelnou část webové aplikace. Může se jednat například o formulář pro správu uživatelů, kde je administrátor aplikace schopen přidávat, odebírat či upravovat data o uživatelích aplikace. Dále se může jednat například o formulář pro zobrazení majetku firmy či přehled pacientů konkrétního lékaře. Formulář tedy reprezentuje jakési podokno aplikace sloužící pro zobrazení a případnou editaci dat.

Tyto formuláře jsou z hlediska funkčnosti relativně jednoduché a často se na nich opakuje určitá funkčnost (zobrazení nějakého seznamu, editace označeného záznamu, vyvolání nějaké akce u označeného záznamu atp.). Tato repetetivnost vedla vývojáře ve firmě k tomu, vytvořit určitou abstrakci nad procesem tvorby těchto formulářů. Není totiž žádoucí pro každý takový formulář, kterých ve výsledné aplikaci mohou být stovky, programovat funkčnost „od znova“. Rozhodli se tedy pro vytvoření znovupoužitelných komponent, ze kterých se bude sestavovat výsledný formulář.

Rozpad na komponenty nicméně nepředstavuje finální krok k zefektivnění tvorby formulářů. Problém totiž představuje následné seskládání komponent tak, aby vytvořily funkční formulář. K tomu je zapotřebí zaměstnanců znalých programování, což představuje dodatečné náklady na vývoj. Jakožto řešení tohoto problému byl vytvořen nástroj XGEN, který umožňuje vytvářet definice jednoduše pomocí grafického editoru a odstraňuje nutnost znalosti programování u jeho uživatelů. Výstupu aplikace XGEN se říká definice formuláře a obsahuje informace o samotném formuláři, jeho komponentách a vlastnostech těchto komponent.

Posledním a neméně důležitým faktorem pro vznik formulářů byla technologická květnatost aplikací společnosti. V současné době společnost udržuje tři druhy prohlížečových aplikací, kdy každá je naprogramována v jiné technologii. Každá z těchto aplikací ale zobrazuje principiálně stejné formuláře. Vznikla tedy potřeba, aby jedna definice formuláře fungovala na všech třech aplikacích současně, bez žádných nutných úprav.

## 2.1 Struktura definice formuláře

Struktura formuláře je definovaná pomocí značkovacího jazyka XML. Tato definice obsahuje všechny komponenty formuláře a jejich vlastnosti. Formulář je asociovaný k datové třídě a každá třída může mít několik formulářů.

## 2.2 XML

Jedná se o značkovací jazyk vyvinutý konsorciem W3C. XML je zkratkou pro „Extensible Markup Language“, tedy rozšiřitelný značkovací jazyk. Využívá se k popisu datových struktur a dokumentů, přenosu dat mezi systémy anebo pro tvorbu strukturovaných konfiguračních souborů. XML soubory využívají příponu „.xml“. Mezi základní prvky XML patří jeho deklarace, značky, elementy a atributy [3].

### 2.2.1 XML Deklarace

Jedná se o speciální značku obsahující detaily o vytvářeném XML souboru (verze XML, kódování znaků, schopnost zobrazení bez externích zdrojů). Je nepovinná, ale pokud je uvedena, musí se vyskytovat na prvním řádku dokumentu [3].

```
<!-- Příklady XML deklarace-->
<?xml version="1.0" ?>
<?xml version="1.0" encoding="utf-8" ?>
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<?xml version="1.0" encoding="windows-1250" standalone="no"?>
```

Obrázek 2 Příklady podoby deklarace XML

### 2.2.2 Značka

Značka je syntaktický prvek XML používaný k definování struktury a obsahu souboru. Skládá se ze startovací značky, obsahu a ukončovací značky. Existují tři druhy značek, startovací, ukončovací a tzv. prázdné značky [3].

```
<znacka> <!-- startovací značka-->
</znacka> <!-- ukončovací značka-->
<znacka/> <!-- prázdná značka-->
```

Obrázek 3 Druhy XML značek

### 2.2.3 Element

Element je logickou součástí XML dokumentu, která představuje podmnožinu dat (např. osoba, doklad, cena, …). Element se skládá ze startovací značky, ukončovací značky a obsahu mezi těmito dvěma značkami. Element bývá často zaměňován se značkou, kde element reprezentuje sémantickou podmnožinu dat XML dokumentu a značka jeho syntaktickou součást identifikující prvek. Elementy se rovněž můžou vyskytovat

v samostatném souboru a nemusí mít uvedenou deklaraci, jelikož je předpokládáno, že se element stane součástí větší XML struktury [3].

### 2.2.4 Atribut

Atribut je konstrukt umožňující přidávání vlastností jednotlivým značkám XML dokumentu. Jedná se o dvojici název-hodnota, která se může vyskytovat na startovací či prázdné značce, ne však na ukončovací značce. Dvojice je oddělena znaménkem „=“ a zároveň platí, že hodnota musí být obklopena jednoduchými případně dojitými uvozovkami. Uvozovky však nesmí být zaměněny, tedy že by před hodnotou byla jednoduchá závorka a za ní dvojitá či naopak [3].

```
<znacka nazevAtributu="hodnotaAtributu" nazevJinehoAtributu='hodnotaJinehoAtributu'>
    Obsah
</znacka>
```

Obrázek 4 Příklad použití XML atributů

## 2.3 Podoba definice formuláře

Každá definice obsahuje kořenový element „Form“. Tento element pak obsahuje element „Components“ ve kterém jsou uvedeny všechny další komponenty formuláře. Tyto komponenty jsou reprezentovány značkou „Component“, kde konkrétní druh komponenty je určen pomocí XML atributu „XGClass“.

```
<Form Left="0" Top="0" Width="1000" Height="800" ... AllowResponsiveLayout="True">
    <Components>
        <Component XGClass="LiteListFrame" I="1" Left="12" ... ObjectName="" />
        <Component XGClass="LiteListFrame" I="2" Left="12" ... masterFrame="LiteListFrame1">
            <Component XGClass="LiteListFrame" I="3" Align="alClient" Left="0" ... ObjectName="" />
        </Component>
    </Components>
</Form>
```

Obrázek 5 Příklad možné podoby definice formuláře

Tento způsob zápisu definice však není optimální z hlediska velikosti výsledného XML souboru. Vedení společnosti bylo navrženo, aby byl odebrán element „Components“ a aby konkrétní druh komponenty byl uveden již jako název značky, a ne jako hodnota atributu. Zamýšlená podoba je vidět na obrázku níže. V rámci zpětné kompatibility s původním nástrojem XGEN a dalšími nástroji společnosti TESCOSW a.s. však zapracování tohoto návrhu bylo odloženo do budoucích fází vývoje Designeru formulářů.

```
<Form Left="0" Top="0" Width="1000" Height="800" ... AllowResponsiveLayout="True">
    <LiteListFrame I="1" Left="12" ... ObjectName=""/>
    <DetailFrame I="2" Left="12" ... masterFrame="LiteListFrame1">
        <GridPanel I="3" Align="alClient" Left="0" ... ObjectName="" />
    </DetailFrame>
</Form>
```

Obrázek 6 Návrh optimalizace zápisu definice formuláře

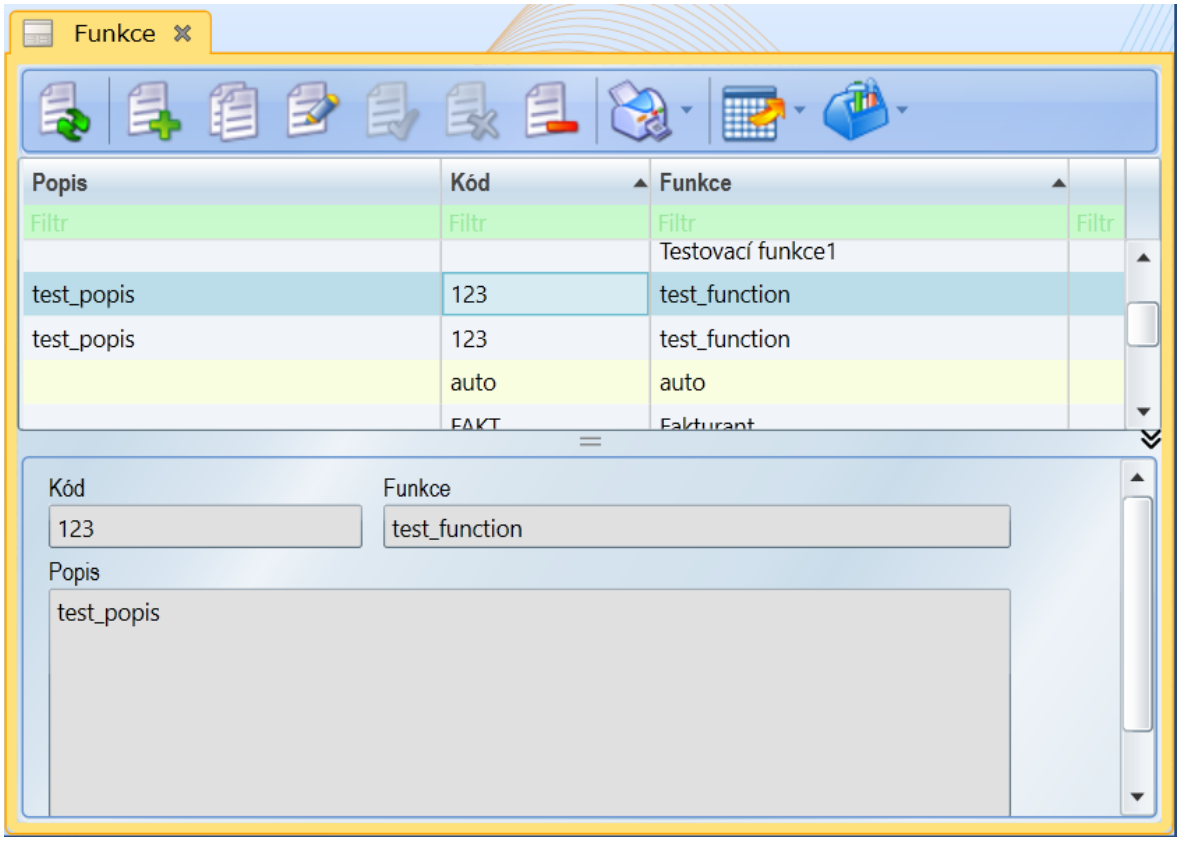
## 2.4 Klientská prostředí využívající formuláře

Zobrazování definic formulářů podporují tři klientská prostředí vyvinutá společností TESCOSW a.s. Konkrétně se jedná o prostředí zvaná Silverlight, LightWeb a MultiWeb. Každé prostředí započalo vývoj v jiné době, kdy nejstarším je Silverlight. Zároveň platí, že každé je vytvořeno pomocí jiné technologie. Aktuální vrchol představuje prostředí MultiWeb, který je ve vývoji od roku 2016. Naopak Silverlight, kvůli ukončení podpory od společnosti Microsoft, je již na ústupu a postupně je nahrazován novějšími prostředími [4].

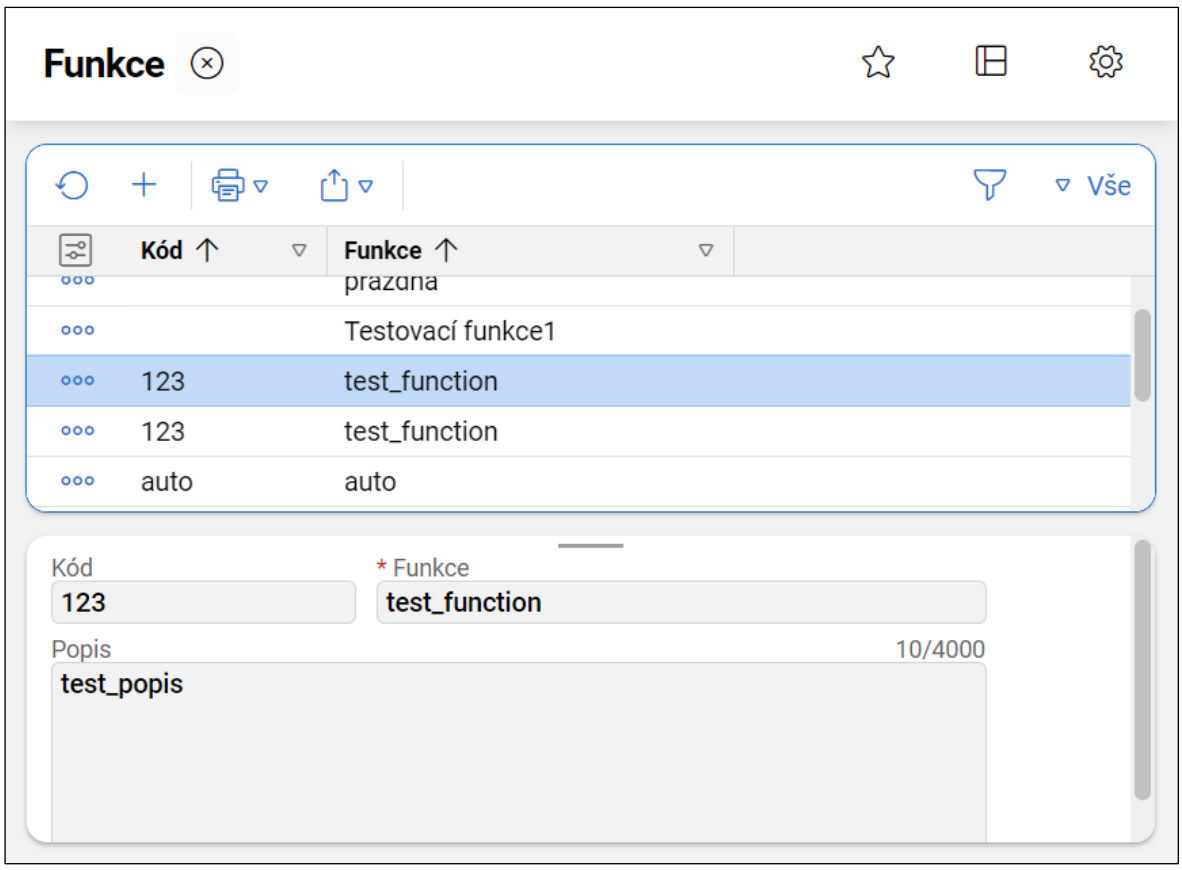
Vztah mezi formulářem a klientskými prostředími je takový, že každá definice formuláře by měla být bez jakýchkoliv úprav zobrazitelná v libovolném prostředí. Toto pravidlo však není vždy dodrženo a vyskytují se případy, kdy je nutné provést nějaký zásah do definice. To představuje problém zejména při přechodu ze starších klientských prostředí do novějších. Jedním z nepřímých efektů této diplomové práce bude redukce tohoto problému, kde díky eliminaci duplicitních či zastaralých formulářových komponent bude možné tyto definice jednodušeji převést do modernější podoby. Na následujících obrázcích Obrázek 7 a Obrázek 8 lze vidět porovnání zobrazení jedné definice pro formulář *Funkce* mezi SL a MW klienty. Formulář se skládá ze dvou hlavních komponent, konkrétně komponent *ListFrame* a *DetailFrame*.

První zmíněná komponenta je určena pro zobrazování dat z databáze ve formě přehledné tabulky obsahující sloupce, které reprezentují atributy dané tabulky. Nachází se v horní polovině stránky a jejími dalšími funkcemi jsou například filtrování nebo označování konkrétních záznamů tabulky.

Komponenta *DetailFrame* slouží pro detailní zobrazení označeného záznamu a jeho přímou editaci. Sama o sobě této funkce komponenta ale není schopna a vyžaduje ve svém těle specifické editační komponenty, kterým musí být definováno, jaké atributy označeného záznamu mají editovat. *DetailFrame* tedy slouží pouze jako jakýsi kontejner řešící komunikaci s přiřazenou komponentou *ListFrame*.



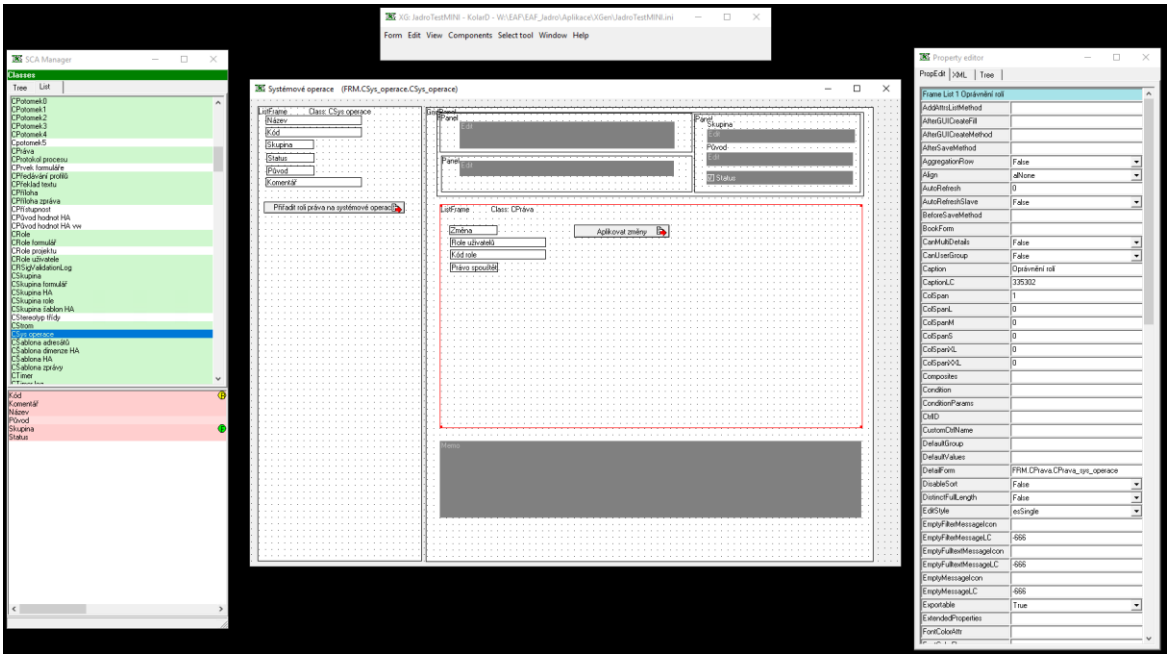

Obrázek 7 Podoba formuláře *Funkce* v prostředí Silverlight




Obrázek 8 Podoba formuláře *Funkce* v prostředí MultiWeb

# 3 NÁSTROJ XGEN

Jedná se o dosavadní řešení tvorby a správy definic formulářů ve firmě, které úzce spolupracuje s programem Select Architect, který využívá pro persistenci dat. Select Architect je popsán v samostatné kapitole. Nástroj XGEN je naprogramován v jazyce Delphi a jeho vývoj započal v roce 2010. V dnešní době se jedná o poněkud problematický produkt, jehož technologický dluh brání společnosti k přechodu na modernější technologie a agilnější vývojový proces. Při spuštění je načtena uživatelsky definovaná konfigurace, která určuje, s jakým datovým modelem aplikace pracuje. Datovým modelem označujeme výčet všech tříd a jejich prvků, které se nacházejí v koncové zákaznické aplikaci.



Obrázek 9 Podoba aplikace XGEN

## 3.1 Důvody nevhodnosti dalšího používání

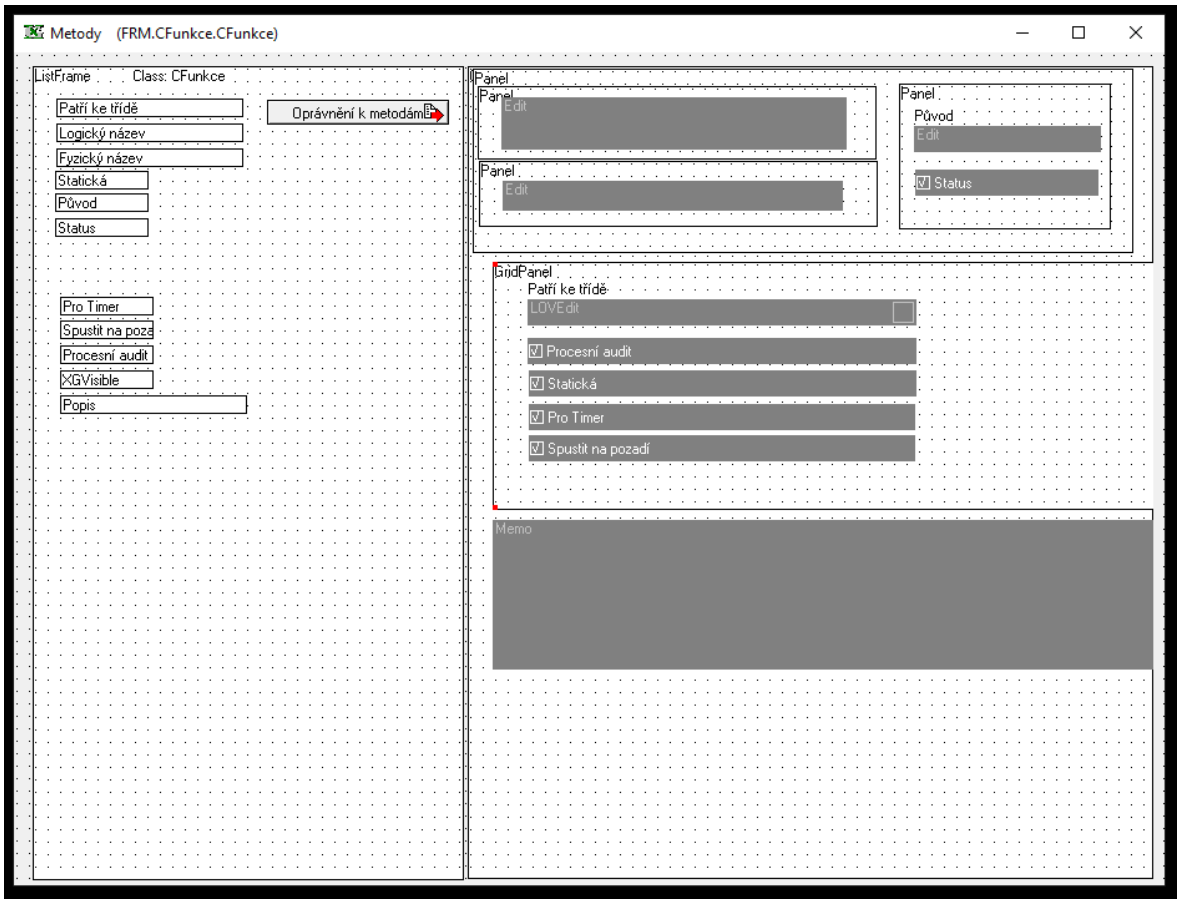
Mezi hlavní důvody pro přepis starého nástroje patří použitý programovací jazyk Delphi, pro který se v dnešní době obtížně hledají vývojáři [5]. Dále je to uživatelská nepřívětivost, morální zastarání a nekonzistence dat popisujících jednotlivé komponenty formulářů a jejich vlastností. Pro Designér formulářů jsou ve společnosti i vize pro využití u koncových zákazníků, kteří by si mohli změnit vzhled formulářů či si vytvořit úplně nové, bez nutnosti kontaktovat podporu společnosti TESCOSW a.s.

## 3.2 Hlavní uživatelské prvky

Program zobrazuje své dílčí části do samostatných oken, které umožňují libovolné pozicování a změnu velikosti. Tento přístup je velice nepraktický a často dochází k situacím, kdy, pokud má uživatel jedno z oken na externím monitoru, tak při odpojení tohoto monitoru okno nezmění pozici a uživateli tak toto okno „zmizí“.

### 3.2.1 Okno schématického náhledu

Okno obsahuje grafické znázornění aktuálně editované definice formuláře. Je založeno na systému WYSIWYG (What you see is what you get), tedy editovaný obsah je prezentován ve formě, v jaké přibližně bude vypadat v reálném nasazení/použití. Komponenty jsou hierarchicky vrstveny přes sebe a lze je v rámci okna posouvat tažením myši. Komponenty lze rovněž i pomocí kurzoru zvětšovat či zmenšovat. Alternativním způsobem změny rozměrů může být přímé nastavení hodnot vlastností pro výšku, šířku, vzdálenost od horního okraje či vzdálenost od levého okraje.



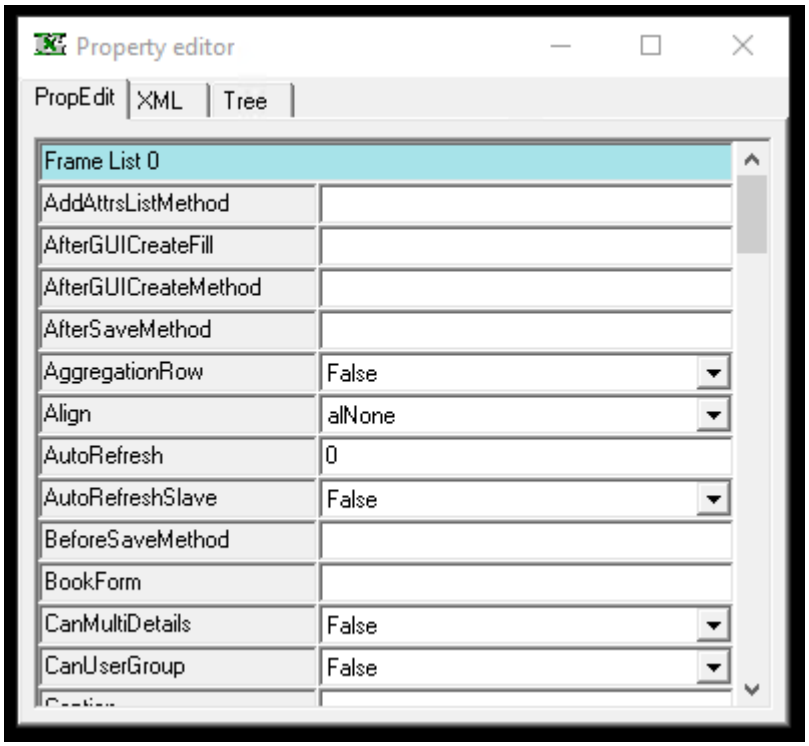

Obrázek 10 XGEN – schématický náhled

Tato schopnost však představuje i problém. Ne u každé komponenty v reálném nasazení má totiž smysl uvažovat o výšce, šířce či pozici. Například komponenta *Attribute*, která reprezentuje sloupec datového seznamu je v aplikaci XGEN libovolně pozicovatelná. V reálném nasazení však pozice nemá význam. Sloupce totiž vždy začínají od vrchu a zleva doprava. XGEN pak tedy zbytečně naplňuje definice formulářů vlastnostmi, které nejsou nijak využity. Designer formulářů pro tyto případy bude mít upravené zobrazování komponent a tyto jejich vlastnosti bude schopen nevyplňovat do výsledné definice. Pro zachování zpětné kompatibility je ale prozatím bude uvádět.

Každou komponentu lze kliknutím označit (červené ohraničení) a editovat její vlastnosti v okně s jejich výčtem. Označenou komponentu lze smazat pomocí klávesy „delete“.

### 3.2.2 Okno s nastavením vlastností

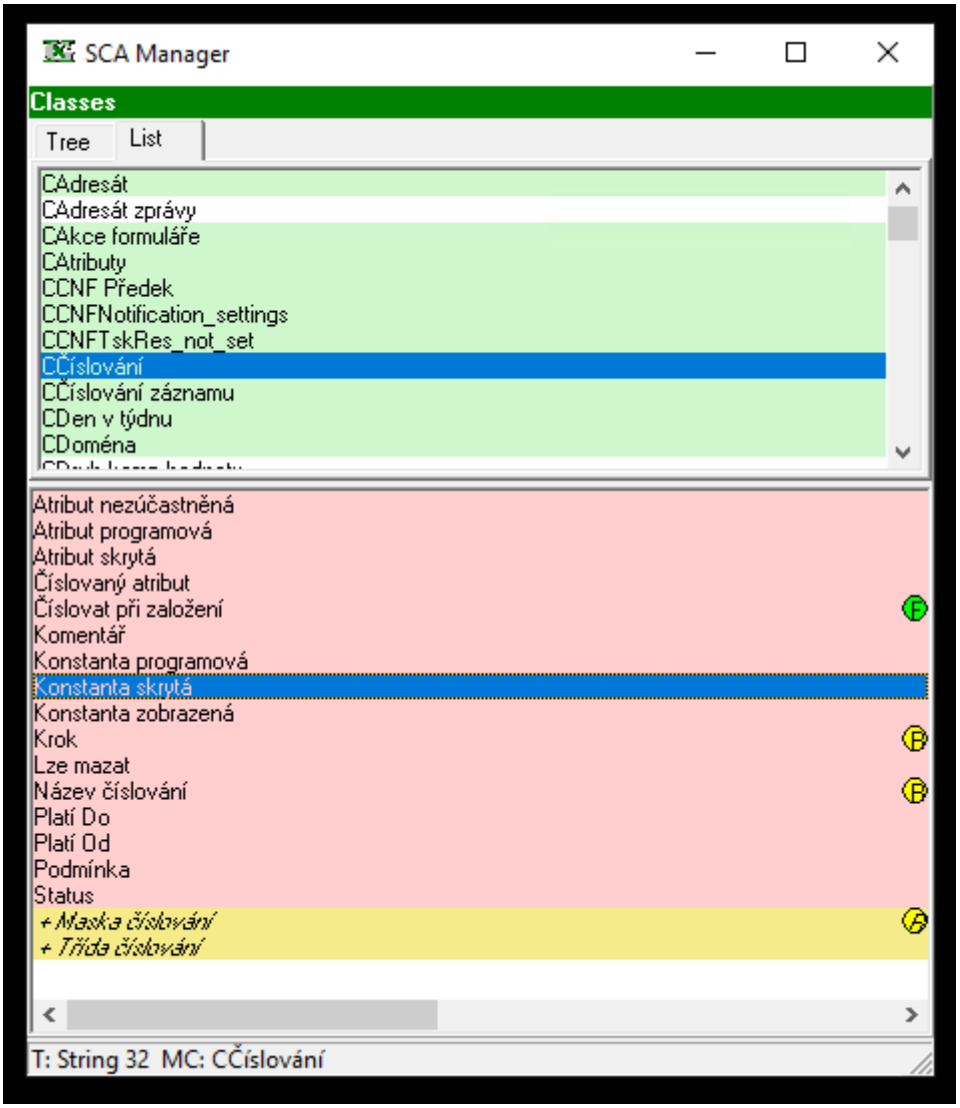
Jedná se o prostou tabulku obsahující výčet všech vlastností právě označené komponenty. V levém sloupci se zobrazují názvy jednotlivých vlastností a v tom pravém jejich hodnoty, které lze přímo editovat. Tabulka na svém vrcholu obsahuje pruh s názvem právě označené komponenty. Komponenty pro editaci jednotlivých hodnot vlastností se liší v závislosti na datovém typu každé vlastnosti.




Obrázek 11 XGEN – okno nastavení vlastností

### 3.2.3 Okno s datovým modelem

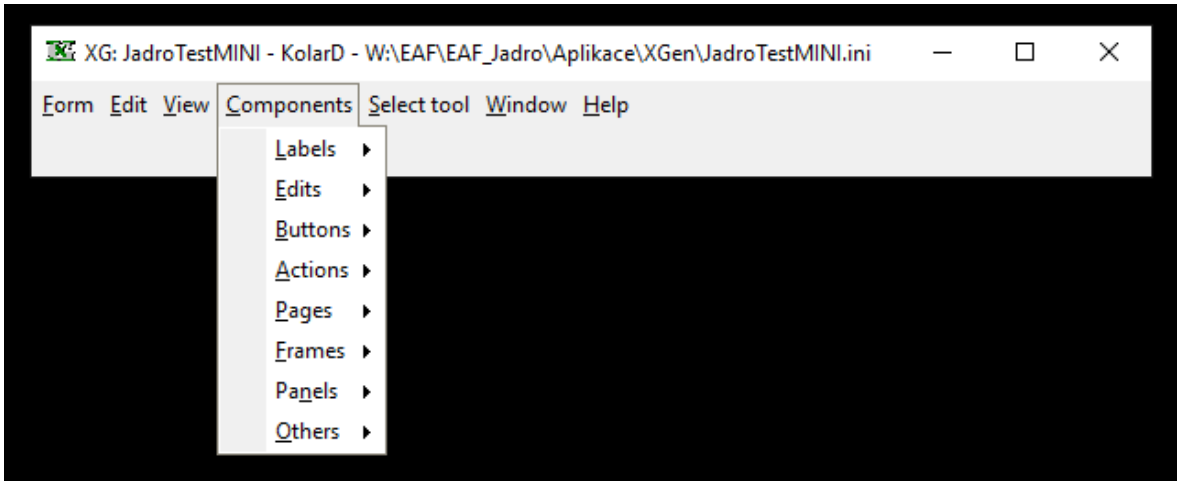
Toto okno slouží pro zobrazování datového modelu zákaznické aplikace, pro kterou jsou vytvářeny formuláře. V horní části je uveden výčet tříd, pro které je možno vytvořit formulář. V dolní části je pak uveden seznam atributů, metod a vazeb označené třídy. Tento seznam je z hlediska dědičnosti kompletní, tedy jsou zobrazovány i prvky z předků označené třídy. Vazby na jiné třídy lze rozkliknout a po této akci dojde k zobrazení stromové struktury, kde je opět uveden seznam všech atributů, metod a vazeb třídy na kterou existuje vazba. Zelené podbarvení u tříd znamená, že tyto třídy již mají přiřazen aspoň jeden formulář.




Obrázek 12 XGEN – okno s datovým modelem

### 3.2.4 Okno s navigačním menu

Okno obsahuje různá menu umožňující přidávání či odebírání komponent, správu zobrazení jednotlivých oken, ukládání či kopírování definice formuláře a další, méně podstatné možnosti.

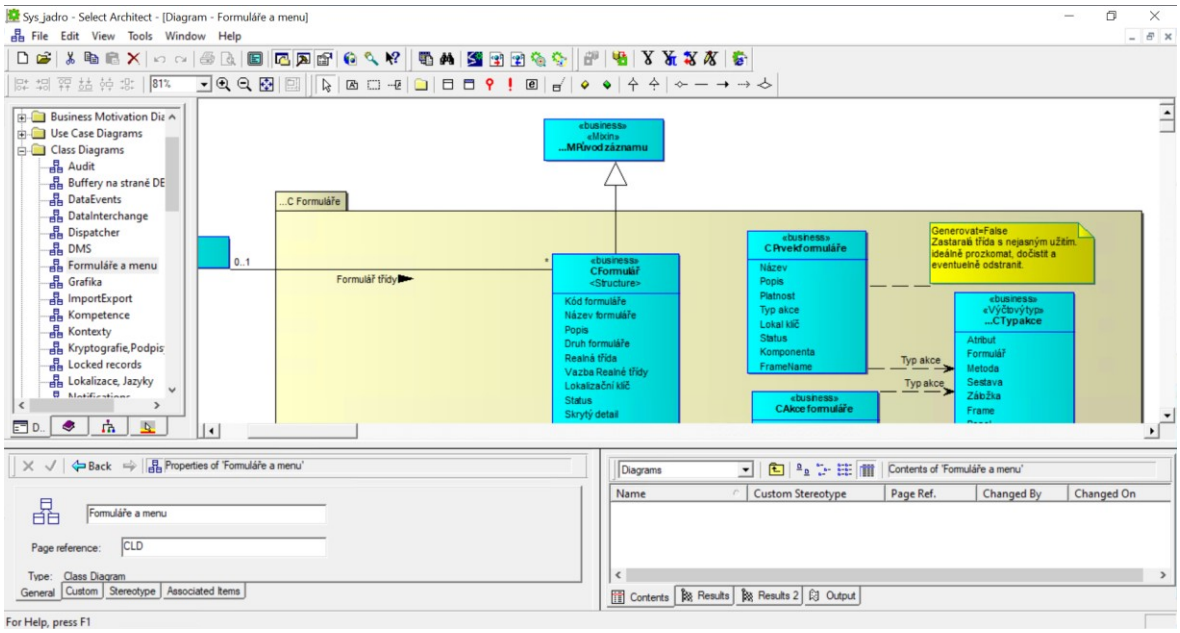



Obrázek 13 XGEN – navigační menu

Rozbalená položka menu „Components“ obsahuje rovněž důležitou informaci o kategorii každé komponenty. Tato informace bude převzata do Designeru formulářů a bude vizuálně lépe znázorněna. Aktuální řešení v aplikaci XGEN pouze prodlužuje dobu výběru, kvůli nutnosti procházet rozbalovacími nabídkami s názvy kategorií. Další nevýhodou tohoto přístupu je, že nově přidanou komponentu nejde při jejím úvodním umístění nijak pozicovat, jsou jí nastaveny výchozí hodnoty pozice (levý horní roh) a až po vložení komponenty může uživatel komponentu dodatečně umístit dle potřeby. Posledním nedostatkem je pak absence jakéhokoliv popisku, který by novým uživatelům nabídl informaci o tom, k čemu daná formulářová komponenta slouží a jaká je její případná podpora v klientských prostředích společnosti.

# 4 SELECT ARCHITECT

Jedná se o CASE nástroj společnosti Select Business Solutions, Inc., zaměřený na modelování celopodnikových procesů prostřednictvím BPMN (Business Process Model and Notation), modelování systémů pomocí UML (Unified Modeling Language) a datové modelování. Kvůli absenci důvěryhodných zdrojů nelze uvést referenci, výrobce již tento produkt na svých stránkách nenabízí.



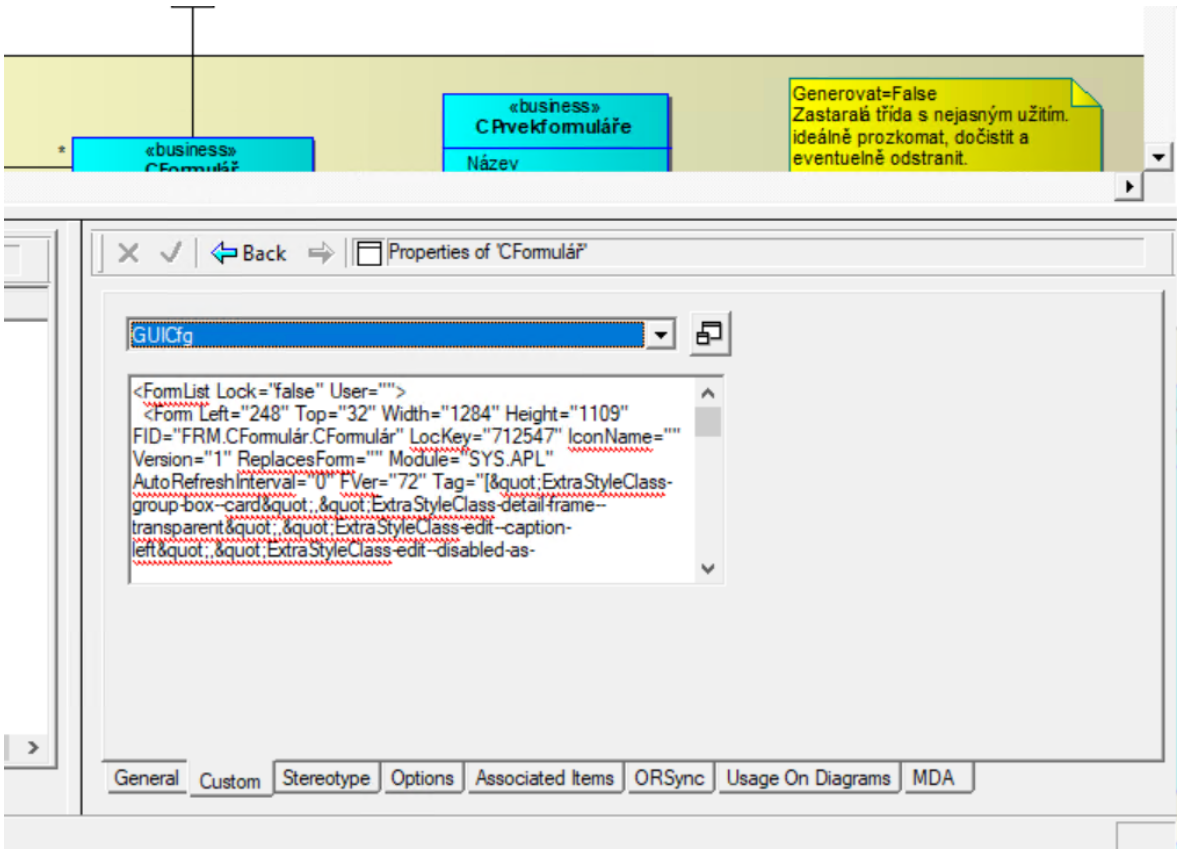

Obrázek 14 Ukázka nástroje Select Architect

V TESCOSW a.s. je využívána převážně jeho schopnost tvorby UML diagramů, dle kterých se automatizovaně vytvářejí skripty pro tvorbu databázových struktur a programových kódů. Tato schopnost automatizace v Model First přístupu tvorby aplikací tkví za finančním úspěchem společnosti, protože jí umožnuje rychle a spolehlivě produkovat aplikace na míru.

S UML diagramy úzce souvisí i formuláře samotné, respektive jejich definice. Select Architect umožňuje vytvářet tzv. custom tags (dále jen „značky“), které umožňují asociovat k různým entitám modelu (třídám, atributům, asociacím, …) libovolný text do velikosti 200 kB. Právě této funkčnosti je využíváno pro ukládání definic formulářů na jednotlivé datové třídy. Každá třída může obsahovat až desítky těchto značek, a to umožňuje asociovat vícero formulářů k jedné třídě.

Logika zápisu do značek funguje tak, že všechny formuláře třídy jsou obaleny elementem FormList. Takto vzniklý dokument se podle své velikosti ukládá do jedné až *n* značek s názvy „GUICfg – GUICfg_*n*“ v programu Select Architect. Obvykle pro uložení všech

definic asociovaných jedné třídě stačí jedna tato značka. Nejsou však neobvyklé ani případy, kdy je vyžadováno tří a více značek.




Obrázek 15 Značka GUICfg u označené třídy

Definice formulářů jsou později spolu s celým modelem obsahujícím UML diagramy vyexportovány do XMI (XML Metadata Interchange) exportu. XMI export je formát založený na XML, který umožňuje výměnu metadat o nadesignovaném modelu mezi různými aplikacemi. Když uživatel provede export, vznikne soubor ve formátu XML, který obsahuje informace o všech třídách a vazbách mezi nimi. Spolu s těmito informace jsou exportovány i značky obsahující definice formulářů. Tyto soubory jsou ukládány do speciálního sdíleného úložiště v rámci firmy, kde další aplikace zpracovávající tyto exporty k nim mají přístup. Jednou z těchto aplikací je i serverová část Portálu vývojáře.

# 5 ANALÝZA KOMPONENT A JEJICH VLASTNOSTÍ

K nalezení množiny všech možných komponent a jejich vlastností bylo zapotřebí analyzovat export těchto údajů z dosavadního nástroje XGEN. Ten umožňuje v přehledné XML struktuře všechny tyto informace vyexportovat do samostatného souboru.

Export je vytvářen pomocí reflexe, kdy se vypisují všechny veřejné vlastnosti všech tříd reprezentujících komponenty. Tento přístup však neobsahoval zcela přesnou reprezentaci skutečného stavu a bylo nutné jej za asistence interních programátorů opravit.

Častým problémem bylo například to, že některé komponenty podle exportu obsahovaly vlastnosti, které ale reálně komponenta nepodporovala. Tento jev byl způsoben nedostatečně řešenou dědičností v rámci třídního návrhu komponent v nástroji XGEN, kdy v jeho grafickém rozhraní bylo explicitně podmíněno, které vlastnosti se mají zobrazovat na bázi toho, zdali vlastnost v programu obsahovala informaci o úložišti (např. *property XGReadVerioned stored false*) [6]. Po zahrnutí této logiky do algoritmu exportu byl tento problém vyřešen.

Dalším problém představovala absence exportu všech možných hodnot těch vlastností, které byly výčtového typu. Jelikož takových vlastností byly desítky, bylo nutné tyto informace do exportu přidat.

Posledním problémem bylo nepřesné pojmenování komponent a vlastností, kdy se do exportu vypisovaly názvy vlastností tříd a názvy tříd samotných tak, jak byly uvedeny ve zdrojových kódech nástroje XGEN. Nikoliv tak, jak se zapisovaly do výsledné XML definice formuláře. Tento problém byl vyřešen tak, že byla vytvořena kolekce dvojic, která mapovala jména ze zdrojových kódů na jména z XML definice. S touto kolekcí se pak při vytváření exportu pracovalo a nahrazovala se tak nesprávná pojmenování.

Výsledkem bylo 106 objevených komponent s více než 400 jedinečnými vlastnostmi. Tato množina dat je zcela kompletní, nicméně mezi požadavky společnosti byla i revize těchto dat (odebrání nepoužívaných nebo duplicitních komponent a jejich vlastností).

## 5.1 Revize komponent

Proces revize komponent probíhal jednak formou diskuze s firemními analytiky, kteří měli představu o komponentách, které již není nutno uvádět v novém Designeru formulářů a jednak analýzou všech definic formulářů zákaznických aplikací v podpoře. Přebytečné komponenty by se daly shrnout do 2 kategorií.

První kategorií jsou duplicitní komponenty, které prvně vznikly pro klientské prostředí Silverlight a po uplynutí doby cca pár let byly nahrazeny jinou komponentou určenou pro klientské prostředí MultiWeb. Klientské kódy sice byly upraveny tak, aby předchozí komponentu považovaly za tu novou, nicméně z aplikace XGEN tyto komponenty nikdy nebyly odstraněny kvůli zpětné kompatibilitě definic. Řešením tohoto problému v Designeru formulářů je prostá ignorace těchto komponent. Respektive při otevření definice se zastaralou duplicitní komponentou dojde k zobrazení chybové hlášky, že definice formuláře obsahuje nepodporované komponenty. Toto řešení bylo zvoleno z důvodu, že je velice malá pravděpodobnost reálného výskytu takovéto komponenty v definicích zákaznických aplikací v podpoře.

Druhou kategorií jsou tzv. pseudo komponenty, což jsou komponenty, které nejsou nijak implementovány v klientském prostředí (MW, SL či LW) a slouží pouze pro potřeby nástroje XGEN. Mezi tyto komponenty se například řadí komponenta *LocalizedText*, která se stará o načítání lokalizací různých komponent z lokalizační databáze ve schématickém náhledu v aplikaci XGEN. Načítání probíhá tak, že komponenta *LocalizedText* je asociovaná ke komponentě obsahující vlastnost s lokalizačním klíčem (ID na záznam v lokalizační databázi). Pro tento klíč načte hodnotu lokalizace a zobrazí ji nad komponentou se kterou je asociována.

Tento způsob práce s lokalizacemi byl nejenom nepřehledný, ale hlavně nekonzistentní, protože v nástroji XGEN již existovaly jiné komponenty, které obsahovaly vlastnosti s lokalizačními klíči a hodnotu této lokalizace si dokázaly načíst samy. Na základě těchto faktů bylo rozhodnuto komponentu odebrat a zanést tuto úpravu i zpětně do aplikace XGEN.

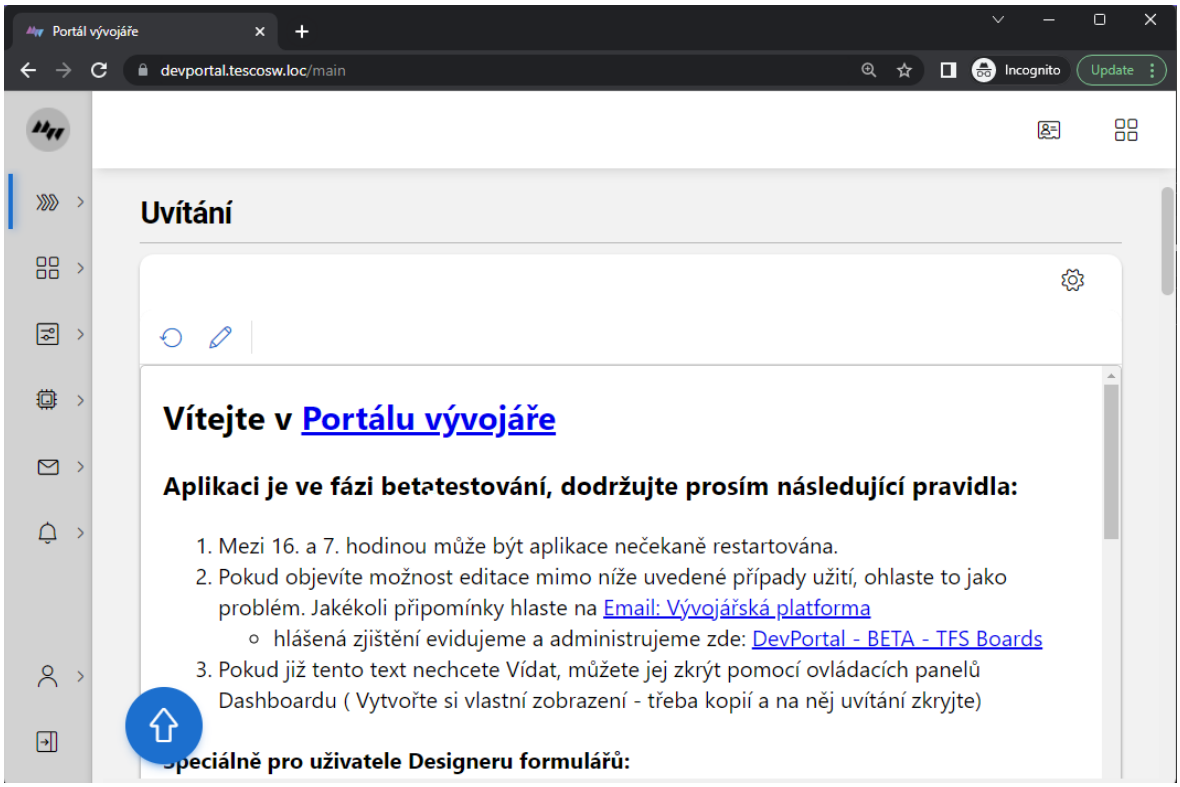
Celkově tak byl zredukován počet komponent ze 106 na rovných 100. Potenciálních adeptů pro redukci se našlo ještě více, bohužel odebrání ještě není možné a těmito návrhy bude zaobíráno znovu, až bude ukončena podpora starého nástroje.

## 5.2 Revize vlastností

Po revizi komponent byla nutná i revize jejich vlastností. Podobně jako u komponent, tak i mezi vlastnostmi se vyskytovaly tzv. pseudo vlastnosti, které sloužily pouze pro potřeby nástroje XGEN. Dalšími potřebami revize bylo vytvoření lokalizovaných popisů jednotlivých vlastností a zjištění, které vlastnosti by měly být evidovány jako povinné.

# 6 APLIKACE PORTÁLU VÝVOJÁŘE

Designer formulářů má běžet v rámci aplikace zvané Portál vývojáře. Jedná se o aplikaci, která je určena interním vývojářům společnosti TESCOSW a.s. a má zjednodušovat proces výroby a vývoje aplikací pro cílové zákazníky. V aktuální podobě aplikace zatím zobrazuje a udržuje seznam všech aplikací vyvíjených společností a jejich datové modely. Právě tyto informace však budou sloužit jako zdroj pro Designer formulářů, který bude spouštěn s parametry v podobě množiny datových modelů a identifikátoru formuláře, jehož definici má upravovat. Portál vývojáře již obsahuje logiku správy uživatelů a přístupových práv k datům díky využití technologických frameworků TEAF a MultiWeb, které jsou detailněji popsány v následující kapitole.




Obrázek 16 Podoba aplikace Portál vývojáře

Do aplikace bude Designer formulářů integrován pomocí možnosti frameworku MultiWeb vkládat do své běžící instance tzv. *CustomControl*, což je proprietární technologie umožňující vložení nestandardních komponent do aplikace. Může se jednat o komponentu formuláře nebo o vlastní implementaci již existující komponenty, která může být definována v externím JavaScript souboru. Pro Designer formulářů bylo využito cesty vlastní implementace komponenty *DetailFrame*. Výsledný formulář spouštějící Designer formulářů vypadá následovně.

```
<Form I="0" FID="FRM.EditorGUI.EditorGUI" FVer="4" LocKey="670055" Module="DP_FormDesigner">
  <Components>
    <Component XGClass="DetailFrame" ... CustomCtrlName="EditorGuiClass" masterFrame="FrameList1" />
    <Component XGClass="ListFrame" ... fName="FrameList1" ObjectName="FDForm" Visible="False">
      <Component XGClass="Attribute" ... LocKey="777934" Name="FormId" />
    </Component>
  </Components>
</Form>
```

Obrázek 17 Definice formuláře Designeru formulářů

V ukázce definice si lze povšimnout pojmenování *EditorGUI*. Toto pojmenování vzniklo v raném vývoji formuláře, kdy se operovalo s různými interními názvy pro náhradu aplikace XGEN a je v plánu toto pojmenování změnit a sjednotit na *FormDesigner*. Nejdůležitější částí je nicméně třetí řádek s komponentou *DetailFrame*, která obsahuje vlastnost *CustomCtrlName.* Ta svou hodnotou určuje identifikátor, pod kterým má MultiWeb načítat externí JavaScript soubor inicializující Designer formulářů. Cesta k souboru musí být posléze uvedena v konfiguračním souboru klientské části zvaného *initParams.json*.

Designer formulářů je tedy do aplikace Portálu vývojáře integrován pomocí klasického formuláře, který obsahuje vlastní implementaci komponenty *DetailFrame*.

# 7 POPIS POUŽITÝCH TECHNOLOGIÍ A KNIHOVEN

Následující kapitola stručně popisuje technologie a knihovny použité při vytváření Designeru formulářů. Je rozdělena do dvou částí, a to na technologie, které jsou vyvinuty společností TESCOSW a.s. a technologie veřejně dostupné.

## 7.1 Technologie vyvinuté společností TESCOSW a.s.

Uvedený výčet reprezentuje pouze technologie využité v rámci tvorby Designeru formulářů. Nejedná se o kompletní výčet technologií vyvinutých společností.

### 7.1.1 MultiWeb

Jedná se o technologický framework pracující na technologiích HTML5, TypeScript a LESS. Pod tímto názvem jsou zahrnuty tři dílčí prvky. Framework je využíván mnoha úseky napříč firmou a aktuálně se jedná o vlajkovou loď společnosti. Vývoj započal v roce 2015 a do budoucna se plánuje hlubší integrace knihovny React a rozpad na drobnější samostatné části, které usnadní znovupoužitelnost knihoven.

Jako první je to webová aplikace, která se interně označuje jako klient. Jedná se o jednostránkovou aplikaci (SPA – Single Page Application). Tyto aplikace fungují tak, že uživatel se pohybuje stále na jedné stránce, které je dynamicky měněn obsah na základě uživatelových akcí. Nedochází tak k žádným přesměrováním, načítají se data namísto celých HTML stránek či jejich částí a není potřeba přenačtení. Další výhodou jednostránkových aplikací je jejich schopnost poskytovat plynulý uživatelský zážitek srovnatelný s nativní desktop aplikací. Nevýhodami těchto aplikací však může být redukovaná optimalizace pro vyhledávače (SEO – Search Engine Optimization) právě kvůli dynamickému načítání obsahu. Vyhledávače jej totiž nemusí indexovat. Jejich vývoj je rovněž náročnější oproti vícestránkovým aplikacím.

Druhou částí frameworku MultiWeb jsou knihovny, které jsou primárně vyvinuty pro funkci klienta. Mohou ale být využity i v jiných aplikacích, které budou vyžadovat např. pouze funkce pro komunikaci se serverem. Knihovny obecně poskytují prezentační komponenty pro zobrazování a editaci dat, dále rozhraní pro komunikaci s API webové služby TEAF, funkce či prostředky pro práci s kryptografickými knihovnami či další rozhraní pro komunikaci s jinými službami vytvořenými společností TESCOSW a.s.

Třetí částí jsou pak podpůrné součásti potřebné pro vývoj. Mezi ně se řadí dokumentace, která je ve formě Wiki stránek, částečně generovaných ze zdrojových kódů. Dále je to webová aplikace Storybook, která slouží pro náhled a příklady použití prezentačních komponent. MultiWeb je jedním ze tří klientských prostředí, které se v rámci společnosti využívají.

### 7.1.2 TEAF Server

TESCOSW Enterprise Application Framework (TEAF) je backendový balíček služeb usnadňující tvorbu formulářových aplikací. Ve třívrstvé architektuře aplikací se stará o databázovou a aplikační vrstvu. Tím se liší od frameworku MultiWeb, který se zaměřuje na prezentační vrstvu.

Databázová vrstva umožňuje perzistentní uložení dat za pomocí technologií MSSQL nebo Oracle. Jiné databázové systémy zatím nejsou podporovány. Databázové struktury jsou z naprosté většiny automaticky generovány z UML modelů a je tak využito MDD (Model Driven Development). Pro aktualizaci těchto struktur jsou k dispozici automatické nástroje, které z rozdílů starého a nového UML modelu jsou schopny vytvořit rozdílové SQL skripty. Tyto nástroje jsou schopny vývojáře upozornit na rizikové operace, jako je smazání sloupců či celých tabulek.

Aplikační vrstva představuje jádro celé aplikace a obsahuje veškerou business logiku. Je naprogramovaná pomocí technologie ASP.NET a provádí se zde veškeré výpočty a operace s daty. Zároveň se jedná o prostředníka mezi databázovou a prezentační vrstvou. Pro komunikaci s aplikační vrstvou se využívá její SOAP rozhraní. Aplikační vrstva rovněž umožňuje vložení vlastních aplikačních modulů, které obsahují business logiku pro konkrétní potřeby aplikace. Této funkčnosti bude využito v serverové části Designeru formulářů.

## 7.2 Veřejně dostupné využité technologie

Vzhledem k požadavkům na podobu Designéru formulářů, které určovaly, v jakých technologiích má být vyvinut, nebylo možné volit alternativy k následujícímu výčtu využitých technologií.

### 7.2.1 TypeScript

TypeScript je open-source programovací jazyk vyvinutý firmou Microsoft, který rozšiřuje funkčnost jazyka JavaScript o silnější typovou kontrolu a další funkce, které jej činí vhodnějším pro použití v rozsáhlejších projektech. TypeScript je nadmnožinou jazyka JavaScript a platí, že každý JavaScriptový kód je validním TypeScriptovým kódem. Jednou z jeho nejdůležitějších funkcí je statická typová kontrola, umožňující definovat datové typy proměnných, funkčních parametrů a návratových hodnot tak, aby kompilátor mohl detekovat případné chyby přiřazení hodnot proměnným nekompatibilního datového typu. TypeScript je překládán do běžného JavaScriptu, takže může běžet na každé platformě podporující JavaScript [7].

### 7.2.2 React

Jedná se o open-source JavaScriptovou knihovnu pro vytváření znovupoužitelných prvků uživatelského rozhraní ve formě komponent. React je vyvíjen společností Facebook a je využíván pro tvorbu webových a mobilních aplikací. Komponenty jsou v Reactu vytvářeny pomocí rozšířené syntaxe JavaScriptu zvané JSX. Toto rozšíření umožňuje psát kód, který je kombinací HTML a JavaScriptu v jednom souboru, což usnadňuje tvorbu komponent a zlepšuje přehlednost kódu [8].

Díky těmto vlastnostem je tedy React vhodným nástrojem pro tvorbu Designeru formulářů, a to nejen pro tvorbu formulářových komponent samotných, ale i pro komponenty designeru jako takového. Tedy např. editační okna, záložky, tlačítka, …

### 7.2.3 Redux

Pro správu stavu aplikace byla vybrána JavaScriptová knihova Redux, která je určena pro použití s Reactem (případně jinými knihovnami pro uživatelská rozhraní). Hlavním důvodem pro tuto volbu bylo to, že tato knihovna se již využívá v MultiWebu a její implementace pro Designer formulářů by tedy byla jednodušší.

V Reduxu je stav aplikace uložen v centrálním úložišti zvaném store. Jednotlivé komponenty aplikace mohou k tomuto stavu přistupovat pomocí tzv. selektorů, což jsou funkce vracející určitou část stavu (např. pole označených prvků, jméno přihlášeného uživatel, …). Ke změně stavu je využíváno tzv. akcí, což jsou jednoduché objekty popisující, co se v aplikaci stalo a případně obsahující novou hodnotu daného podstavu. Poslední

komponentou jsou pak reducery, což jsou funkce, které reagují na vyvolané akce starající se o příslušnou změnu stavu [9].

Reducer také podporuje middleware funkce, které lze použít k volání asynchronních funkcí, jako např. volání API. Middleware funkce jsou volány před tím, než akce dorazí do reduceru. Těchto funkcí využijeme pomocí knihovny Redux-Saga, která je popsána v kapitole níže.

### 7.2.4 Redux-Saga

Jedná se o JavaScriptovou knihovnu pro správu vedlejších efektů aplikací využívajících knihovnu Redux. Vedlejšími efekty jsou asynchronní operace, jako např. dotazy na server, časovače či manipulace s funkcemi prohlížeče. K tomuto využívá Redux-Saga tzv. generátory což jsou funkce, které umožňují pozastavení průběhu funkce, předání kontroly jiné funkci a následnému pokračování od místa, kde se funkce pozastavila. Pro předání kontroly se používá klíčového slova „yield“ [10].

### 7.2.5 ASP.NET

Pro serverovou část aplikace byl zvolen framework ASP.NET. Vzhledem k tomu, že tento framework využívá i TEAF Server, jehož funkčnosti musí být dle zadání využito, nebyla zvažována jiná alternativa. ASP.NET je open-source webový Framework vytvořený společností Microsoft. K datu tvorby diplomové práce je ve společnosti TESCOSW a.s. využíván ve verzi 4.8 [11].

# II. PRAKTICKÁ ČÁST

# 8 ARCHITEKTURA DESIGNERU FORMULÁŘŮ

V následující kapitole bude rozebrána architektura frontend a backend části aplikace Designeru formulářů, kdy pod frontend část spadají ty zdrojové kódy, které budou exekuované ve webovém prohlížeči a pod backend část ty kódy, které budou běžet na serveru v procesu spadajícím pod celkovou aplikaci Portálu vývojáře a hostovaném v aplikaci IIS (Internet Information Service).

## 8.1 Popis architektury frontend části

Vzhledem k povaze knihovny React je značná část frontend kódu psána funkcionálně. Objektově orientovaného přístupu se využívá pouze minimálně a v celém zdrojovém kódu se vyskytuje pouze pár tříd, které jsou potřebné kvůli integraci s frameworkem MultiWeb.

V této kapitole budou probrány nejdůležitější prvky zdrojových kódů jako je integrace do frameworku MultiWeb, hlavní části stromu React komponent, vykreslovací modul Picasso, moduly se ságami, moduly s reducery a modul s konfigurací aplikace.

### 8.1.1 Integrace do MultiWeb klienta

Integrace probíhá v souboru *index.ts*, kde je vytvořena událost pro registraci komponent Designeru formulářů. Do klienta jsou zaregistrovány moduly reduceru a ságy spolu s factory funkcí pro tvorbu instance třídy FormDesigner, která slouží pro načtení Designeru formulář v klientovi. Tento soubor je zároveň i vstupním bodem exekuce aplikace.

TS index.ts M ●

FormDesigner > TS index.ts > ...

```
1   import { ReduxRegistrator } from "UI/UI.React/customization/index";
2   import { MainForm } from "UI/UI/DefaultInterface/MainForm";
3   import { reducer } from "./src/reducers";
4   import { saga } from "./src/saga";
5   import { FormDesigner } from "./src/FormDesigner";
6   import { STORE_NAME } from "./src/constants";
7   import { Registrator } from "UI/UI/UI/Controls/Registrator";
8
9   MainForm.registrations.add(() => {
10      ReduxRegistrator.addReducer(STORE_NAME, reducer);
11      ReduxRegistrator.addSaga(saga);
12      Registrator.add("EditorGuiClass", FormDesigner); // Registrace CustomControl
13  });
14
```

Ukázka kódu 1 Registrace Designeru formulářů do MultiWeb klienta

### 8.1.2 Struktura stromu React komponent

- FormDesigner
    - DataSourcePanel
    - HierarchyView
    - Toolbar
    - Workbench
        - PicassoEditor
        - XmlView
        - PreviewWindow
    - PropertyPanel
    - ComponentsPanel

Diagram výše popisuje základní strukturu stromu React komponent v aplikaci. Ve zdrojových kódech je tato struktura poněkud komplexnější a obsahuje více mezikomponent, které se starají o napojení na data z Redux storu.

### 8.1.3 Modul picasso.ts

Pro zobrazení definice formuláře ve schematickém náhledu bylo potřeba vytvořit modul, který umožní vykreslování na plochu, jež bude plynulé při posouvání komponent formuláře a zároveň bude umožňovat komponenty zvětšovat, respektive zmenšovat. Za tímto účelem byl vytvořen modul Picasso.

Co se množství komponent týká, tak se jedná o poměrně malý modul, obsahující dvě React komponenty (nepočítáme komponenty, které se zabývají napojením ne Redux). První komponentou je *PicassoPaintingComponent.tsx*, která obsahuje logiku vykreslování plátna, které je možné posouvat pravým tlačítkem myši a zároveň lze toto plátno přibližovat a oddalovat pomocí kolečka myši. Přibližování je vytvořeno tak, že kurzor myši slouží jako těžiště přiblížení a daný bod, nad kterým byl kurzor před přiblížením, zůstává ukotven na svém původním místě (obdobně, jako například v mapách od společnosti Google). Druhou významnou komponentou je pak komponenta *MoveableComponent.tsx*, která se stará o vykreslování každé dílčí komponenty (v případě Designeru formulářů každé formulářové komponenty) a rovněž o vykreslování úchopů pro zvětšování a zmenšování

komponenty. Každá instance *MoveableComponent* je schopna renderovat komponentu podle jiného stylu.

### 8.1.4 Modul reducer.ts

Modul řídící podobu datového stavu aplikace. Reaguje na příchozí akce a podle jejich typu a vstupních hodnot mění adekvátně stav. V tomto modulu se můžou vyskytovat pouze takové algoritmy, které mění stav přímo, neobsahují žádné vedlejší efekty a nevyžadují asynchronních událostí (např. načítání dat z databáze). Pokud některý z těchto prvků je potřeba, využívá se modulu *saga.ts*. Typ popisující datový stav aplikace (Redux store) se vyskytuje v souboru */src/types.ts*.

TS types.ts M ×

FormDesigner > src > TS types.ts > ...

```
57  export interface IFormDesignerInstanceState {
58      /** Stav s grafickou reprezentací komponent. */
59      readonly picasso: IPicassoState;
60      /** Stav s datovou reprezentací komponent. */
61      readonly xgenComponents: readonly IXgenComponent[];
62      /** Přepínač určující, zdali je aplikace ve fullscreen režimu. */
63      readonly isFullscreen: boolean;
64      /** Popisuje, zdali byl formulář editován. */
65      readonly touched: boolean;
66      /** Stav wizard okna. */
67      readonly isWizardVisible: boolean;
68      /** Určuje, zdali instance FormDesigneru byla otevřena přes PutFormInPanel. */
69      readonly isModalChild: boolean;
70      /** Hodnota aktuálního nastavení zobrazení. (XS, S, M, ...) */
71      readonly generalSizeDefinition: SizeOptions;
72      /** Přepínač určující, zdali je ručně editována definice formuláře.  */
73      readonly definitionEditMode: boolean;
74      /** Načtené lokalizace formuláře. */
75      readonly formLocalizations: TypedObject<ILocalization>;
76      /** Hodnoty filtračních řádků v jednotlivých záložkách. */
77      readonly searchKeywords: TypedObject<string>;
78      /** Určuje, který panel je aktivní. (schéma, definice, náhled) */
79      readonly workbenchMode: WorkbenchMode;
80      /** Stav dialogového okna pro uživatelské potvrzení akcí. */
81      readonly userResponseDialogState: UserResponseDialogState;
82      /** Označená třída v záložce Model. */
83      readonly highlightedClass: IHighlightedClass;
84      /** Stav s daty o datovém modelu záložky Model. */
85      readonly dataSourceModel: IDataSourceModel;
86      /** Popisuje cestu stromem komponent, která má být ve schématu zaměřena. */
87      readonly requestedComponentsPath: readonly number[];
88  };
```

Ukázka kódu 2 Typ popisující datový stav aplikace (Redux store)

Na tomto typu je patrné, že každá jeho vlastnost je nastavená jako *readonly*. Je to z důvodu toho, na jakém principu je založena knihovna Redux, kdy komponenty v aplikaci jsou překresleny v případě, kdy dojde k jakékoliv změně stavu. Aby ale došlo k detekci změny, tak objekty nesmí být tzv. *mutable* (proměnlivé), musí být *immutable* (neměnné). Je to z toho důvodu, že Redux pro detekci změny využívá mělké porovnávání, kdy u dvou objektů dochází pouze k porovnávání jejich referencí, nikoliv hodnot jejich vlastností. Právě z toho důvodu jsou všechny objekty označeny jako *readonly*, protože pro jejich „změnu“ musí dojít k vytvoření zcela nového objektu s novou referencí.

Ve stejném souboru jsou definovány i typy pro akce. Akcemi se rozumí objekty popisující konkrétní událost, která v aplikaci nastala a na kterou Redux store nějakým způsobem může reagovat.

types.ts M

FormDesigner > src > types.ts > IInitPicassoHostAction

```
341
342  export interface IUpdateXgenComponentPropertyAction extends IPicassoHostAction {
343      readonly payload: {
344          readonly index: number;
345          readonly propertyName: string;
346          readonly newValue: string;
347      };
348  }
349
350  export class IUpdateXgenComponentPropertyAction {
351      public static [Symbol.hasInstance](instance: any) {
352          return instance != null && instance.type === UPDATE_XGEN_COMPONENT_PROPERTY;
353      }
354  }
```

Ukázka kódu 3 Ukázka definice jedné z akcí

Na ukázce si lze povšimnout zvláštnosti v podobě toho, že akce se skládá ze dvou částí, a to z rozhraní a stejnojmenné třídy. Důvod pro to je ten, že TypeScript neumožňuje použít operátor *instanceof* s rozhraními. Problém byl tedy vyřešen tím, že akce je definována nejen rozhraním, ale i třídou, která má implementovánu logiku určující, zdali je nějaký objekt instancí dané třídy. Tento princip je v jazyce TypeScript pojmenován jako slučování deklarací [12]. V reduceru pak použití toho principu lze vidět na následujícím obrázku.

reducers.ts

FormDesigner > src > reducers.ts > isFullscreen

```
26  const xgenComponents: Reducer<readonly IXgenComponent[]> = (state = EMPTY_ARRAY, action) => {
27      if (action instanceof ILoadXgenComponents)
28          return action.payload.xgenComponents;
29
30      if (action instanceof IUpdateXgenComponentAction)
31          return produce(state, (draft: any) => {
32              const { index, newXgenComponent } = action.payload;
33              draft[index] = newXgenComponent;
34          });
35
36      if (action instanceof IUpdateXgenComponentPropertyAction)
37          return produce(state, (draft: any) => {
38              const { index, propertyName, newValue } = action.payload;
39              draft[index].properties[propertyName].value = newValue;
40
41              if (propertyName === ENUM_NAME)
42                  draft[index].properties[HIDE_VALUES].value = "";
43          });
44
45      return state;
46  };
```

Ukázka kódu 4 Příklad použití *instanceof* v reduceru

### 8.1.5 Modul saga.ts

V tzv. ságách probíhá veškerá logika, která vyžaduje použití vedlejších efektů, případně asynchronní načítání či zpracování dat. Podobně jako reducer reaguje na vyvolané akce, tak i ságy můžou reagovat na vyvolání těch samých akcí. V Designeru formulářů je ság využíváno primárně pro načtení dat ze serveru potřebných pro zobrazení formuláře (např. lokalizace, našeptávače pro hodnoty některých vlastností komponent, …). Rovněž v nich je obsažena i složitější logika při manipulaci s komponentami (mazání, přidávání, kopírování) a logika pro zpracování uživatelských odpovědí pro potvrzování akcí. V ságách je navíc možné přistupovat i ke všem hodnotám uložených Redux storu.

saga.ts M

FormDesigner > src > saga.ts > ...

```
/** Postará se o uložení definice formuláře. */
export const saveFormDefinitionSaga = function* saveFormDefinitionSaga(action: ISaveFormDefinitionAction): any {
    const { hostKey } = action.meta;

    const formManagerInstance = yield call(getFormManagerInstance, hostKey);
    const formXgenComponent = yield select(getXgenComponent, hostKey, FORM_PICASSO_INDEX);
    const currentFverValue = formXgenComponent.properties[F_VER].value
        || formXgenComponent.properties[F_VER].defaultValue;

    yield put(createUpdateXgenComponentPropertyAction(
        hostKey, FORM_PICASSO_INDEX, F_VER, (Number(currentFverValue) + 1).toString())
    );

    const successfulSave = yield call(formManagerInstance.saveFormObject);

    if (successfulSave)
        yield put(createSetIsTouchedAction(hostKey, false));
    else
        yield put(createUpdateXgenComponentPropertyAction(hostKey, FORM_PICASSO_INDEX, F_VER, currentFverValue));
};
```

Ukázka kódu 5 Příklad ságy pro uložení definice formuláře

### 8.1.6 Modul konfigurace

V tomto modulu je obsažena veškerá konfigurace komponent formuláře a jejich vlastností. Je zde využito pokročilých typů v rámci TypeScriptu, které zajišťují, že povinné informace o komponentách či vlastnostech jsou vyplněny a zároveň zabraňují přiřazení nesprávného datového typu hodnotě. Konfigurace je řešena kompozicí, nikoliv dědičností. Každá konfigurace komponenty obsahuje kompozitní objekt složený z jednotlivých konfigurací vlastností. Vlastnostem je rovněž umožněno přetěžování hodnot jejich konfigurace. Například defaultní hodnota vlastnosti může být na jedné komponentě jiná než na druhé komponentě. Každá komponenta i vlastnost ve své konfiguraci obsahují lokalizační klíč, odkazující na lokalizovaný popis daného prvku.

Všechny vlastnosti jsou definovány v souboru *propertiesConfig.ts*, kde některé vlastnosti jsou shlukovány do jednoho společného objektu v případech, kdy spolu úzce souvisí a neexistuje komponenty, která by všechny tyto vlastnosti nevyžadovala.

propertiesConfig.ts

FormDesigner > src > xgenConfiguration > config > propertiesConfig.ts > ...

```
1306  export const SHOWING_PROPERTIES: ReadonlyRecord<ShowingPropsNames, ComponentProperty> = {
1307      ShowIf: {
1308          datatype: "condition",
1309          defaultValue: "",
1310          kind: "Behavior",
1311          isForExperts: false,
1312          required: false,
1313          locKey: LOC_PROP_SHOW_IF,
1314      },
1315      ShowIfDefault: {
1316          datatype: "boolean",
1317          defaultValue: "False",
1318          kind: "Behavior",
1319          isForExperts: false,
1320          required: false,
1321          locKey: LOC_PROP_SHOW_IF_DEFAULT,
1322      },
1323  };
```

Ukázka kódu 6 Příklad definice konfigurace vlastností

Každý takovýto objekt, ať už obsahuje jednu či více definic vlastností, je pak referencován v konkrétních konfiguracích komponent. Příklad takové konfigurace lze pozorovat na následujícím obrázku. Kde objekt *props* definuje množinu vlastností, s jakými komponenta pracuje a objekt *PagesConfig* tento objekt konzumuje a přidává metadata o komponentě.

TS PagesConfig.ts ×

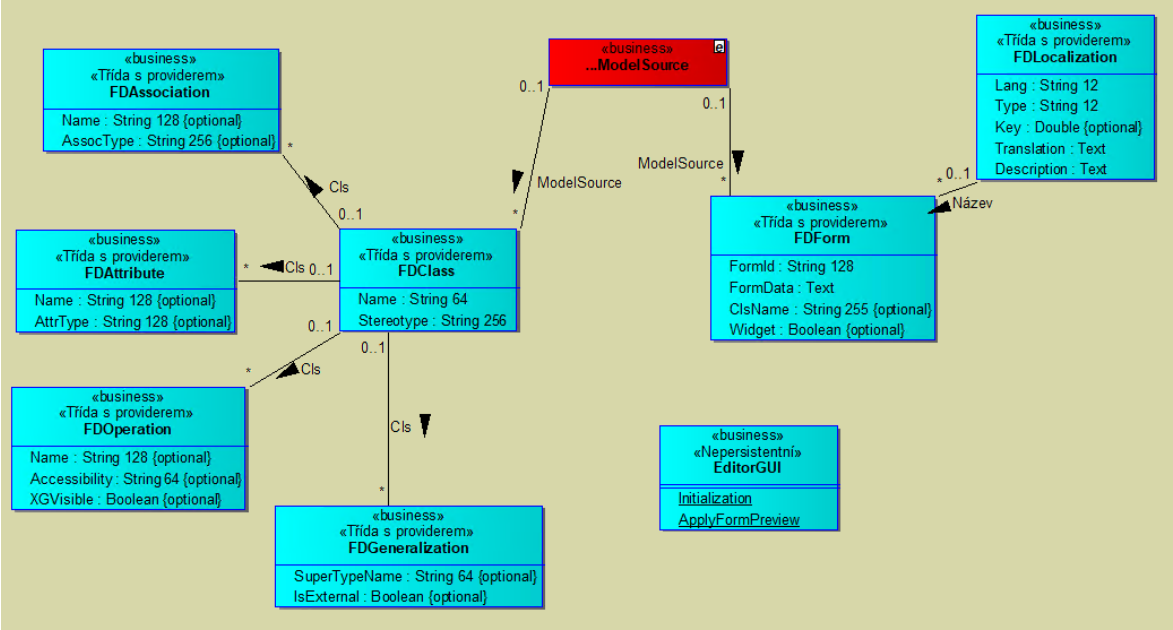
FormDesigner > src > xgenConfiguration > config > componentsConfigs > TS PagesConfig.ts > ...

```
8   const props: ReadonlyRecord<string, ComponentProperty> = {
9       ...BASE_PROPERTIES,
10      ...BASE_FORMLESS_PROPERTIES,
11      ...CUSTOM_CTRL_NAME_PROPERTY,
12      ...STYLEABLE_PROPERTIES,
13      ...READ_ONLY_PROPERTIES,
14      ...ALIGN_PROPERTY,
15      ...SHOWING_PROPERTIES,
16      ...CHECK_VISIBILITY_BY_CHILD_PROPERTY,
17      ...SHOW_TAB_STRIP_PROPERTY,
18      ...STYLEABLE_PROPERTIES,
19      Height: {
20          ...(BASE_PROPERTIES.Height),
21          defaultValue: "150",
22      },
23      Width: {
24          ...(BASE_PROPERTIES.Width),
25          defaultValue: "300",
26      },
27  };
28
29  export const PagesConfig: XgenComponentConfiguration = {
30      name: PAGES,
31      locKey: LOC_COMP_PAGES,
32      location: "DetailFrame, Konteinery",
33      kind: "Konteiner",
34      compatibility: { sl: "Ano", lw: "Ano", mw: "v12.01" },
35      iconCode: IconCode.docPaperStack,
36      category: CATEGORIES.Pages,
37      properties: props,
38  };
39
```

Ukázka kódu 7 Příklad definice konfigurace komponent

## 8.2 Popis architektury backend části

Backendová část, na rozdíl od frontendové, je psaná čistě pomocí objektově orientovaného paradigmatu. Základní struktury jsou automaticky generovány z UML modelu, který byl v rámci této diplomové práce vytvořen. Automatické generování probíhá pomocí nástroje vyvinutého společností TESCOSW zvaného ProcGen. Hlavními částmi jsou C# projekty DevPortal_FormDesigner a ExpertDataProvider.




Obrázek 18 UML diagram projektu DevPortal_FormDesigner

### 8.2.1 Integrace do TEAF

Pro začlenění business logiky Designeru formulářů je potřeba využít funkčnosti načtení aplikačních modulů v aplikační vrstvě frameworku TEAF. Načtení probíhá pomocí reflexe při inicializaci procesu serveru a probíhá tak, že dojde k načtení všech „.dll“ souborů v umístění binárních souborů serverové aplikace a následně se pro každý takový „.dll“ soubor zjišťuje, zda se jedná o aplikační modul. Aby „.dll“ soubor byl aplikačním modulem, musí mít v sobě zakompilovanou třídu s názvem *Registrator*, která implementuje rozhraní *IModuleRegistrator<IBLEntitiesRegistrator>*.

### 8.2.2 Projekt DevPortal_FormDesigner

V tomto projektu jsou nadefinovány třídy objektového modelu potřebné pro fungování Designeru formulářů. Rovněž je zde naprogramována validační logika spouštěná před uložením záznamu, která hlídá, že definice formuláře je validním XML souborem, že je formulář korektně přiřazen ke své datové třídě a že jednotlivé komponenty formuláře mají uvedeny všechny povinné vlastnosti, případně, že vlastnosti obsahují validní hodnoty. Správnost vlastností a komponent je validována vůči konfiguračnímu JSON souboru, který vzniká při aktualizaci firemní Wiki dokumentace ze zdrojových kódů Designeru formulářů.

### 8.2.3 Projekt ExpertDataProvider

ExpertDataProvider je projekt sloužící pro komunikaci Designeru formulářů s aplikací Select Architect, do které jsou ukládány definice formulářů. V projektu je řešena logika čtení a ukládání definic do jednotlivých tříd datového modelu, se kterým je právě editovaný formulář asociován.

Komunikace s aplikací Select Architect probíhá pomocí Automation mechanizmu (dříve OLE Automation), který využívá COM (Component Object Model) rozhraní. COM je standardizovaný binární interface vytvořený společností Microsoft, který umožňuje meziprocesovou komunikaci a který je využíván právě mechanizmem Automation [13]. Získání objektu pro komunikaci s aplikací Select Architect je pak znázorněno na následující ukázce kódu.

```
Type modelsType = Type.GetTypeFromProgID("SCA.Models");
dynamic scaModelsObejct = Activator.CreateInstance(modelsType);
```

Ukázka kódu 8 Vytvoření komunikačního objektu se Select Architect

Tento objekt pak umožňuje čtení všech informací o datovém modelu, a hlavně čtení a zápis definice formulářů.

# 9 AUTOMATIZACE TVORBY DOKUMENTACE

Po vybudování celistvé konfigurace formulářových komponent a jejich vlastností bylo zapotřebí vytvořit automatizovaný systém, který by v přehledné formě průběžně aktualizoval firemní Wiki dokumentaci. Bylo rozhodnuto, že pro každou Wiki stránku reprezentující formulářovou komponentu budou vytvářeny tabulky, které budou obsahovat výčet všech vlastností dané komponenty, její lokalizovaný popis, kód lokalizace, datový typ, defaultní hodnotu a podporu v rámci klientských prostředí SL, LW a MW.

Pro pravidelné nahrávání této dokumentace bylo využito možnosti aplikace Azure DevOps Server vytvářet tzv. „pipeline“, které poskytují možnost spouštět v rámci repositáře sekvenci skriptů, pracujících s daty repositáře. Za tímto účelem byl tedy vytvořen skript, který načte objekt konfigurace, převede ho na soubor typu „json“, tento soubor je pak přetransformován na tabulky ve formátu, který je čitelný pro Wiki a následně pomocí API, které nabízí firemní Wiki stránka, jsou tyto tabulky nahrány na příslušné stránky konfigurace.

Toto je stručný popis fází procesu tvorby generované dokumentace, každá z těchto fází bude detailně popsána v následujících odstavcích.

## 9.1 Azure DevOps Server

Platforma, dříve známá pod názvem Team Foundation Server, usnadňující spolupráci týmů nad korporátními projekty [14]. Umožňuje správu projektů a jejich verzování, průběžnou integraci, testování a nasazení. Azure DevOps Server umožňuje týmům hostovat vlastní server ve vlastním prostředí a rovněž nabízí sadu služeb jako Azure Boards pro správu požadavků, Azure Repos pro správu repositářů obsahujících zdrojové kódy projektů, Azure Test Plans pro plánování a vyhodnocování testů nad projekty a mnohé další. Významnou službou je i Azure Pipelines, která je využita v procesu automatické tvorby dokumentace.

Azure Pipelines nabízí flexibilní a přizpůsobitelný způsob, jakým sestavovat a vydávat aplikace [15]. Podporuje široké množství technologií a jazyků včetně .NET, Java, Python, Node.js a dalších. Azure Pipelines těchto cílů dosahují prostřednictvím tzv. „pipeline“, které se dají označit jako sekvence kroků, které se mají vykonat při určité události (nahrání nových úprav od programátora, nastání určitého času, nedostupnost některé webové služby, …).

Právě této „pipeline“ bylo využito i pro tvorbu dokumentace. Byla nastavena tak, aby se spustila každý pracovní den v 5 hodin ráno a vykonala určitou sekvenci kroků, které povedou k nahrání výsledné dokumentace na interní Wiki stránky. Mezi kroky této sekvence patří

stažení závislostí projektu Designeru formulářů, spuštění skriptu pro vytvoření JSON souboru s definicí konfigurace a posledním krokem je spuštění programu pro transformaci konfigurace na jednotlivé Wiki tabulky a jejich následné nahrání na server.

## 9.2 Skript pro převod z JS na JSON

Jelikož je modul s konfigurací formulářových komponent psaný v jazyce TypeScript, je zapotřebí jej dostat do podoby, která je jednoduše zpracovatelná jinými programy. Zvolen byl formát JSON, který je jednoduše vytvořitelný z objektu reprezentujícího konfiguraci v rámci běhového prostředí Node.js.

```
31  console.log("Creating xgenComponents.json...");
32  var xgenConfiguration = require('./build/xgenConfig/index');
33  var data = JSON.stringify(xgenConfiguration.xgenConfiguration);
34  fs.writeFileSync('xgenComponentsConfig.json', data);
```

Ukázka kódu 9 Vytvoření JSON souboru

## 9.3 Transformace z JSON na tabulku ve Wiki formátu

Po vytvoření souboru je zapotřebí provést transformaci dat z formátu JSON na formát Wiki tabulky. K tomu bylo nutno vyrobit vlastní převaděč. Převaděč je vytvořen jako konzolová aplikace psaná v jazyce C#, která jako vstupní parametr přijímá cestu k JSON souboru s konfigurací. Po spuštění dojde k načtení souboru a deserializaci JSON-u na instance tříd reprezentujících datovou strukturu daného souboru. Po tomto kroku jsou načteny lokalizace všech vlastností (konfigurační soubor obsahuje pouze kód lokalizace, ne lokalizaci samotnou).

S načtenými lokalizacemi lze již sestavit finální výčet tabulek. Za tímto účelem byly sestaveny dvě šablony. První šablona je pro tabulku jako celek a jsou v ní definovány informace o sloupcích. Druhá šablona slouží pro tvorbu řádku tabulky a specifikuje, na jakou pozici se má umístit jaká vlastnost.

```
==={0}==
<section begin="{0}" />
'''Níže uvedený popis je generován automaticky z konfigurace [[Designer formulářů|designeru formulářů]].'''
{{| class="wikitable sortable"
|-
!|Název
!|Datový typ
!|Druh
! class="unsortable" |Výchozí hodnota
! class="unsortable" |Silverlight <ref name="Silverlight">Podpora v prostředí SilverLight (Ano/Ne).</ref>
! class="unsortable" |LightWeb <ref name="LightWeb">Podpora v prostředí LightWeb (Ano/Ne).</ref>
! class="unsortable" |MultiWeb <ref name="MultiWeb">Podpora v prostředí MultiWeb (Ano/Ne).</ref>
!|Klíč poznámky
! class="unsortable" |Poznámka
{1}
|}}
<section end="{0}" />
<references />
```

Ukázka kódu 10 Šablona pro tvorbu tabulky

```
|-
|'''{0}'''
|{1}
|{2}
|{3}
|{{{{Ano/Ne - barvy| {4} }}}}
|{{{{Ano/Ne - barvy| {5} }}}}
|{{{{Ano/Ne - barvy| {6} }}}}
|{7}
|{8}
```

Ukázka kódu 11 Šablona pro tvorbu řádku tabulky

V šablonách jsou uvedeny zástupné znaky ve tvaru „{*číslo*}“. Tyto zástupné znaky jsou při procesu transformace nahrazeny finálními hodnotami jednotlivých vlastností. Všechny tabulky jsou pak sloučeny do jednoho velkého textového řetězce, který je nahrán na firemní Wiki stránku s názvem GeneratedXgenComponentsProperties. Důvod tohoto řešení bude vysvětlen v samostatné kapitole.

## 9.4 Upload na firemní Wikipedii

Upload probíhá v rámci stejného programu, který se zabývá transformací. Při komunikaci se serverem Wikipedie je napřed důležité získat CSRF token, což je bezpečnostní mechanismus zajišťující ochranu před neautorizovanými či škodlivými akcemi na webové aplikaci [16]. Po získání tohoto tokenu je možné poslat na server žádost o nahrání dat na wiki stránku, kde dotaz musí obsahovat ve svém těle kromě samotné nové definice stránky i načtený CSRF

token. Pokud jsou tyto podmínky splněny, tak server odpoví s kladnou odpovědí v podobě kódu 200 a provede požadovanou změnu. Celý tento proces lze vidět na ukázce kódu níže.

```
1   public static async Task UploadToWikiPage(string newPageContent, string wikiPageTitle)
2   {
3       using var client = new HttpClient(new HttpClientHandler {
4           UseDefaultCredentials = true,
5           PreAuthenticate = true
6       });
7
8       var csrfToken = await GetCsrfToken(client);
9       var values = new Dictionary<string, string> {
10          { "text", newPageContent }, { "token", csrfToken }
11      };
12      var postRequestContent = new FormUrlEncodedContent(values);
13      var postResponse = await client.PostAsync(
14          $"{Apiurl}?action=edit&format=json&title={wikiPageTitle}", postRequestContent
15      );
16      var response = await postResponse.Content.ReadAsStringAsync();
17
18      if (response.Contains("badtoken"))
19          throw new Exception("Wiki reported a bad token error!");
20  }
```

Ukázka kódu 12 Nahrání nové konfigurace na Wiki stránku

Seznam všech tabulek s konfigurací se zasílá na jednu konkrétní Wiki stránku z toho důvodu, že ne každá komponenta má ve firemní Wikipedii svou vlastní stránku. Dalším důvodem bylo to, že i když už Wiki stránka existuje, tak pozice požadované tabulky se vždy mírně lišila a nešlo by tak jednoduše vytvořit jednotný algoritmus pro umístění tabulky. Problém byl tedy vyřešen tím, že se tabulky s konfigurací zapisují do jedné Wiki stránky s názvem GeneratedXgenComponentsProperties a každá tato tabulka je pak dle potřeby referencována na konkrétní stránce s komponentou.

Reference se provádí vložením řetězce v požadovaném tvaru do těla cílové wiki stránky, takže například v těle stránky popisující komponentu *ListFrame* je uveden tento řádek „*{{#section:GeneratedXgenComponentsProperties|ListFrame}}*“. Výsledná podoba tabulky je pak vyobrazena na následujícím obrázku.

Vlastnosti [editovat]

**Níže uvedený popis je generován automaticky z konfigurace designeru formulářů. Neprovádějte ruční změny! Požádejte Oddělení Vývojářská platforma o změnu údajů.**

| Název | Datový typ | Druh | Výchozí hodnota | Silverlight [4] | LightWeb [5] | MultiWeb [6] | Klíč poznámky | Poznámka |
|---|---|---|---|---|---|---|---|---|
| AddAttrsListMethod | string | Behavior | | Ano | Ano | v14.03 | DEV-766982 | "Název metody, která se vyhodnotí nad Master záznamem a vrátí seznam sloupců, jenž se mají v seznamu zobrazit navíc ke standardně definovaným sloupcům. Výstupem metody má být textový řetězec, v němž je na každém řádku struktura <název atributu> |
| AfterGUICreateFill | string | Behavior | | Ano | Ano | v13.00 | DEV-766825 | Středníkem oddělený seznam dvojic <proměnná>=<hodnota>, jenž slouží pro přednaplnění atributů, aux hodnot atd. při vzniku objektu prostřednictvím tohoto formuláře. Viz AfterGUICreateFill. |
| AfterGUICreateMethod | string | Behavior | | Ano | Ano | v13.00 | DEV-766826 | Název metody, která se spustí při vytvoření nového objektu prostřednictvím daného formuláře a obvykle slouží k přednaplnění atributů. |
| AfterSaveMethod | string | Behavior | | Ano | Ano | TODO | DEV-766946 | Název metody, která se spustí po uložení záznamu prostřednictvím tohoto formuláře. |
| AggregationRow | boolean | Behavior | False | Ne | Ne | v12.01 | DEV-766923 | Příznak, který určuje, zda se má seznam zobrazit se zapnutým agregačním řádkem. |
| Align | enum | Layout | alNone | Ano | Ne | v12.01 | DEV-766813 | Určuje zarovnání komponenty vůči nadřízené komponentě. Hodnota alClient znamená roztažení na celou plochu nadřízené komponenty, hodnoty alTop, alBottom, alLeft, alRight pak přichycení k hornímu, dolnímu, levému nebo pravému okraji. |

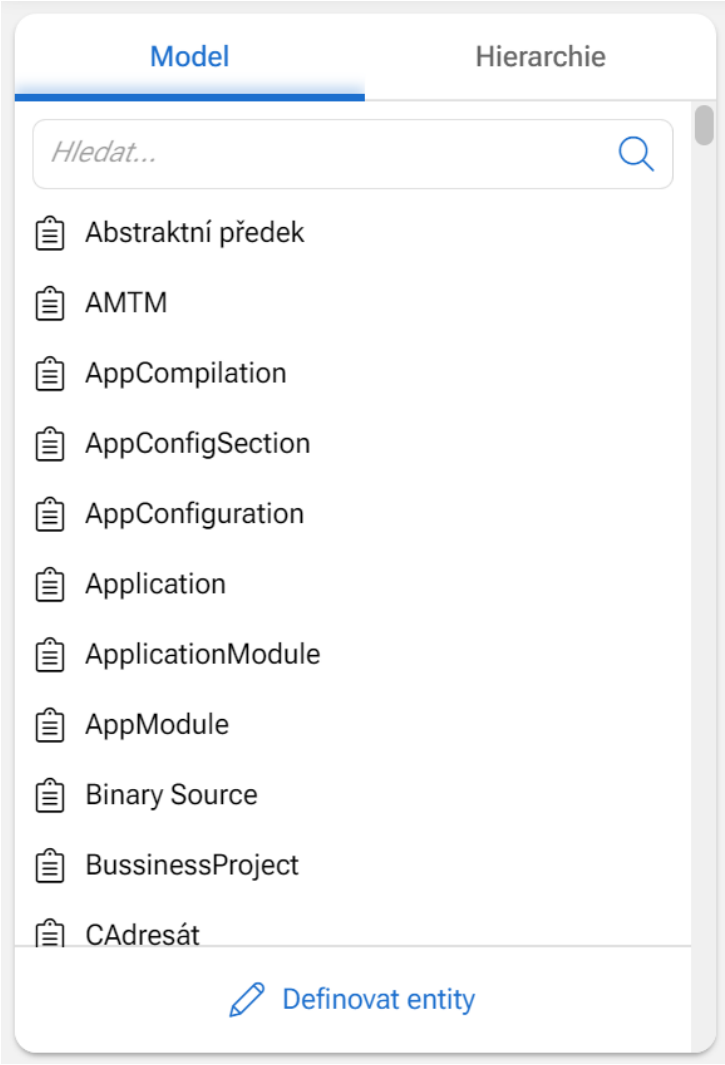
Obrázek 19 Generovaná tabulka vlastností komponenty *ListFrame* (část)

# 10 POPIS OVLÁDACÍCH PRVKŮ

V následující kapitole bude popsáno uživatelské rozhraní Designeru formulářů. Popsány budou komponenty, ze kterých se skládá a jejich možnosti používání. Dále budou rozebrány význačné komponenty panelu schématického náhledu, které reprezentují konkrétní komponenty formuláře. Uvedeny budou i možné nedostatky a případný budoucí směr.

## 10.1 Záložka Model

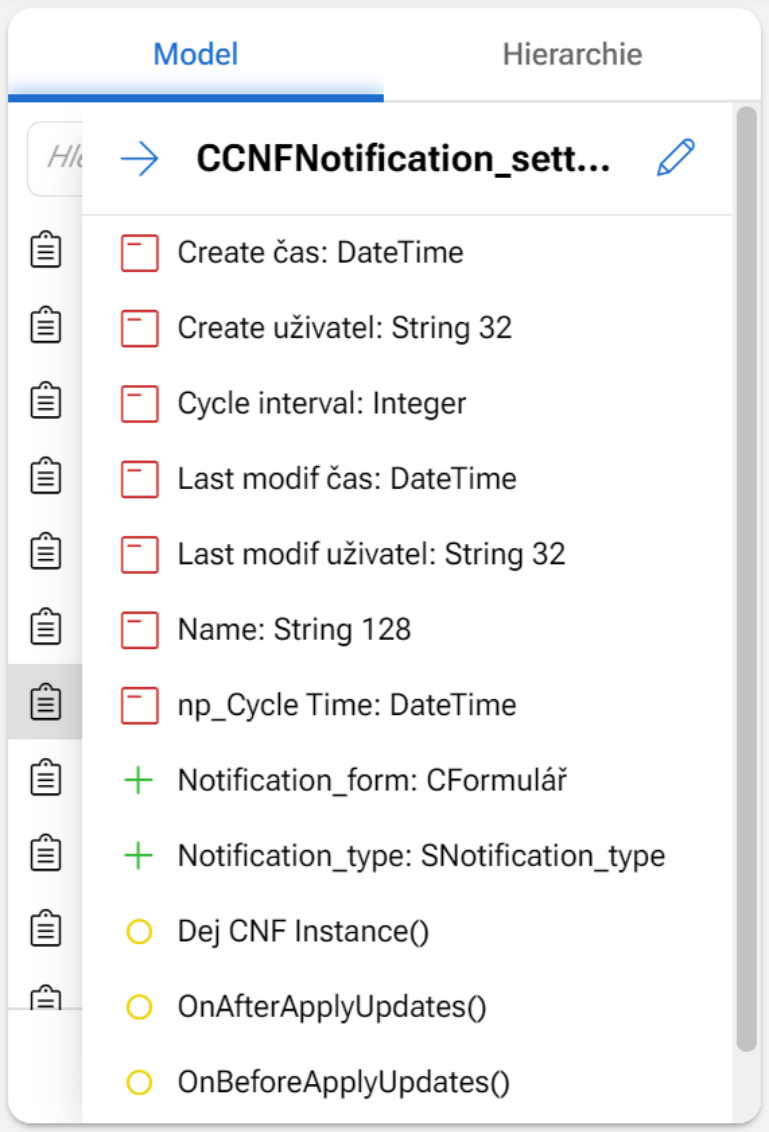
Záložka umožňuje jednodušší práci s datovým modelem aplikace, pro kterou je vytvářen formulář. Jedná se o seznam s postupným načítáním záznamů, kde každý záznam představuje třídu, která je součástí finální aplikace. Po kliknutí na libovolnou třídu dochází k zobrazení detailu o třídě a načtení všech atributů, vazeb a metod, které jsou na třídě dostupné.




Obrázek 20 Záložka Model

V rámci vazeb lze rekurzivně pouhým kliknutím zobrazovat atributy, vazby a metody třídy, jejíž instance kliknutá vazba je. Názvy těchto rozbalených vazeb se pak můžou seskupovat v horní části seznamu, pokud by mělo dojít k jejich skrytí v důsledku posunutí celého seznamu směrem nahoru.

Vyhledávání v záložce funguje pouze v rámci názvů tříd. Nelze vyhledávat v prvcích tříd.

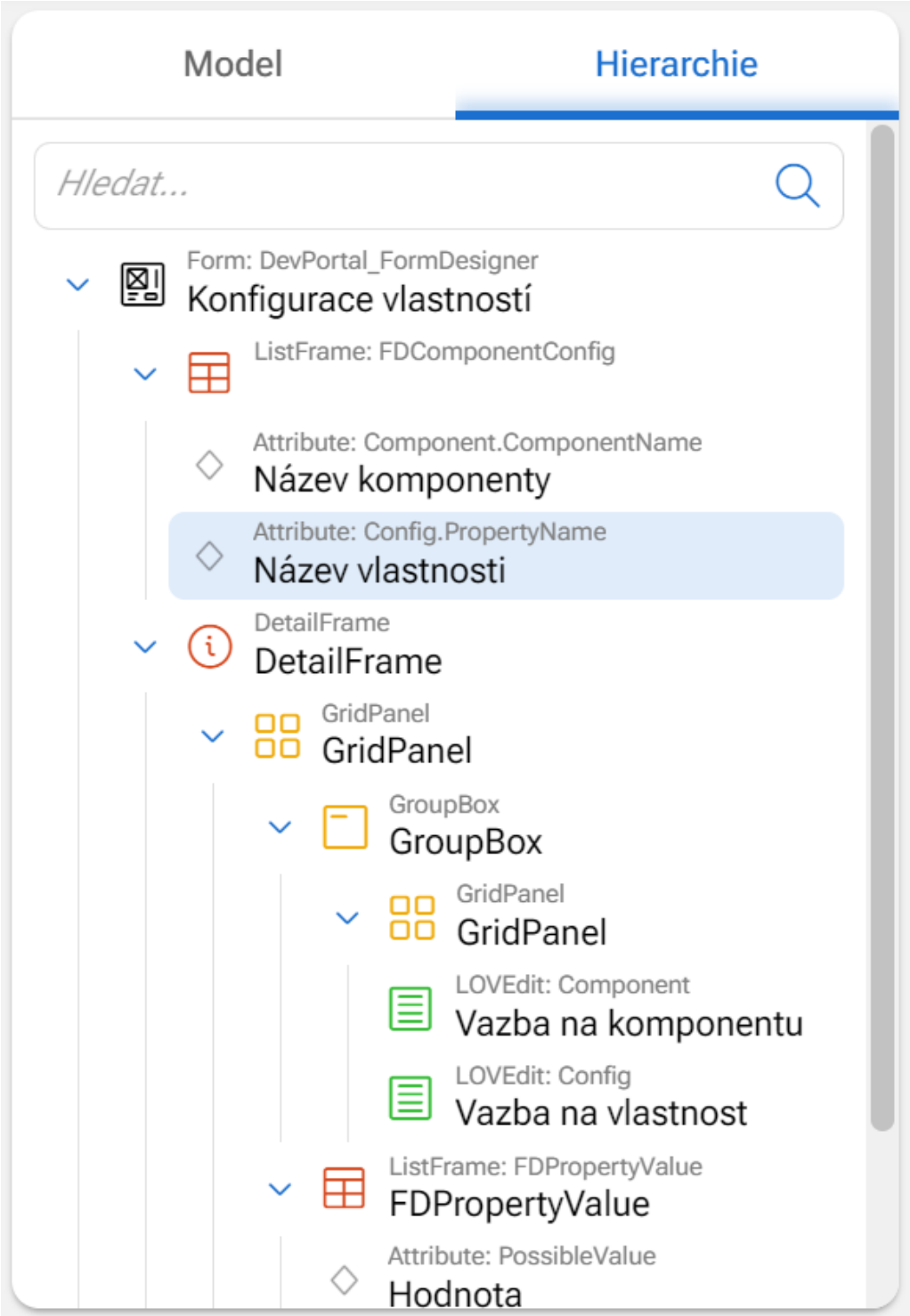



Obrázek 21 Detail označené třídy záložky Model

## 10.2 Záložka Hierarchie

Jelikož je definice formuláře hierarchickou strukturou a může se stát, že některé komponenty na sobě budou navrstveny tak, že se budou zcela překrývat, je nutné mít přehledné zobrazení ve formě stromového seznamu. Tento seznam vždy zobrazuje aspoň jeden prvek (komponentu formuláře) a umožňuje vyhledávání na základě jména komponenty. Každá

komponenta formuláře zobrazuje v hierarchickém zobrazení nejen svůj název, ale i případnou lokalizaci a hodnotu dalšího význačné vlastnosti komponenty.




Obrázek 22 Záložka Hierarchie

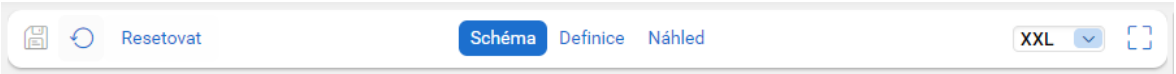
Komponenty, které mají aspoň jednoho potomka, tak mají na levé straně rozbalovací tlačítko umožňující skrytí/zobrazení potomků. Na pravé straně řádku s komponentou se po najetí myší vyskytuje tlačítko pro zneviditelnění komponenty ve schematickém náhledu. Tato funkce může být praktická pro simulaci podmíněného skrytí komponenty v reálném nasazení formuláře.

Další významnou funkcí této záložky je možnost měnit pořadí komponent pomocí „drag & drop“ operací. Stačí jednoduše uchytit řádek s požadovanou komponentou a umístit ji na požadované místo. Změna pořadí funguje pouze v rámci komponent, které mají stejného rodiče, nelze umístit komponentu pod jiného rodiče.

Poslední důležitou funkcí je zaměřování komponent pomocí dvojkliku. Tato akce označí požadovanou komponentu a lokalizuje ji ve schematickém náhledu. Lokalizace probíhá formou plynulé animace plátna schématického náhledu, které umístí komponentu na horní třetinu obrazovky.

## 10.3 Panel nástrojů

V horní části aplikace se vyskytuje panel nástrojů. Nahrazuje funkci okna navigačního menu nástroje XGEN. Tento panel se skládá ze tří hlavních sekcí, kde první sekce slouží pro manipulaci se záznamem (ukládání a případné přenačtení editovaného formuláře). Rovněž je zde tlačítko „Resetovat“, které umožňuje uvést pozici plátna schématického zobrazení do původního bodu. V druhé části je přepínač sloužící pro volbu mezi schématickým, definičním nebo náhledovým režimem. Tyto režimy budou vysvětleny v následujících kapitolách. V třetí části se pak nachází přepínač pro volbu velikosti zobrazení formuláře, kdy tato funkčnost je užitečná při práci s komponentami *GridPanel*. Uvnitř této části je i tlačítko pro zapnutí režimu přes celou obrazovku, kdy boční panely se záložkami mohou být zcela skryty.




Obrázek 23 Panel nástrojů

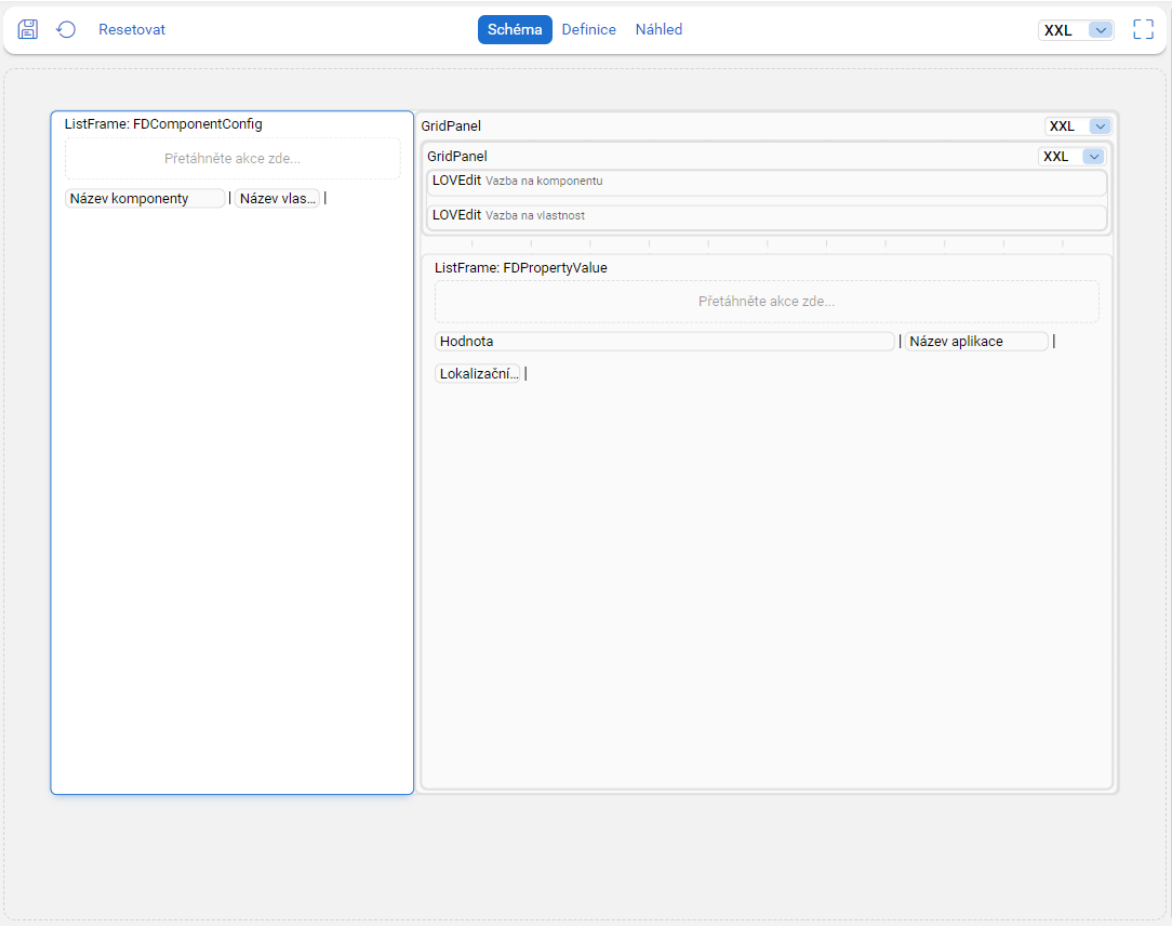
## 10.4 Panel schématického zobrazení

Jedná se o nejvýznamnější prvek Designeru formulářů. Schematické zobrazení umožňuje vizualizovat definici formuláře a přímo ji editovat. Celé zobrazení je možné pomocí pravého tlačítka myši posouvat a pomocí kolečka myši přibližovat či oddalovat. Při přibližování je

jako ohnisko přiblížení použit kurzor myši. Jinými slovy, bod, který je pod kurzorem myši zůstává fixně na svém místě v průběhu přibližování či oddalování.

Některými komponentami je jejich uchopením levým tlačítkem myši možno posouvat v rámci jejich rodičovské komponenty a rovněž je možno tyto komponenty zvětšovat či zmenšovat pomocí posuvníků umístěných na okrajích označených komponent. Podporována je i možnost hromadného posunu či změny velikosti. Jediným omezením je to, že lze označovat pouze komponenty se stejnou rodičovskou komponentou. Alternativně je pak posun řešen pomocí směrových tlačítek na klávesnici. Označené komponenty lze i kopírovat a vkládat pomocí klávesových zkratek „CTRL + C“ a „CTRL + V“. Klávesa „delete“ pak slouží pro smazání označených komponent.

V následujících podkapitolách budou uvedeny příklady hlavních komponent schématického zobrazení a popisy jejich ovládání.




Obrázek 24 Panel schématického zobrazení

### 10.4.1 ListFrame

Komponenta *ListFrame* patří mezi tzv. seznamové komponenty zobrazující data z databázové tabulky. Podporuje vkládání atributových komponent, komponent akcí a komponenty *PutFormInPanel*.

Atributy, které na výsledném formuláři reprezentují sloupce seznamu, se vkládají do prostřední části a jsou řazeny zleva doprava a od vrchu dolů. Je možné jednoduše měnit jejich šířky pomocí svislých oddělovačů mezi nimi.

Akce se umísťují do horní části komponenty a pokud mají vyplněnou lokalizaci, zobrazují i její hodnotu. Pokud dojde k situaci, kdy *ListFrame* obsahuje více akcí, než jeho šířka umožňuje zobrazit, je aktivován horizontální posuvník („scrollbar“).

U komponenty je možné pomocí dvojkliku zobrazit přidruženou třídu v záložce modelu aplikace. Toto zobrazení je možné pouze tehdy, jeli tato záložka aktivní.

### 10.4.2 Pages

Jedná se o komponentu pro zobrazování záložek, u které platí, že vždy může být zobrazen obsah právě jedné záložky. V horním panelu se zobrazují jednotlivé záložky s případnou lokalizací. Na konci tohoto seznamu se vyskytuje modré tlačítko s ikonou znaménka „plus“, které umožňuje rychle přidat novou záložku. Přidávání je ale umožněno i klasicky pomocí „drag & drop“ operací ze seznamu komponent v pravé části aplikace.

### 10.4.3 GridPanel

Komponenta slouží pro pozicování svých dílčích komponent do přehledné mřížky. Počet sloupců této mřížky je řízen vlastností „Columns“, která může být nastavena v rozmezí 0 až 12, kde 0 nula označuje stav „nevyplněno“. Tato vlastnost je dále rozpadnuta na jednotlivé velikosti XS až XXL, které reprezentují šířku komponenty v prohlížeči. Účelem tohoto rozpadu je mít možnost měnit počet sloupců při zmenšování či zvětšování okna prohlížeče. V horní pravé části komponenty se vyskytuje přepínač, který umožňuje měnit aktuální zobrazení komponenty opět v rozmezí XS až XXL.

Pro definování počtu sloupců, případně řádků, které má potomek komponent *GridPanel* zabírat, je nutné nastavit vlastnost „ColSpan“, respektive „RowSpan“

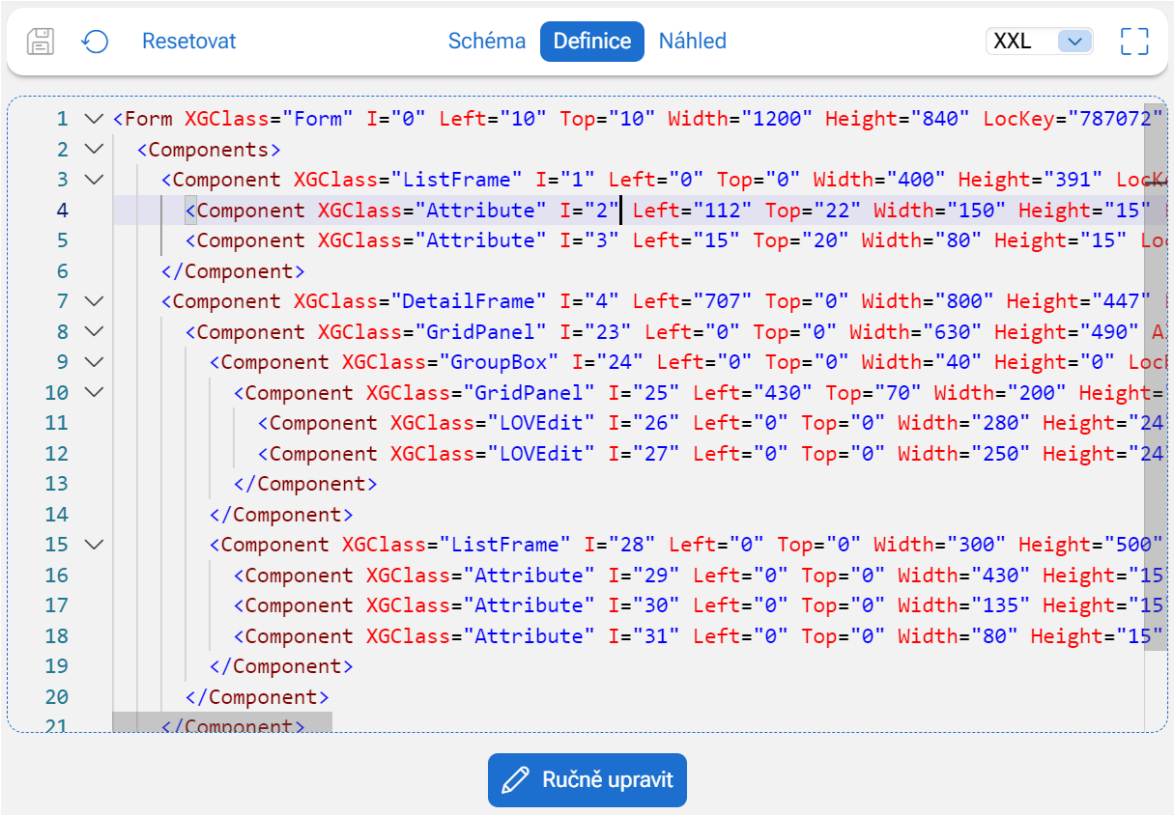
### 10.4.4 PutFormInPanel

*PutFormInPanel* umožňuje vložení jiné definice formuláře do aktuálně editovaného formuláře v podobě reference na kód formuláře. Tato funkčnost je užitečná pro znovupoužití již vytvořených definic a zkracuje tak čas potřebný pro jejich tvorbu. Zároveň redukuje množství duplicitních definic.

## 10.5 Panel definičního zobrazení

V této záložce je vyobrazena aktuální XML definice editovaného formuláře, ve které jsou provedeny všechny uživatelské změny. Uživateli je umožněno tuto definici přímo editovat. Tato vlastnost je užitečná v případech, kdy uživatel potřebuje zkopírovat již existující definici formuláře do nové definice. Po aplikování změn je definice validována na syntaktickou a sémantickou správnost.

V rámci XML náhledu je podporováno klikání na jednotlivé řádky, kdy po této akci dojde k označení asociované formulářové komponenty. Rovněž je možné označené komponenty mazat pomocí klávesy „delete“.

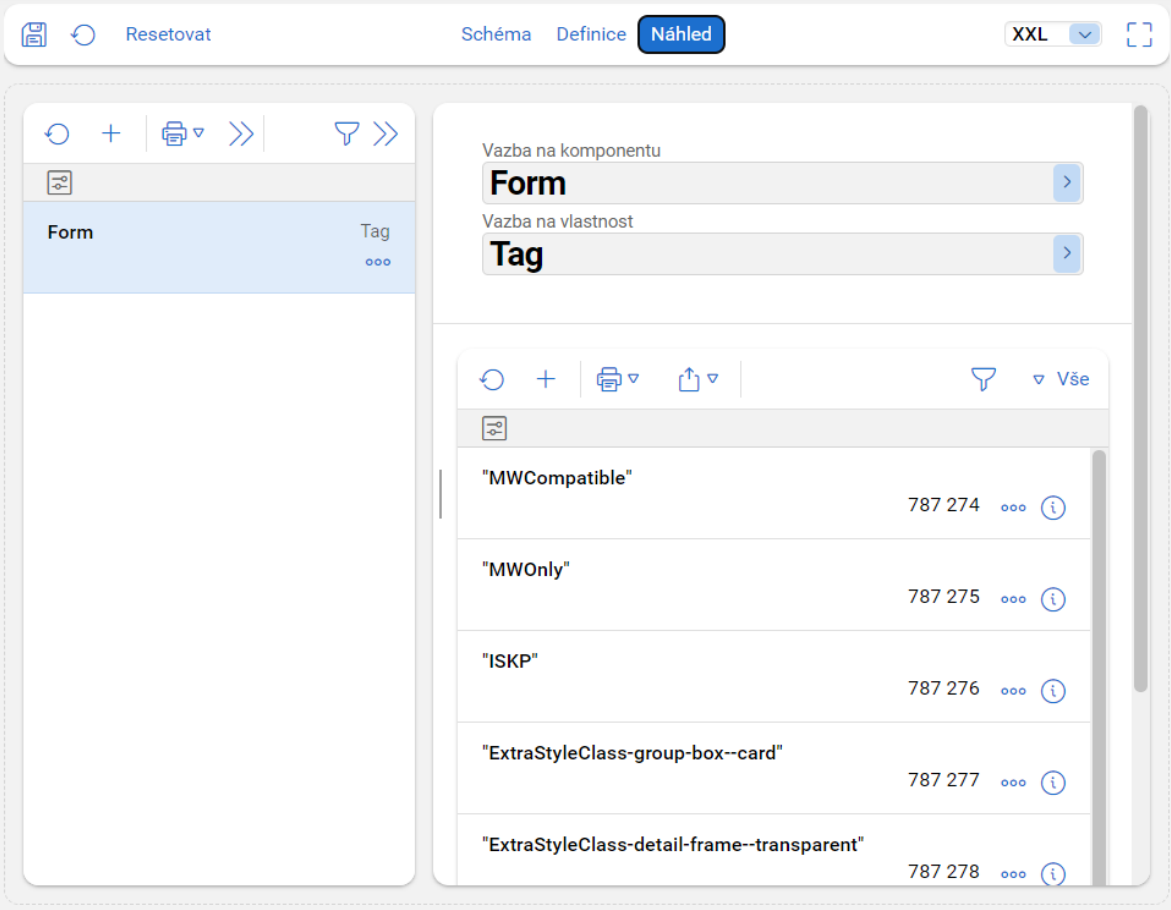



Obrázek 25 Panel definičního zobrazení

## 10.6 Panel náhledového zobrazení

Do značné míry se jedná o experimentální funkčnost, která umožňuje vykreslit formulář v reálné podobě tak, jak by vypadal v prostředí MultiWeb. Funkčnost je značně limitovaná tím, že je možné zobrazit pouze ty formuláře, jejichž asociované třídy jsou v datovém modelu aplikace, ve které běží Designer formulářů. Toto omezení je způsobené striktním napojením datového modelu na formulář v prostředí MultiWeb a zároveň jeho absencí možnosti injektáže tzv. mockovacích dat. Mockovacími daty se rozumí taková data, která strukturou odpovídají datům reálným, ale obsahují vymyšlené či náhodné hodnoty.

V budoucnu se plánuje panel náhledového zobrazení upravit tak, aby všechny tyto problémy byly eliminovány a aby se tím urychlil proces tvorby formulářů.

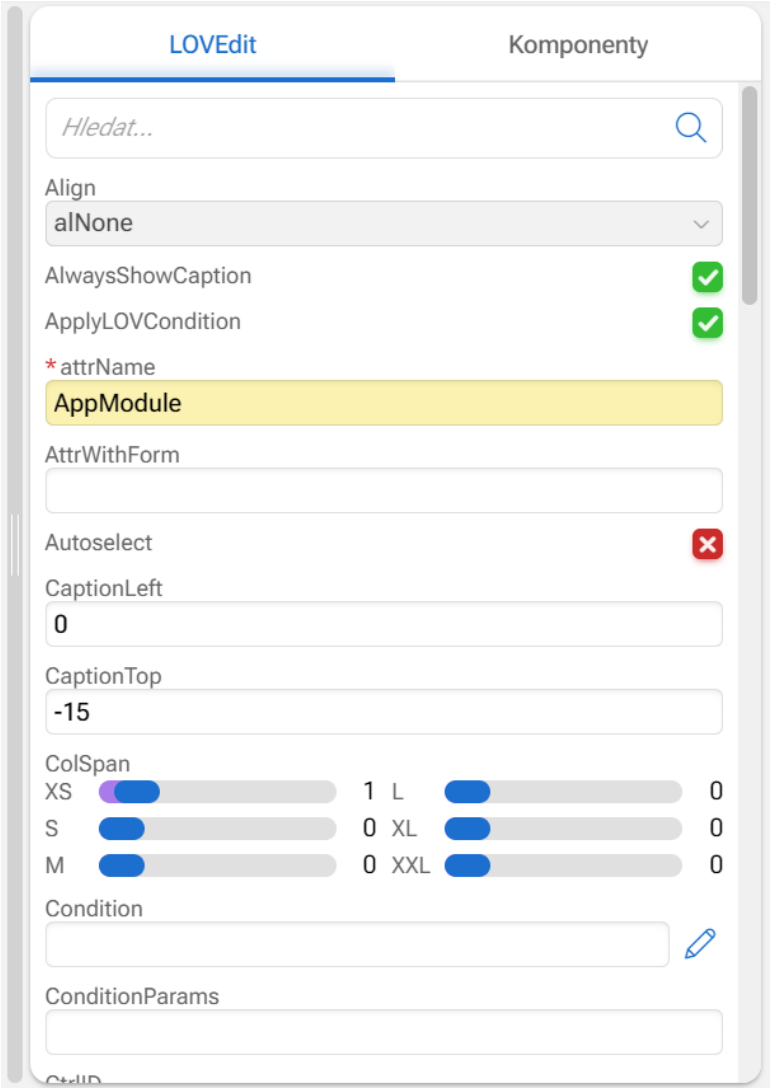



Obrázek 26 Panel náhledového zobrazení

## 10.7 Záložka vlastností

Umožňuje editaci vlastností označené komponenty. Po označení libovolné komponenty dojde na pozadí k načtení přidružených vlastností a jejich promítnutí do popisované záložky.

Každá vlastnost je specifikována svým datovým typem a tento typ určuje podobu editační komponenty na daném řádku vlastnosti. Přidanou hodnotou oproti nástroji XGEN je i okamžitá validace hodnot vlastností, kdy v případě, že se uživatel pokusí zadat nevalidní hodnotu do editačního pole (např. písmeno do pole vyžadujícího čísla), tak dojde k červenému podbarvení editačního pole a uživatel je tak vizuálně upozorněn na skutečnost, že provedl nevalidní operaci. Nad editačním polem je uveden název vlastnosti a po najetí myší se zobrazí lokalizovaná nápověda vysvětlující její účel.

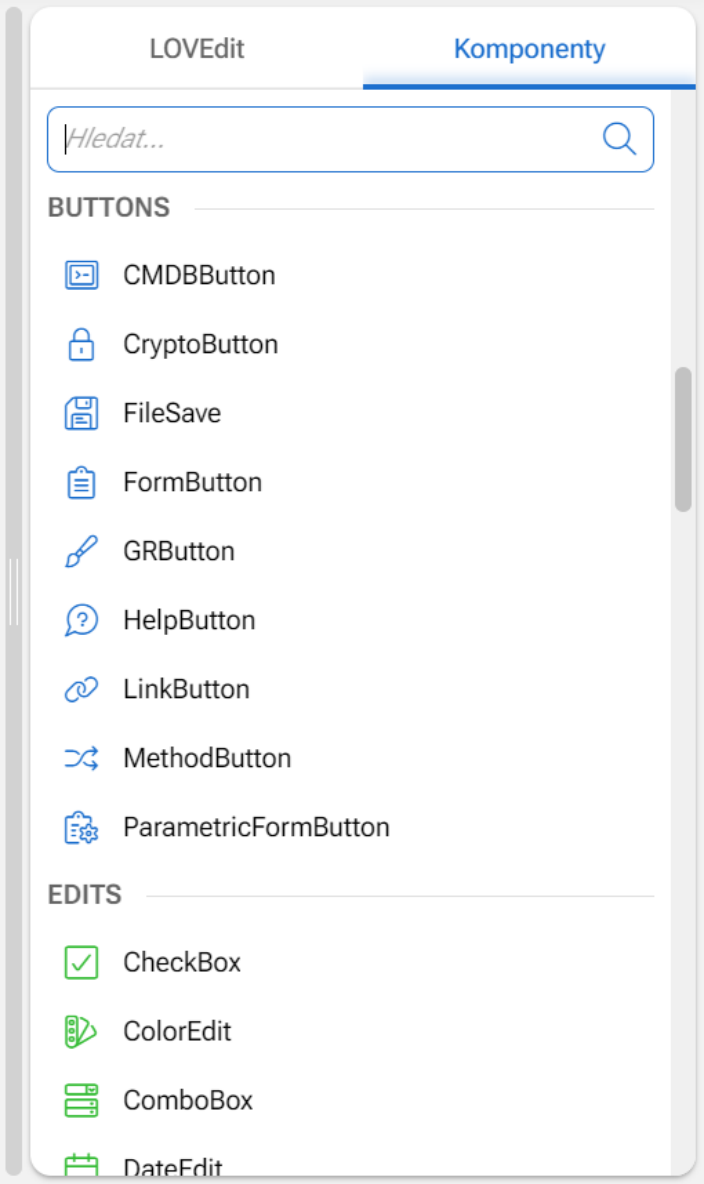



Obrázek 27 Záložka vlastností

## 10.8 Záložka Komponenty

Tato záložka obsahuje výčet všech komponent, které je možno na formulář umístit. Komponenty jsou seřazeny do logických skupin a v rámci každé skupiny je pořadí určeno abecedně. Pokud se pro nalezení komponenty využije filtrace pomocí horního

vyhledávacího pole, je seskupení podle logických skupin odebráno. Každá skupina má kromě svého názvu i přiřazenou barvu, která se projevuje u ikonky každé komponenty. Tato barva spolu s názvem je uvedena v konfiguraci komponent.

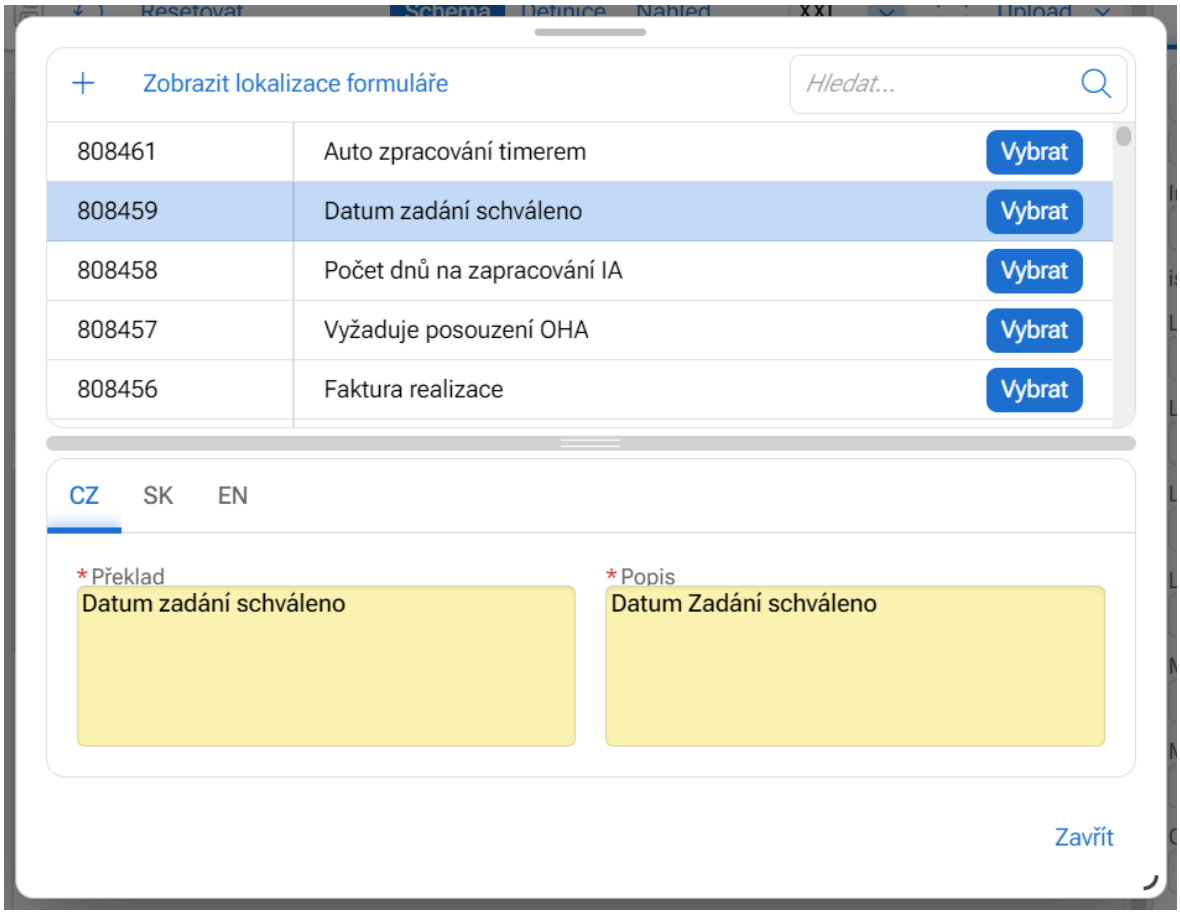



Obrázek 28 Záložka komponent

Komponenty se do formuláře umisťují pomocí „drag & drop“ operací, tedy uchycením pomocí levého tlačítka myši, následným tažením nad cílovou komponentu a puštěním levého tlačítka myši. Designer formulářů je díky tomuto přístupu schopen vizualizovat situace, kdy komponentu není možné umístit do cílové komponenty nebo naopak. Tato vizualizace je uskutečněna buď zobrazením ikony přeškrtnutého kolečka v případě, kdy komponentu nelze umístit, či zbarvením ohraničení cílové komponenty do zelena v případě, kdy lze komponentu umístit.

## 10.9 Editor lokalizací

Editor lokalizací je modálním oknem, které se zobrazí při editaci těch vlastností komponent, které jsou datového typu lokalizace. Okno funguje jako propojení s lokalizační databází společnosti TESCOSW a.s., kde se ukládají všechny formulářové lokalizace. V rámci okna je možno lokalizace vytvářet či editovat, nikoliv mazat. Množina jazyků, pro kterou lze vytvářet lokalizace je řízena konfiguračně. Pro každou aplikaci může být jiná. V rámci okna lze i vyhledávat mezi již existujícími lokalizacemi a umožnit znovupoužitelnost překladů. Vyhledávání je možné buď pomocí textu lokalizace anebo pomocí jejího kódu.

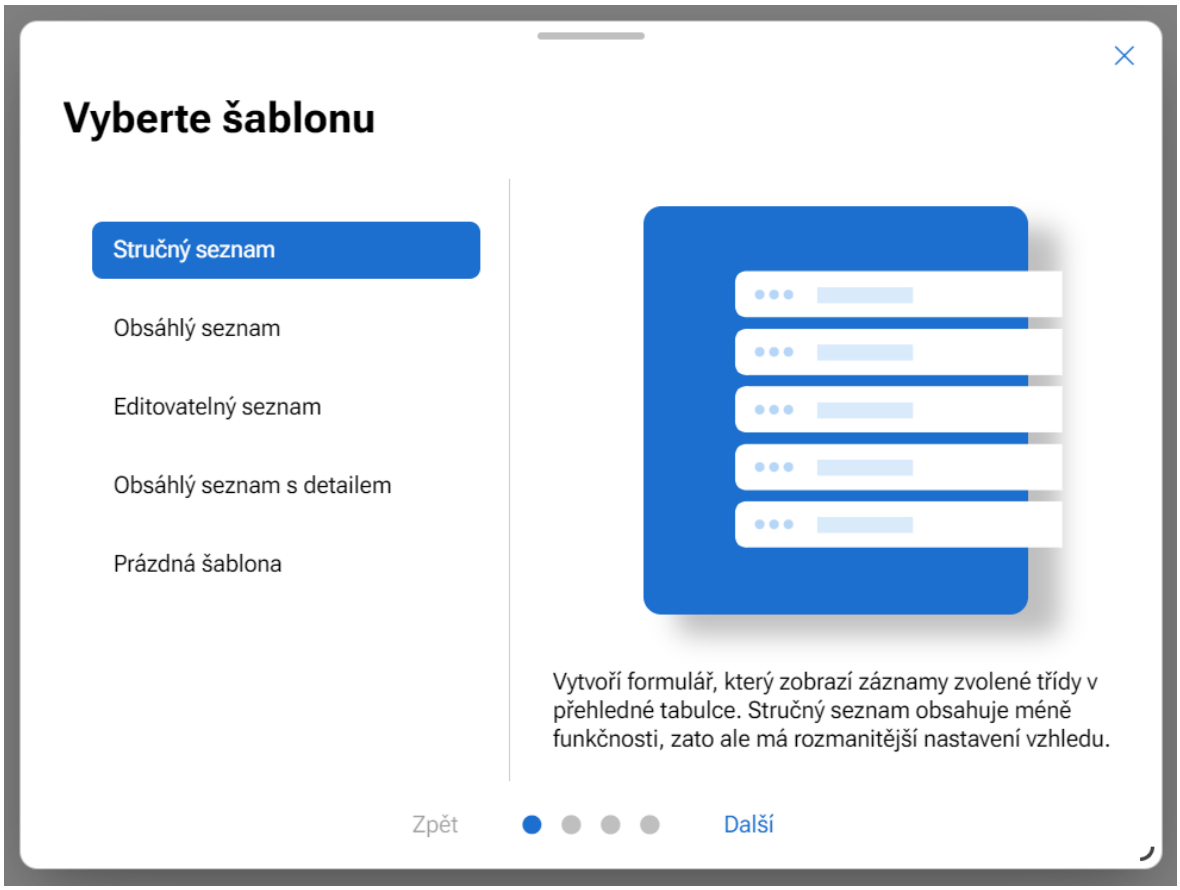



Obrázek 29 Editor lokalizací

Každá lokalizace se skládá z textu a popisu. Text označuje hlavní řetězec, který chceme přeložit (např. „Datum splatnosti“) a popis slouží pro detailnější deskripci textu (např. „Poslední den, kdy musí být zaplacena pohledávka“). Obě tyto komponenty se musí přeložit pro každý jazyk. Výsledná lokalizace je pak identifikována pomocí číselného kódu, kterému se říká lokalizační klíč. Tento klíč je následně propsán i do definice formuláře jako hodnota konkrétní vlastnosti.

## 10.10 Průvodce vytvoření formuláře

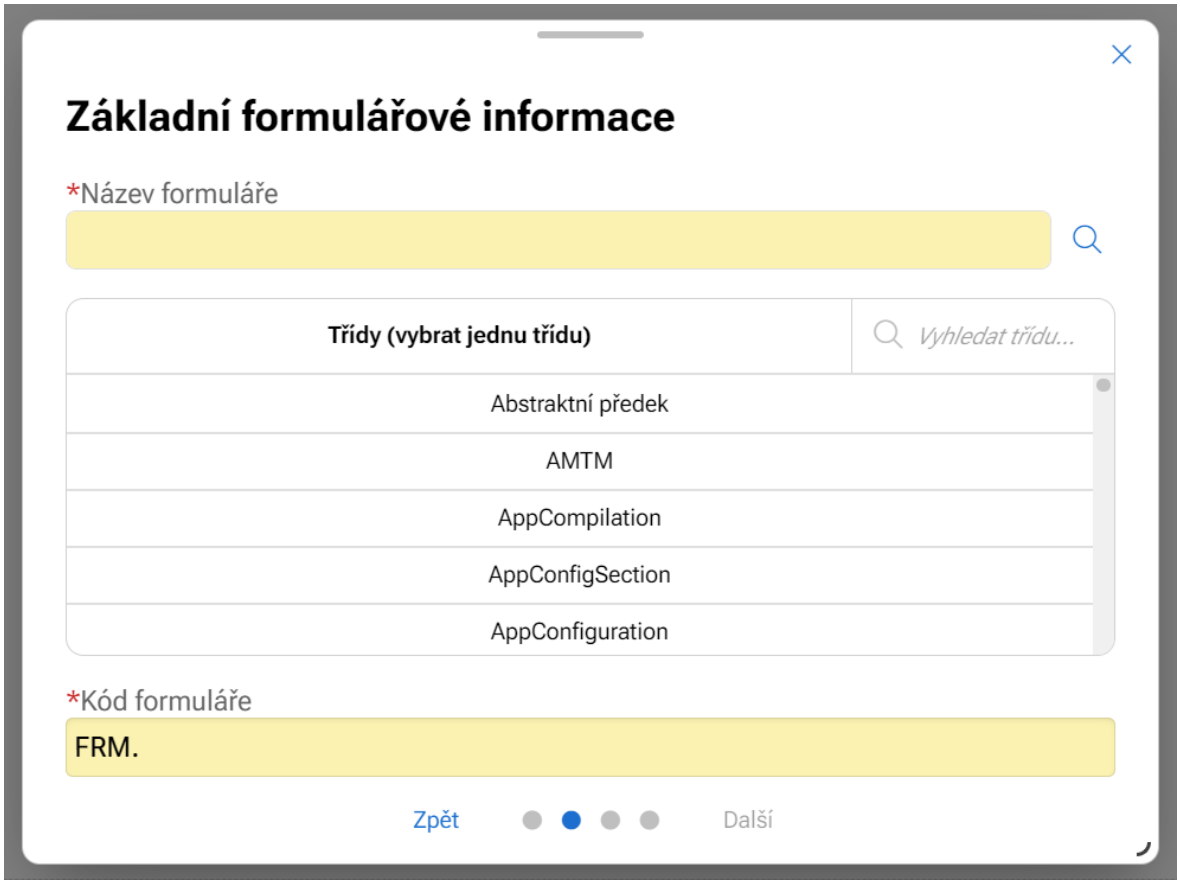
Pro nové uživatele Designeru formulářů může být první použití poněkud matoucí. Jejich znalost komponent formulářů a jejich celkového rozložení je omezená a z toho důvodu byl vytvořen určitý průvodce, který umožní jednoduše vytvořit jednu z nejvíce používaných podob formulářů, které se ve společnosti vyskytují. Průvodce se zobrazí vždy při vytváření nové instance formuláře a po jejím uložení již není možné průvodce znovu zobrazit.




Obrázek 30 Průvodce vytvořením formuláře

Volba je umožněna z prozatím pěti možností. Každá možnost je vizuálně znázorněna piktogramem, který reprezentuje přibližnou podobu formuláře v reálné aplikaci. Zároveň je pod tímto piktogramem i stručný popis charakterizující danou volbu. Poslední volba, tedy „Prázdná šablona“ je určena pro ty uživatele, kterým nevyhovuje žádná z poskytnutých šablon a kteří již mají zkušenosti s vytvářením formulářů.

Všechny šablony sdílí jeden společný krok, kde uživatel vyplní povinné atributy formuláře. Konkrétně se jedná o lokalizovaný název formuláře, jeho třídu se kterou je asociován a jeho kód. Kód musí být ve tvaru „FRM.{*Název třídy*}.{*Libovolný název*}“.




Obrázek 31 Společný krok v průvodci pro všechny šablony

Po dokončení všech kroků ve zvolené šabloně je průvodce ukončen a uživatel je ponechán se základním, ale funkčním prototypem formuláře, který je snadnější pro následnou editaci.

S celým oknem průvodce je možno posouvat pomocí horního šedého pruhu a zároveň je možné měnit jeho velikost pomocí malého šedého symbolu v pravé spodní části okna. Obsah průvodce se těmto změnám responzivně přizpůsobuje.

# 11 ZHODNOCENÍ APLIKACE

Celkově bych aplikaci ohodnotil jako úspěšnou. S mým tvrzením se shodují i názory vedení společnosti, a hlavně i pilotních uživatelů, kteří si pochvalují především urychlení práce a větší přehlednost. Jednou z nejvíce kvitovaných nových vlastností je možnost ručně upravovat definici formuláře pomocí panelu definičního zobrazení a také nový způsob zobrazování komponenty *GridPanel*, který lépe reflektuje skutečnou podobu výsledného formuláře.

Jako nedostatkem aplikace bych uvedl skutečnost, že nejsou nijak omezována práva uživatelů k formulářům, které patří do aplikací mimo jejich pole působnosti. Tento nedostatek pramení z faktu, že neexistuje žádný dokument či databáze obsahující tyto informace a její vytvoření by bylo značně náročné a vyžadující průběžnou aktualizaci. Omezení práv je nicméně jedním z budoucích plánovaných rozvojů Designeru formulářů a je v plánu čerpat informace z ještě neexistující zaměstnanecké databáze oddělení Lidských zdrojů společnosti pro asociaci množiny aplikací a jejich formulářů každému zaměstnanci.

Limitujícím faktorem je i použití samotného nástroje Select Architect pro ukládání definic formulářů, které je z hlediska doby odezvy a rychlosti poskytování informací pomalé, zejména při porovnání s dedikovanými databázovými systémy či dokonce s prostým souborovým systémem. Pro tyto důvody je v plánu v budoucnu odstoupit od využívání aplikace Select Architect a ukládat formuláře coby samostatné soubory ke zdrojovým kódům jednotlivých modelů, které podléhají verzovacímu systému a umožňují tím průběžnou revizi změn na každém z těchto formulářů.

Mezi dílčí úspěchy, kterých touto diplomovou prací bylo dosaženo, bych uvedl vytvoření TypeScript modulu zvaného Picasso umožňujícího zobrazení plátna s libovolnými prvky, které je možno přesouvat či měnit jejich velikost. Tohoto modulu je nyní využíváno i v jiných aplikacích společnosti.

Průběh tvorby Designeru formulářů byl relativně hladký a nebylo zapotřebí žádných velkých ústupků či změn směrů vývoje oproti odhadům. Práce na aplikaci budou i nadále pokračovat a s nimi i spolupráce s firmou TESCOSW a.s.

# ZÁVĚR

Cílem této diplomové práce bylo analyzovat a vytvořit náhradu dosavadního řešení pro tvorbu definic formulářů společnosti TESCOSW a.s. Rovněž bylo požadováno vytvořit ucelenou dokumentaci popisující skutečný přehled všech formulářových komponent a jejich vlastností spolu s jejich lokalizovanými popisy. Pro rekapitulaci, zde jsou zásady vypracování této diplomové práce.

- Seznamte se s technologickým frameworkem společnosti TESCOSW a.s.
- Seznamte se s aktuálním procesem tvorby definic formulářů a jejich životním cyklem.
- Analyzujte strukturu definic formulářů, zjistěte množinu možných komponent a jejich vlastností.
- Naprogramujte v požadovaných technologiích klientskou a serverovou část aplikace.
- Vyhodnoťte řešení aplikace.
- Vypracujte závěr a navrhněte budoucí vývoj aplikace.

Všech těchto cílů bylo úspěšně dosaženo a výsledkem je aplikace, kterou lze nadále jednoduše rozšiřovat o potřebnou funkčnost, která zároveň urychluje a usnadňuje dosavadní proces tvorby formulářů. Designer formulářů v plném rozsahu nahrazuje dosavadní řešení v podobě aplikace XGEN, a tak společnost TESCOSW a.s. plánuje jeho brzké ostré nasazení a současné ukončení provozu stávajícího řešení.

Vytvořená dokumentace, která je automaticky generovaná ze zdrojových kódů, poskytuje jediný zdroj pravdy a umožňuje snazší seznámení se všemi prvky formulářů. Dokumentace rovněž obsahuje informace o podpoře jednotlivých komponent, případně jejich vlastnostech, ve všech klientských prostředích, které společnost vyvíjí, a zpřehledňuje tak práci vývojářům a vývojářkám při testování nových případů užití.

V rámci diplomové práce bylo poskytnuto společnosti několik návrhů pro další optimalizaci procesu tvorby formulářů. Mezi tyto návrhy patří například množina duplicitních formulářových komponent a jejich vlastností, které by mohly být odstraněny či návrh úpravu struktury XML souborů definic formulářů, který by mohl zredukovat velikost každé definice až o 10 %.

Prostor k budoucímu rozvoji je hlavně v oblasti náhledového režimu, který by měl umožňovat real-time náhled finální podoby formuláře i bez dostupných dat se kterými má formulář pracovat. Dalším budoucím využitím bude předat tuto aplikaci i přímým zákazníkům, aby si mohli sami měnit vzhled aktuálních formulářů, či vytvářet zcela nové, bez nutnosti kontaktovat technickou podporu společnosti TESCOSW a.s.

# SEZNAM POUŽITÉ LITERATURY

[1] TESCO SW a.s.: Olomouc IČO 25892533 - Obchodní rejstřík firem. *KurzyCZ: Rejstřík firem* [online]. [cit. 2023-05-14]. Dostupné z: https://rejstrik-firem.kurzy.cz/25892533/tesco-sw-as/

[2] TESCOSW a.s.: We make IT easy. *Tescosw* [online]. Olomouc, 2023 [cit. 2023-05-14]. Dostupné z: https://www.tescosw.cz

[3] W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition). *W3C* [online]. 2008 [cit. 2023-05-14]. Dostupné z: https://www.w3.org/TR/xml/

[4] Microsoft: Silverlight End of Support. *Microsoft* [online]. [cit. 2023-05-14]. Dostupné z: https://support.microsoft.com/en-us/windows/silverlight-end-of-support-0a3be3c7-bead-e203-2dfd-74f0a64f1788

[5] Statistics & Data: The Most Popular Programming Languages – 1965/2022 – New Update. *Statistics & Data* [online]. 2022 [cit. 2023-05-14]. Dostupné z: https://statisticsanddata.org/data/the-most-popular-programming-languages-1965-2022-new-update/

[6] Free Pascal: Storage information. *Free Pascal: Storage information* [online]. [cit. 2023-05-14]. Dostupné z: https://www.freepascal.org/docs-html/ref/refsu38.html

[7] TypeScript: TypeScript is JavaScript with syntax for types. *TypeScript: TypeScript is JavaScript with syntax for types.* [online]. [cit. 2023-05-14]. Dostupné z: https://www.typescriptlang.org/

[8] React: The library for web and native user interfaces. *React: The library for web and native user interfaces* [online]. [cit. 2023-05-14]. Dostupné z: https://react.dev/

[9] Redux: A Predictable State Container for JS Apps. *Redux: A Predictable State Container for JS Apps* [online]. [cit. 2023-05-14]. Dostupné z: https://redux.js.org/

[10] Redux-Saga: An intuitive Redux side effect manager. *Redux-Saga: An intuitive Redux side effect manager.* [online]. [cit. 2023-05-14]. Dostupné z: https://redux-saga.js.org/

[11] ASP.NET: Free. Cross-platform. Open source. A framework for building web apps and services with .NET and C#. *ASP.NET: Free. Cross-platform. Open source. A framework for building web apps and services with .NET and C#.* [online]. [cit. 2023-05-14]. Dostupné z: https://dotnet.microsoft.com/en-us/apps/aspnet

[12] TypeScript: Declaration Merging. *ASP.NET: Declaration Merging* [online]. 2023 [cit. 2023-05-14]. Dostupné z: https://www.typescriptlang.org/docs/handbook/declaration-merging.html

[13] Microsoft: Automation. *Microsoft: Automation* [online]. [cit. 2023-05-14]. Dostupné z: https://learn.microsoft.com/en-us/cpp/mfc/automation?view=msvc-170

[14] Microsoft: Azure DevOps Server. *Microsoft: Azure DevOps Server* [online]. [cit. 2023-05-14]. Dostupné z: https://azure.microsoft.com/en-us/products/devops/server

[15] Microsoft: Azure Pipelines. *Microsoft: Azure Pipelines* [online]. [cit. 2023-05-14]. Dostupné z: https://azure.microsoft.com/en-us/products/devops/pipelines/

[16] MediaWiki: Manual:Edit token. *Microsoft: Azure Pipelines* [online]. 2010 [cit. 2023-05-14]. Dostupné z: https://www.mediawiki.org/wiki/Manual:Edit_token

[17] BANKS, Alex a Eve PORCELLO. Learning React: functional web development with React and Redux. Beijing: O'Reilly Media, 2017. ISBN 978-1-491-95462-1.

[18] TROELSEN, Andrew W. a Philip JAPIKSE. Pro C# 7: with .net and .net core. Eighth edition. New York, NY: Apress, [2017]. ISBN 978-1-4842-3017-6.

[19] HOTEK, Mike. Microsoft SQL Server 2008: krok za krokem [online]. Brno: Computer Press, 2009 [cit. 2022-10-21]. ISBN 978-80-251-2466-6.

[20] FENTON, Steve. Pro TypeScript: application-scale JavaScript development. [New York]: Appress, [2014]. The expert's voice in TypeScript. ISBN 978-1-4302-6791-1.

[21] LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.

# SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

| | |
|---|---|
| API | Application Programming Interface |
| BPMN | Business Process Model and Notation |
| CASE | Computer-Aided Software Engineering |
| CSRF | Cross-site Request Forgery |
| LW | LightWeb |
| MDD | Model Driven Development |
| MW | MultiWeb |
| SEO | Search Engine Optimization |
| SL | Silverlight |
| SOAP | Simple Object Access Protocol |
| SPA | Single Page Application |
| TEAF | TESCOSW Enterprise Application Framework |
| UML | Unified Modeling Language |
| WYSIWYG | What you se eis what you get |
| XML | Extensible Markup Language |
| XMI | XML Metadata Interchange |

# SEZNAM OBRÁZKŮ

Obrázek 1 Sídlo společnosti TESCOSW a.s. ..... 11
Obrázek 2 Příklady podoby deklarace XML ..... 14
Obrázek 3 Druhy XML značek ..... 14
Obrázek 4 Příklad použití XML atributů ..... 15
Obrázek 5 Příklad možné podoby definice formuláře ..... 15
Obrázek 6 Návrh optimalizace zápisu definice formuláře ..... 16
Obrázek 7 Podoba formuláře *Funkce* v prostředí Silverlight ..... 17
Obrázek 8 Podoba formuláře *Funkce* v prostředí MultiWeb ..... 17
Obrázek 9 Podoba aplikace XGEN ..... 18
Obrázek 10 XGEN – schématický náhled ..... 19
Obrázek 11 XGEN – okno nastavení vlastností ..... 20
Obrázek 12 XGEN – okno s datovým modelem ..... 21
Obrázek 13 XGEN – navigační menu ..... 22
Obrázek 14 Ukázka nástroje Select Architect ..... 23
Obrázek 15 Značka GUICfg u označené třídy ..... 24
Obrázek 16 Podoba aplikace Portál vývojáře ..... 27
Obrázek 17 Definice formuláře Designeru formulářů ..... 28
Obrázek 18 UML diagram projektu DevPortal_FormDesigner ..... 42
Obrázek 19 Generovaná tabulka vlastností komponenty *ListFrame* (část) ..... 48
Obrázek 20 Záložka Model ..... 49
Obrázek 21 Detail označené třídy záložky Model ..... 50
Obrázek 22 Záložka Hierarchie ..... 51
Obrázek 23 Panel nástrojů ..... 52
Obrázek 24 Panel schématického zobrazení ..... 53
Obrázek 25 Panel definičního zobrazení ..... 55
Obrázek 26 Panel náhledového zobrazení ..... 56
Obrázek 27 Záložka vlastností ..... 57
Obrázek 28 Záložka komponent ..... 58
Obrázek 29 Editor lokalizací ..... 59
Obrázek 30 Průvodce vytvořením formuláře ..... 60
Obrázek 31 Společný krok v průvodci pro všechny šablony ..... 61

# SEZNAM UKÁZEK KÓDŮ

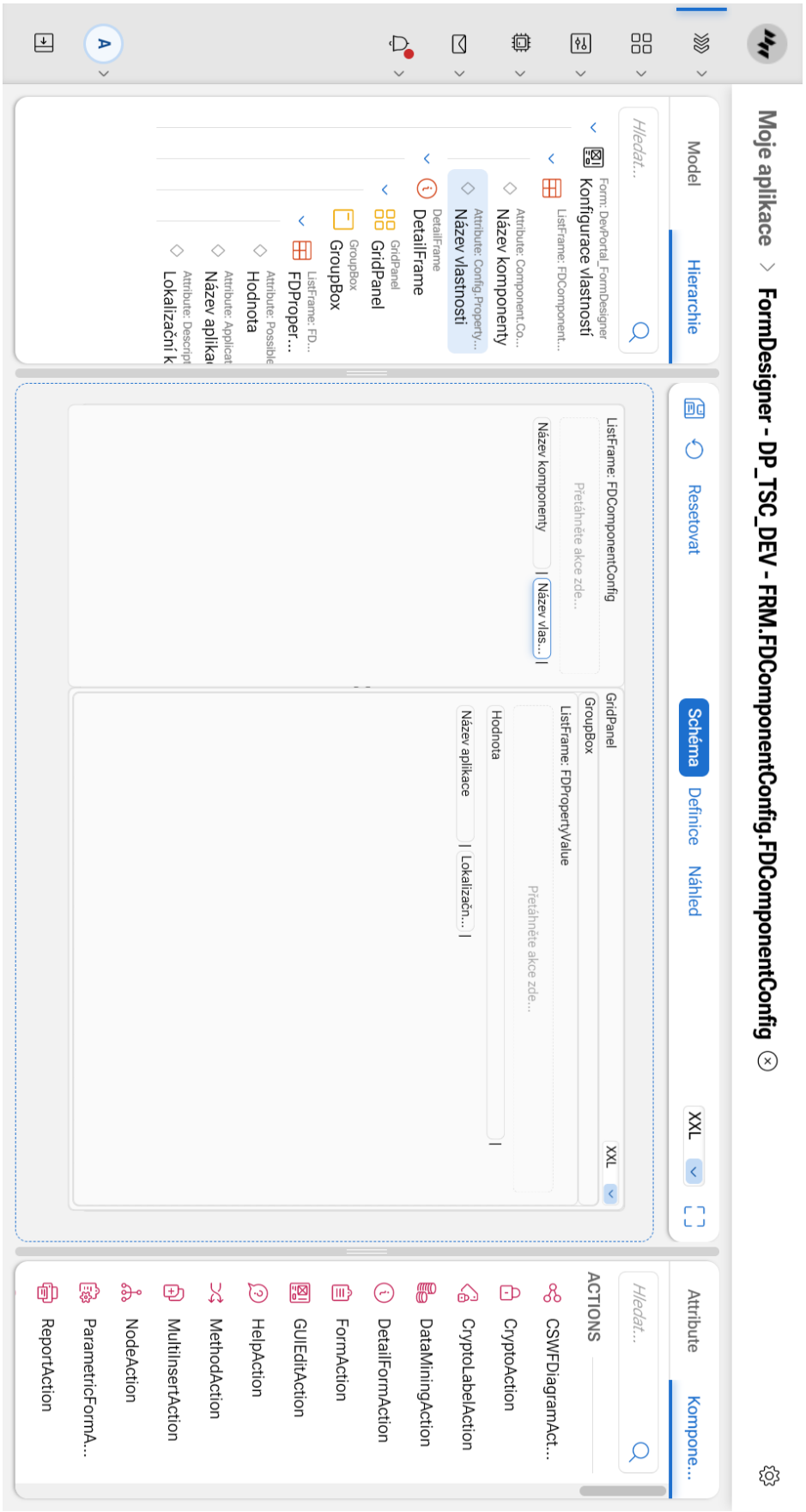
Ukázka kódu 1 Registrace Designeru formulářů do MultiWeb klienta....34
Ukázka kódu 2 Typ popisující datový stav aplikace (Redux store) ....36
Ukázka kódu 3 Ukázka definice jedné z akcí....37
Ukázka kódu 4 Příklad použití *instanceof* v reduceru ....38
Ukázka kódu 5 Příklad ságy pro uložení definice formuláře....39
Ukázka kódu 6 Příklad definice konfigurace vlastností ....40
Ukázka kódu 7 Příklad definice konfigurace komponent....41
Ukázka kódu 8 Vytvoření komunikačního objektu se Select Architect ....43
Ukázka kódu 9 Vytvoření JSON souboru....45
Ukázka kódu 10 Šablona pro tvorbu tabulky....46
Ukázka kódu 11 Šablona pro tvorbu řádku tabulky ....46
Ukázka kódu 12 Nahrání nové konfigurace na Wiki stránku....47

# SEZNAM PŘÍLOH

Příloha P I: Celkový pohled na Designer formulářů

Příloha P II: CD se zdrojovými kódy

# PŘÍLOHA P I: CELKOVÝ POHLED NA DESIGNER FORMULÁŘŮ



# PŘÍLOHA P II: CD SE ZDROJOVÝMI KÓDY

CD obsahuje soubory klientské i serverové části Designeru formulářů. Jejich spuštění je možné pouze v rámci aplikace Portálu vývojáře uvnitř interní sítě společnosti TESCOSW a.s. Pro prohlížení je doporučeno použít program Visual Studio Code.