VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro práci s Office Open XML v OS Android

Application for Office Open XML in OS Android

2012 **Bc. Jaroslav Vrba**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jaroslav Vrba**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace pro práci s Office Open Xml v OS Android
Application for Office Open XML in OS Android

Zásady pro vypracování:

Jelikož operační systém Google Android je poměrně nový, neexistuje pro něj takové množství aplikací jako třeba pro Microsoft Windows Mobile. Jednou z nepostradatelných aplikací každého systému by měl být balík pro práci s dokumenty. Těch však v androidu není dostatek a většinou umí dokument pouze otevřít. Pokud uživatel potřebuje vytvářet dokumenty, musí sáhnout po placené verzi, kterých i tak není dostatek. Hlavním cílem práce je vytvořit je srovnat již existující balíky. Srovnat jejich výhody a nevýhody, porozumět práci s otevřenými formáty dokumentů OpenXML. Na základě těchto poznatků pak navrhnout a implementovat vlastní aplikaci, která by umožňovala nejen čtení ale i vytváření a editaci dokumentů OpenXML.
Práce bude obsahovat:
1. Porovnání již existujících balíků. Zjištění jejich výhod a nedostatků.
2. Analýza OpenXml.
3. Návrh implementace vlastního kancelářského balíku.
4. Implementace kancelářského balíku - prvního prototypu se základní funkčností čtení a zápisu "Word" a "Excel" dokumentů.
5. Zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Svatopluk Štolfa, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 2.5.2012 .....

Poděkování

Rád bych poděkoval Ing. Svatopluku Štolfovi Ph.D. za odbornou pomoc a konzultace při vytváření této diplomové práce.

Dále bych chtěl poděkovat rodičům za jejich dlouhodobou podporu, a to jak morální, tak finanční.

Abstrakt

Diplomová práce se zabývá návrhem a tvorbou aplikace, sloužící k vytváření, čtení a editaci dokumentů Open Office XML, konkrétně formátu *Docx* a *Xlsx*. Aplikace je vyvinuta pro operační systém Android. Účelem bylo vyvinout aplikaci s volně přístupnými zdrojovými kódy. Práce se skládá z teoretické a praktické části. V teoretické části je popsán standard OpenXML a programy, vytvořené pro práci s tímto standardem. Praktická část je realizace programu, který je navrhnut a detailně popsán v teoretické části.

Klíčová slova

OOXML, DOCX, XLSX, ANDROID, ELEMENT

Abstract

The diploma thesis deals with a proposal and creation of application which is used to create, read and edit documents Open Office XML, specifically DOCX and XLSX format. The application is developed for the Android operating system. The purpose of this thesis was to develop an application with open source code . The work is consisted of theoretical and practical part. The theoretical part describes the OpenXML standard and programs designed to work with this standard. The practical part is the realization of application which is designed and described in detail in the theoretical part.

Key words

OOXML, DOCX, XLSX, ANDROID, ELEMENT

Seznam použitých symbolů a zkratek

Zkratka	Anglický význam	Český význam
OOXML	Open Office XML	Otevřené kancelářské XML
XML	Extensible Markup Language	Rozšířený značkovací jazyk

Obsah

1 Úvod

Operační systém Google Android je poměrně nový. Pro tento operační systém neexistuje mnoho programů, které by sloužily ke čtení a editaci dokumentů standardu OpenXML, konkrétně *Docx* a *Xlsx*. Pokud však existují, jsou velmi drahé. Z tohoto důvodu vzniká tato nová aplikace, která je open-source, což znamená, že zdrojové kódy jsou volně k dispozici.

V teoretické části práce je porovnávána vytvořená aplikace s existujícími aplikacemi, které se již na trhu vyskytují. U každé aplikace jsou uvedeny její výhody či nevýhody vůči vytvořené aplikaci. V další kapitole je podrobně popsán standard OpenXML s částmi WordProcessingML a SpreadsheetML, kde část WordProcessingML popisuje práci s dokumenty formátu *Docx* a SpreadsheetML popisuje práci s dokumenty standardu *Xlsx*. V kapitole *Návrh implementace vlastního kancelářského balíku* je popsána struktura aplikace a její třídy, které využívá i s proměnnými, sloužících k uložení dat.

Praktickou část představuje zmíněná aplikace pro čtení a editaci souborů OpenXML. Program je rozdělen na dvě části. První část se nazývá *Word,* slouží k vytvoření nového dokumentu *Docx* či k editaci dokumentů a uložení provedených změn. Druhá část se nazývá *Excel*, slouží k vytvoření dokumentu formátu *Xlsx* či k editaci libovolného dokumentu tohoto formátu.

2 Existující programy

Programy pro práci se soubory formátu Open Office XML pro operační systém Android již existují. Většina těchto programů v základní bezplatné verzi nabízí pouze mód prohlížení dokumentů, v rozšířené zpoplatněné verzi nabízejí funkce pro editaci dokumentu, jako je například změna fontu písma. V této práci budou popsány 2 nejčastěji používané programy pro práci s dokumenty.

2.1 Documents To Go

Documents To Go je program sloužící k práci s dokumenty formátu OOXML v mobilních telefonech. Vyvinula jej firma DataViz. Program umožňuje čtení, editování a vytváření nových dokumentů OOXML formátu včetně novější verze 2007. [3].
Aplikace je rozdělena na 4 částí, kde první tři části pracují s Microsoft Office formátem. Jsou to následující části:

- Word To Go
- Sheet To Go – slouží ke zpracování Excelovských dokumentů
- Slideshow To Go – slouží ke zpracování prezentací Power Point
- PDF To Go – slouží k čtení PDF souborů.

2.1.1 Word To Go

Slouží k zpracování Word dokumentů v mobilních telefonech. Po spuštění aplikace je možné vytvořit nový dokument, editovat nebo načíst existující dokument. Při otevření dokumentu se do aplikace postupně načte veškerý jeho obsah a jednotlivé strany se očíslují. Načtený obsah si lze číst (je možné obsah zoomovat), nebo jej lze editovat. [2]

Při editování lze využít tyto operace:

- Vkládání, kopírování a vyjmutí vybraného textu
- Změnit font vybraného textu
- Změnit font písma
- Barvu písma
- Horní index, dolní index, všechna písmena velká či malá

Při editaci či vytvoření nového dokumentu lze dokumentu přidat seznam prvků, který může být buďto číslovaný, nebo odrážkový. Program disponuje průvodcem pro tvorbu tabulek, jehož

výstupem jsou tabulky s daty. Z libovolné části textu lze vytvořit záložku nebo hyperlink, odkazující na část textu v dokumentu či na jinou webovou stránku. Poslední možností úpravy je přidání komentáře nad libovolným slovem dokumentu.

2.1.2 Sheet To Go

Program umožňuje číst, vytvářet nebo editovat existující soubory ve formátech *Xls* či novějších *Xlsx*. Při editaci nebo při tvorbě nového dokumentu lze s textem provádět následující operace: vkládání, kopírování a vyjmutí vybraného textu. Operace s buňky jsou následující: Mazání, vkládání, skrytí obsahu, změna fontu, zarovnání buněk, sloučení buněk, změna velikosti buňky a seřazení obsahu daných buněk.

Aplikace podporuje výpočetní funkce. Pomocí nich lze provádět výpočty nad vybraným obsahem. Při přidávání funkce do aplikace se vybere buňka, do které se umístí její výsledek a dále se funkci předají parametry, uložené v libovolných buňkách a typ funkce.

2.1.3 Slideshow To Go

Slouží k vytváření, úpravě a prohlížení prezentací formátu *ppt* či *pptx* standardu OOXML. Prezentaci lze prohlížet postupně slide po slidu nebo využít navigátoru zobrazující zadanou stránku. Obsah slidu lze editovat. Po kliknutí na tlačítko editace slidu se objeví upravitelný text, který se může mazat či rozšiřovat. V módu editace slidů lze provádět následující operace s prezentací: přidání či smazání slidu, úprava odrážek ve slidu, to je přidání nové odrážky či změna levelu odrážky, přidání či editace poznámky ke slidu, zduplikovat jednotlivé slidy a následně je seřadit. Po skončení módu editace se změny uloží a vrací se zpět mód prohlížení slidů.

2.1.4 Srovnání programu Documents To Go a vytvořenou aplikací

Program Documents To Go podporuje na rozdíl od vytvořené aplikace navíc ještě čtení PDF souborů a vytváření prezentací.

Výhody programu:

- Podporuje více úprav stylů v dokumentu
- Vyhledávání v textu a nahrazování textu
- Zoomovaní obsahu v programu
- Sdílení uložených souborů na sociální sítě či e-mail.
- V Sheet To Go podpora funkcí pro buňky

- Vytváření tabulek a grafů z dat využitých v souboru
- Podporuje čtení dokumentů ve formátu PDF

Největší slabinou programu je jeho cena. Oproti vytvořené aplikaci v diplomové práci však podporuje i starší formáty jako *doc*, *xls*, *ppt*. Další nevýhodou je paměťová velikost programu, která je citelně znát například v mobilních telefonech s malou kapacitou paměti. K nevýhodám lze taktéž přidat procesorovou náročnost na někdy nepříliš výkonných procesorech, převážně u starších mobilních telefonů, na kterých může program běžet velmi pomalu.

2.2 Quickoffice for Android

Pomocí programu lze prohlížet, editovat, vytvářet a sdílet dokumenty formátu OOXML. Program má integrovanou podporu pro sdílení dokumentů do cloudových služeb a to do MobileMe, Dropbox, Google Docx a Box.net.

Dokument lze z uvedených cloudových uložišť stáhnout do zařízení, kde jej lze následně prohlížet či editovat. Editovaný dokument lze updatovat zpět do cloudu. Dále je možné mazat či přejménovávat soubory. Program umožňuje otvírat soubory umístěné přímo v zazipovaném souboru bez případného rozzipovaní. [4]

2.2.1 Quickword

Program je určen k prohlížení a editaci Word souborů formátu *Doc* a *Docx*. Dokáže vytvořit nový soubor, kde uživatel vybírá, v jakém formátu se dokument uloží. Aplikace umožňuje vytvořené či editované soubory ukládat i do jiných formátu (PDF, HTML, RTF). Podporuje předvytvoření nových šablon, které se dají využít při vytváření nového dokumentu.

Dále program umožňuje načtení dokumentu, na kterém lze provést následující úpravy: vkládání, kopírování a vyjmutí vybraného textu; změnu fontu písma a jeho barvu; nahrazování vyhledaného textu; vložení obrázků a změnu jejich velikosti, případně otáčení; zarovnání odstavce vlevo /vpravo / střed a zvýraznění textu

2.2.2 Quicksheet

Program slouží k otevření, editaci a sdílení souboru formátu *Xls* a *Xlsx*. Při vytvoření nového či editaci stávajícího souboru umožňuje provádět tyto operace: editace textu; rozdělení či sloučení buněk; vyjmutí, kopírování či vložení buněk; smazání či přidání nového řádku nebo sloupce;

formátování barev a stylů obsahu vybraných buněk k editaci; vyhledávání textu v celém listu či i ve více listech dané aplikace a základní editaci funkcí, které list obsahuje (140 funkcí)

2.2.3 Srovnání Quickoffice forAndroid s vytvořenou aplikací

Program Quickoffice for Android je stejně jako aplikace vytvářená v rámci diplomové práce rozdělen na části, podle toho, s jakými formáty OOXML program aktuálně pracuje.
Výhody programu vůči vytvořené aplikaci:

- U souboru formátu *Docx* a *Doc* umožňuje vyhledávání v textu
- Program podporuje otevírání PDF dokumentů
- Umožňuje sdílení vytvořených dokumentů pomocí cloud služeb
- Vytvářet seznamy v dokumentech
- Vkládání obrázků a vytváření grafů z dat
- Procházení a otvírání souborů z archivu ZIP
- Při práci s dokumenty formátu Xlsx a Xls podpora matematických rovnic

Srovnání ukazuje, že existující program Quickoffice for Android nabízí mnohem více funkcí oproti vytvořené aplikaci. Je to způsobeno tím, že program vyvíjí desítky programátorů. Aplikace v rámci této diplomové práce neobsahuje tolik funkcí. Její největší výhoda oproti popsanému programu je ta, že je zdarma a nezabírá v paměti zařízení mnoho místa. Dalším výhodou vytvořené aplikace vůči programu je ta, že není tak procesorově náročná.

3 Analýza OpenXML

Standard vznikl roku 2007. Vyvinula jej firma Microsoft. V současné době je tento standard již vlastněný společností ECMA a prosazuje se jako mezinárodní norma ISO.
OOXML je soubor specifikací definující, jakým způsobem se uloží dokumenty, tabulky a prezentace. Jsou to následující specifikace:

- WordprocessingML – XML soubory pro textové informace
- SpreadsheetML – XML soubory k uložení tabulkových informací
- PresentationML - XML pro prezentaci souborů
- DrawingML – XML pro grafické elementy dokumentů
- Custom XML – definice uživatelských XML a následné propojení s obsahem dokumentu [5]

Formát Office Open XML je ZIP soubor, který v sobě obsahuje XML a další potřebné soubory. Výsledkem jsou tak menší soubory než ty binární, které byly vytvářeny předchozími verzemi Microsoft Office. Podle vyjádření společnosti Microsoft je hlavním cílem formátu zpětná kompatibilita s existujícími dokumenty a plná podpora jejich rozšiřujících vlastností. [5].
V případě poškození některého souboru XML v ZIP archivu je možné aplikaci spustit, ale pouze tehdy, pokud není pro běh nezbytně nutný. Například obrázek v textu není pro běh aplikace vyžadován.
Aplikace diplomové práce pracuje s textovými a tabulkovými daty, proto v následujících dvou kapitolách budou podrobně probrány specifikace, které jsou potřebné k jejich činnosti.

3.1 WordprocessingML

Je založen na formátu body – paragraf (odstavec) – run. Popisuje XML soubory pro textové informace.

3.1.1 Struktura Open Office XML aplikace

Soubor je uložen v ZIP archivu. Po dekomprimaci obsahu Open Office XML souboru se zobrazí struktura složek

Obrázek 1

Kde složky _rels, docProps a soubor [Content_Types].xml jsou povinné pro každý dokument formátu OOXML. Složka word napovídá, že se jedná o aplikace Word. V aplikacích Excel se místo složky word objevuje složka xl.

3.1.1.1 Dokument [Content_Types].xml

Soubor se vyskytuje vždy v kořenu aplikace. Tento soubor je povinný. Jsou v něm definovány všechny datové typy nacházející se v aplikaci. Pokud aplikace Word obsahuje soubor MP3 a dále obrázek formátu JPEG, musí být tyto informace obsažené právě v tomto dokumentu. Tyto informace se v souboru značí pomocí tagu *Default* a atribut *Extension* označuje typ, který je v archivu obsažen. Část dokumentu *[Content_Types].xml* je vidět na obrázku číslo 2, kde poslední řádek označuje, že v aplikaci je obrázek typu *jpeg* a dále binární soubor. V uvedeném příkladu to byl soubor MP3, který aplikace převedla na binární data. Soubor MP3 je v aplikaci zastoupen odkazem s obrázkem typu *emf*, do souboru tedy musí být přidán default s atributem *extension*, který má hodnotu zmíněného typu *emf*. Soubor obsahuje ještě další tagy default.

```
<Default Extension="bin" ContentType="application/vnd.openxmlformats-officedocument.oleObject" />
<Default Extension="emf" ContentType="image/x-emf" />
<Default Extension="jpeg" ContentType="image/jpeg" />
```

Obrázek 2

Soubor dále obsahuje tag *Override,* který udává reference na soubory XML, jenž archiv obsahuje. Tag se skládá ze dvou atributů, první atribut s názvem *PartName* udává cestu k XML souboru v archivu a *ContentType* udává *vnd* souboru. Na obrázku číslo 3 lze vidět tag k souboru *document.xml* ve složce word, který je hlavní části aplikace.

```
<Override PartName="/word/document.xml" ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml" />
```

Obrázek 3

3.1.1.2 Složka _rels

Obsahuje souboru *.rels.xml,* v němž jsou uvedeny informace, sloužící k definici vztahů jednotlivých souborů obsažených v ZIP archivu a také k definici schémat v Office Open XML aplikace.
Jednotlivé vazby souborů jsou uloženy v tagu *Relationships.* Každá vazba obsažena v tomto tagu musí mít následující atributy:

1. Id
2. Type
3. Target

Id atribut slouží k definici vazby v tagu vazeb, jelikož každá vazba musí být jedinečná.
Type atribut slouží k definici typu vazby v Open Office XML aplikaci.
Target definuje umístění XML souboru, pro který je vytvořena vazba.
[1]
Soubor.rels.xml obsahuje v základní verzi tři vazby. Jsou to vazby typu:

- OfficeDocument – vazba značí hlavní soubor aplikace, tedy soubor document.xml.
- Core-properties – vazba značí soubor core.xml, sloužící k informacím o aplikaci, tedy například název, předmět, kdo aplikaci vytvořil, datum vytvoření či poslední změny a další informace.
- Extended-properties – vazba značí soubor app.xml, sloužící k rozšířeným informacím o aplikaci, například, kolik má aplikace stran, celkový počet slov, celkový počet znaků, odstavců a další vlastnosti, vypovídající o dokumentu.

Vazby *Core-properties* a *extended-properties* jsou uloženy ve složce *docProps*. Ukázka, jak vypadá *soubor.rels.xml* ve složce *_rels* je zobrazen na obrázku číslo 4.

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Target="docProps/app.xml" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties" Id="rId3"/>
    <Relationship Target="docProps/core.xml" Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties" Id="rId2"/>
    <Relationship Target="word/document.xml" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument" Id="rId1"/>
</Relationships>
```

Obrázek 4

3.1.1.3 *Složka word*

Tuto složku obsahuje archiv ZIP v případě, že se jedná o typ aplikace *Word*. Ve složce je obsažen hlavní dokument aplikace s názvem *document.xml*, ten definuje veškerý obsah aplikace, včetně formátování. Na obrázku číslo 5 je znázorněno tělo dokumentu. Dokument obsahuje tag w:*body*, v němž je uložen obsah dokumentu. Tag w:*p* slouží k definici nového paragraphu – odstavce. Odstavec může obsahovat tag *w:t*, jenž má hodnotu textu obsaženého v odstavci.

```
<w:body>
  <w:p>
    <w:r>
      <w:t>Ukazkový dokument</w:t>
    </w:r>
  </w:p>
</w:body>
```

Obrázek 5

Složka word obsahuje mnoho dalších souborů a složek. Lze je výpis několika složek a souborů:

- styles.xml
- settings.xml
- fontTable.xml
- _rels - složka
- media – složka

V rámci diplomové práce není možné popsat všechny soubory, které složka word obsahuje. Bude zde popsán pouze soubor *styles.xml* se sloužkou *_rels*. Pomocí popisu těchto dvou souborů lze vysvětlit pricnip zpracování obsahu hlavního souboru *document.xml.*

Příklad:

V dokumentu je potřeba změnit vícekrát barvu textu na žlutou a font textu na Courier New. Vytvoří se styl s názvem *TestCharacterStyle*, ten se kvůli znovupoužitelnosti uloží do speciálního XML souboru nazvaného *styles.xml*, jenž je uložen ve složce word. Ukázka styles.xml souboru s definovaným stylem je zobrazena na obrázku číslo 7.

Aby bylo možné tento vytvořený styl použít v hlavním dokumentu *document.xml*, je nutné, aby *document.xml* věděl, kde má styl s názvem *TestCharacterStyle* hledat. Proto v adresáři word existuje složka jménem _rels, obsahující soubor *document.xml.rels*, ve kterém jsou vazby na jiné soubory obsaženy.

Na obrázku číslo 6 je zobrazen soubor *document.xml.rels* s vazbou na soubor *styles.xml*, po jejím přidání lze bez problému v hlavním souboru *document.xml* využít definovaný styl TestCharacterStyle.

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml" />
</Relationships>
```

Obrázek 6

Ukázka využití definovaného stylu v hlavním souboru document.xml je následující. Do těla dokumentu, tedy do tagu *w:body* se vloží tag *w:p* zobrazený na obrázku číslo 7.

```
<w:p>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="TestCharacterStyle"/>
    </w:rPr>
    <w:t>character style</w:t>
</w:p>
```

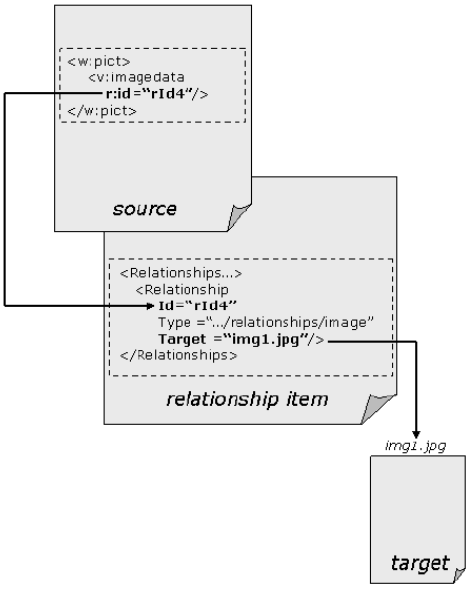
Obrázek 7

Text definovaný v odstavci bude tedy po spuštění aplikace word zobrazen žlutou barvou a mít font Courier New.

3.1.2 Druhy vazeb mezi dokumenty

Explicitní vazba

Hlavní dokument *document.xml* využívá tuto vazbu pomocí jedinečného identifikátoru. V případě, že je do *document.xml* přidán tag obrázku, tedy tag *v:imagedata*, pak musí obsahovat jedinečný identifikátor vazby, na obrázku číslo 8 je identifikátor *id=rId4* v části source.
Tato vazba je definována v souboru *document.xml.rels* ve složce *_rels*. Explicitní vazba v atributu *target* odkazuje přímo na obrázek, který bude přidán. Na obrázku číslo 8, konkrétně je to obrázek s názvem img1.jpg v části **relationship item**.



[6]

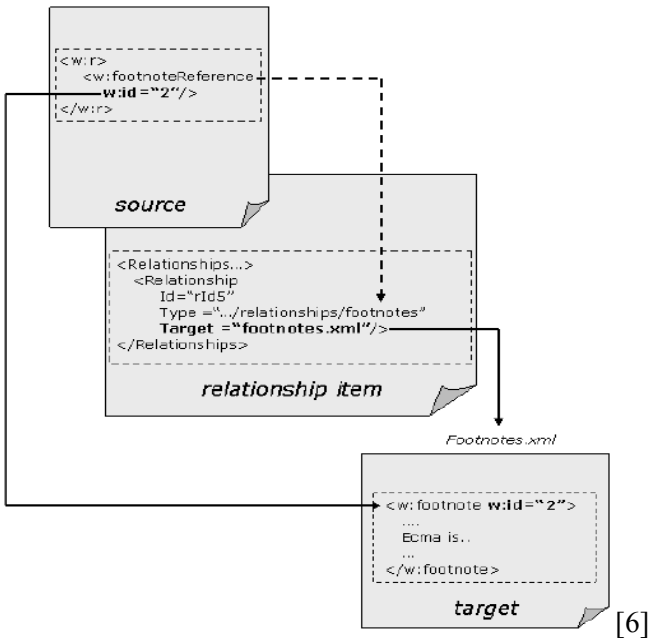
Obrázek 8

Implicitní vazba

V případě, že se v hlavním dokumentu *document.xml* vyskytuje target definovaný standardem Office Open XML a obsahuje id, pak toto id slouží k identifikaci daného prvku v souboru definovaném ve vazbě.

Příklad je zobrazen na obrázku číslo 9. V těle hlavního dokumentu *document.xml* je definován tag *w:footnoteReference* s jedinečným identifikátorem *w:id=2*. Aplikace při překladu zjistí v souboru referencí, ve kterém je v souboru XML vazba s identifikátorem *id=2* vytvořena. Z příkladu na obrázku číslo 9, viz. obdelník **relationship item** je patrné, že je to soubor *footnotes.xm,* definovaný atributem *Target.*

Tag *w:footnoteReference* použitý v hlavním dokumentu *document.xml* bude hledat obsah v souboru footnotes.xml, kde je definován tag w:footnote s *id=2.*



[6]

Obrázek 9

Rozdíl mezi explicitní a implicitní vazbou je ten, že explicitní vazba v referenčním souboru definuje přímo soubor, který má být přidán do hlavního dokumentu *document.xml* a *id* v hlavním dokumentu využívá pouze k identifikaci vazby, naproti tomu implicitní vazba v hlavním dokumentu využívá *id* k identifikaci cíle, který se bude přidávat se souboru definovaného v referenčním souboru. K vyhledání správné vazby se souboru referencí je využit standard OOXML. Například tag *footnoteReference je spojen* s vazbou ../relationships/footnotes. Ke všem ostatním tagům využívající implicitní vazby, jsou v OOXML definovány referenční typy. [6]

3.1.3 Základní prvky dokumentu

3.1.3.1 Odstavec

Je to základní jednotka obsahu celého dokumentu. Při vložení odstavce do textu se vloží do hlavního dokumentu *document.xml* nový element *w:p*, jenž značí právě jeho počátek. Odstavec může obsahovat element *w:pPr*, sloužící k nastavení formátování textu, který obsahuje. Dále odstavec obsahuje runs element (*w:r*). Na obrázku číslo 10 je znázorněn jeden odstavec s elementem *w:pPr,* který nastavuje zarovnání textu doprostřed pro všechny *w:r* element, které obsahuje (na obrázku obsahuje jeden *w:r* element).

```
<w:p >
    <w:pPr>
        <w:jc w:val="center"/>
        <w:rPr>
            <w:b/>
        </w:rPr>
    </w:pPr>
    <w:r >
        <w:rPr>
            <w:b/>
        </w:rPr>
        <w:t>Text uprostřed</w:t>
    </w:r>
</w:p>
```

Obrázek 10

3.1.3.2 Runs element

Prvek slouží ke společné definici stylů a formátování pro část textu odstavce. Ve standardu OOXML je značen jako *w:r* element. Jeden odstavec může obsahovat libovolný počet těchto elementů. Styl pro daný *w:r* element se nastavuje pomocí atributu *w:rPr*.
Text pro daný *w:r* element se ukládá do elementu *w:t*. Pokud tedy *w:r* element obsahuje text, pak musí rovněž obsahovat i *w:t* element.

```
<w:p>
   <w:r>
      <w:rPr>
          <w:b/>
      </w:rPr>
      <w:t xml:space="preserve">Pondělí </w:t>
   </w:r>
   <w:r >
      <w:rPr>
          <w:i/>
          <w:u w:val="single"/>
      </w:rPr>
      <w:t>úterý</w:t>
   </w:r>
</w:p>
```

Obrázek 11

Na obrázku číslo 11 je znázorněn odstavec (*w:p*), který obsahuje dva *w:r* elementy, kde každý má nastaven jiné styly. První *w:r* element s textem nastaveným v elementu *w:t* má nastaven styl tučné písmo (pomocí elementu *w:b*). Druhý *w:r* element má definován text pomocí elementu *w:t* na hodnotu *úterý*. Text má nastaven styl kurzíva s podtržením. Toho je dosáhnuto pomocí elementu *w:i* a *w:u.*

3.1.4 Styly ve WordprocessingML

WordprocessingML umožňuje vytvářet více typů stylů. Všechny tyto typy stylů se ukládají do speciálního souboru jménem *styles.xml*, ve kterém musí být pro každý styl definovány vlastnosti stylu, id stylu a jeho specifické parametry.
Styly ve WordprocessingML se dělí podle jejich použití v dokumentu. Celkově je definováno šest typů stylů. Jsou to tyto typy:

- Charakterový styl
- Odstavcový styl
- Linkový styl
- Tabulkový styl
- Číselný styl
- Defaultní styl

3.1.4.1 Odstavcový styl

Vlastnosti, které jsou ve stylu definovány, se aplikují na celý odstavec, jemuž je daný styl přiřazen. Tedy pokud odstavec obsahuje více *w:r* elementů s textem, pak je styl definován na všechny tyto *w:r* elementy v odstavci. To je rozdíl oproti charakterovému stylu, který lze aplikovat pouze na *w:r* element s textem, nikoliv však na celý odstavec.
Aby se styl pro daný odstavec zapnul, pak musí element značící nový odstavec *w:p* obsahovat element *w:pPr*, jenž obsahuje element *pStyle* s názvem konkrétního odstavcového stylu.

3.1.4.2 Charakterový styl

Tento styl lze aplikovat pouze na *w:r* (runs) element v odstavci. Může být však přiřazen všem *w:r* elementům v odstavci, nikoliv však celému odstavci. U každého *w:r* elementu musí být styl přiřazen jednotlivě, to se provádí pomocí elementu *rStyle*.
Jelikož je styl možné přiradit pouze elementům *w:r*, nikoliv celému odstavci, tak lze u tohoto typu stylu nastavit pouze charakterové vlastnosti, jako je například změna velikosti písma, změna druhu písma a další. Nelze však měnit vlastnosti typické pro odstavcový styl, což jsou například vlastnosti jako zarovnání textu na střed a další.
Na obrázku číslo 7 je vidět aplikovaný styl jménem: TestCharacterStyle na jeden konkrétní *w:r* element.

3.1.4.3 Linkový styl

Je kombinací charakterového a odstavcového stylu. Kombinuje výhody obou dohromady. Umožňuje nastavovat jak vlastnosti pro celý odstavec, tak vlastnosti pouze pro *w:r* (runs) element daného odstavce.
Pokud je linkový styl přiřazen odstavci pomoci *pStyle* elementu v *pPr* elementu, pak jsou na celý odstavec aplikovány vlastnosti, nastavené ve *pPr* daného stylu v souboru *styles.xml*. V případě, že je styl aplikován na jeden či více *w:r* elementů, pak se tomuto elementu nastaví vlastnosti, které jsou v souboru *styles.xml* definovány pro daný styl v atributu *wrPr*, což jsou charakterové vlastnosti stylu.

3.1.4.4 Tabulkový styl

Slouží k definici stylů aplikovaných na tabulky v dokumentu. Může být aplikován na jednu či více tabulek současně. Dělí se na dvě části:

- Tabulkové vlastnosti stylu
- Charakterové vlastnosti stylu

Tabulkové vlastnosti stylu

Pomocí nich lze nastavit orámování, výplň, zarovnání textu, spojení buněk a další vlastnosti. Nastavení pro celou tabulku se přidávají do elementu *tblPr*, vlastnosti takto přidané do *tblPr* se aplikují na celou tabulku, tedy na všechny řádky a sloupce. Element umožňuje definovat styl nejen pro celou tabulku, ale lze definovat styl pro konkrétní sloupec či buňku. Tedy každé buňce tabulky lze nastavit svůj vlastní styl.

Charakterové vlastnosti stylu tabulky

Slouží k definování charakterových vlastností uvnitř tabulky. Tedy nastavují se zde vlastnosti jako je například velikost písma, barva písma, zda má být písmo tučné, či nikoliv a další vlastnosti.

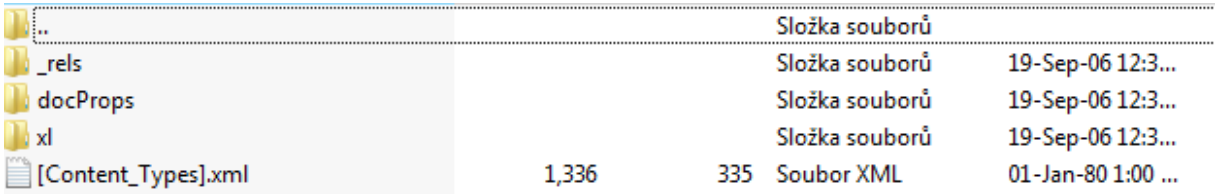
3.1.4.5 Číselný styl

Je v dokumentu aplikován na jakýkoliv číselný nebo odrážkový seznam. Každý takto vytvořený seznam má přiřazen ve *w:pPr* elementu id číslovaného seznamu, jenž je uložen ve speciálním souboru *numbering.xml.*

Pro seznam je vždy vytvořen nový odstavec, tedy *w:p*. Číselný styl lze přiradit pouze jednomu odstavci se seznamem.

3.2 SpreadsheetML

Slouží k uložení excelovských souborů.
Oproti aplikaci Word se liší rozdílnou složkou xl, namísto složky word. Složky *_rels* , *docProps* a soubor *[Content_Types].xml* jsou totožné se soubory používanými v aplikaci Word, tudíž v následující kapitole bude popsána pouze složka xl.



Obrázek 12

3.2.1 Workbook

Česky se nazývá sešit. Je to kolekce k uložení jednoho či více listů. V základní verzi jsou souboru definovány tři listy. Informace o sešitě dokumentu jsou uloženy v souboru *workbook.xml*.

3.2.1.1 workbook.xml

Soubor má definován element *sheets*. Tento element obsahuje všechny listy, které jsou do dokumentu přidány. Prázdný dokument obsahuje jeden list. Způsob uložení informace o listu v elementu *sheets* je zobrazen na obrázku číslo 13. Element *sheet* obsahuje atribut:

- r:Id – relationship identifikátor listu pro soubor *workbook.xml.rels*
- sheetId – jednoznačný identifikátor listu
- name – jméno listu

Prázdný dokument má defaultně předvytvořen jeden list. Při přidání nového listu do dokumentu budou informace o tomto listu uloženy i v elementu *sheets*.

```
<workbook>
    <sheets>
        <sheet r:id="rId1" sheetId="1" name="List1"/>
    </sheets>
</workbook>
```

Obrázek 13

Soubor *workbook.xml* dále tvoří elementy:

- bookViews – specifikuje pohledy pro daný sešit. Jsou v něm uloženy parametry o šířce okna listu a taky o výšce listu.
- fileVersion – obsahuje informace o verzi programu. Například jsou to atributy: lastEdited, backupFile – určující, zda má aplikace provést během uložení zálohu souboru, dále calcMode – indikuje, zda se výpočty budou provádět automaticky, či manuálně, tzn. čekání na akci uživatele
- workbookPr – definuje, které téma bude na listy defaultně aplikováno

3.2.2 Sheets

Excelový dokument si lze představit jako sešit tvořený mnoha listy. Každý takový list je identifikován pomocí jednoznačného názvu. List je reprezentován jako rast tvořený řádky a sloupci, které jsou od sebe odděleny pomocí mřížky. Nejmenší částí listu je buňka, což je průnik jednoho konkrétního sloupce s jedním konkrétním řádkem. Řádky jsou číslovány, sloupce jsou identifikovány pomocí písmen. Buňka přebírá identifikátor z průniku sloupce a řádku. Například je-li buňka průnikem sloupce číslo 5, který má identifikátor písmeno *E* a řádku číslo 9, pak buňka má přidělen identifikátor E9.
V buňkách mohou být uloženy libovolné informace, tedy informace čistě textového charakteru či výpočty, získané z početního úkonu definovaného pro buňku.
Pro každý list je vytvořen ve složce *worksheets* soubor XML s názvem například *sheet1.xml.*

3.2.3 Sheet.xml

Neboli list aplikace excel. Je to nejdůležitější soubor dokumentu. Jsou v něm uloženy veškeré informace z buněk daného listu.

Obsahuje následující elementy:

- Dimension
- sheetViews

- cols
- sheetData

Dimension

Element specifikuje rozsah použitých buněk konkrétního listu. První hodnota určuje buňku, která je v listu nejvíce vlevo a na prvním použitém řádku. Druhá hodnota určuje buňku, jež je nejvíce vpravo a na posledním využitém řádku.

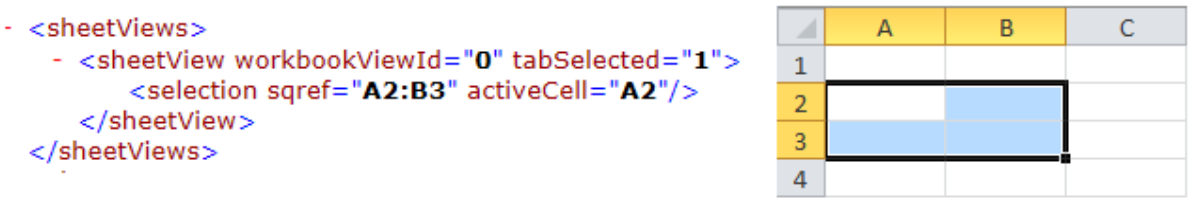
Typ zápisu element dimension: *<dimension ref="B1:N30"/>*. Z příkladu lze vyčíst rozsah využití buněk v daném listu, je to rozsah od buňky B1 až po buňku N30

SheetViews

V případě, když aplikace obsahuje více jak jeden list, musí být po zapnutí jasné, který z těchto listů se má zobrazit. Pokud jsou v aplikaci tři listy a po jejím spuštění se má zobrazit třetí list, pak element *sheetView* třetího listu bude mít hodnotu atributu *tabSelected* rovnou číslu jedna.

Každé definované view v elementu *sheetViews* má svůj jednoznačný identifikátor, jenž je uložen v atributu workbookViewId. Dále obsahuje *view* element *selection*. V Selection elementu jsou v atributech uloženy informace o tom, která buňka bude při zobrazení listu aktivní, tedy bude vybrána, tato vlastnost se nastavuje pomocí atributu *activeCell*. Dalším atributem *selection* elementu je atribut *sqref*, který určuje rozsah vybraných buněk.

Na obrázku číslo 14 je zobrazen příklad uložení elementu *sheetView* s elementem *selection*, který nastavuje pomocí atributu *activeCell* buňku A2 do stavu aktivní – vybrána a pomocí atributu *sqref* nastavuje rozsah výběru od buňky A2 až po buňku B3.



Obrázek 14

Cols

Element obsahuje elementy *col.*

Element *col* nastavuje šířku pro jeden či více sloupců listu a také nastavuje formátování pro sloupec. Údaje k danému sloupci se nastavují pomocí atributů, které jsou uloženy pro daný prvek právě v elementu *col*. Základní atributy jsou:

- width – šířka pro sloupec
- max – udává poslední sloupec, kterému má být nastavena uživatelská šířka
- min – udává první sloupec, u kterého bude měněna šířka
- customWidth – atribut je do elementu *col* přidán v případě, že šířka daného sloupce byla modifikována. Pak se atribut nastaví na hodnotu 1. V opačném případě, tedy tehdy, když sloupec není nijak měněn, se atribut do elementu nepřidává.

Ukázka nastavení rozměrů sloupce pro první sloupec je zobrazena na obrázku číslo 15. Sloupci je nastavena šířka 10.140625

```
<cols>
    <col customWidth="1" width="10.140625" max="1" min="1"/>
</cols>
```

Obrázek 15

SheetData

Element ukládá obsah buněk. Jak již bylo popsáno v kapitole Sheets, list je reprezentován jako rast tvořený řádky a sloupci, které jsou od sebe odděleny pomocí mřížky. Pro ukládání dat tedy byl zvolen takzvaný tabulkový způsob, to znamená, že data buněk se ukládají do řádků a sloupců, kde každá buňka tabulky je přesně identifikována.

Elementu *sheetdata* mohou být přidány pouze elementy *row.* Těch může být libovolný počet.

Element row

Řádek listu. Je uložen v element *sheetData*. Musí mít vždy nastaven identifikátor pomocí atributu *r.* Pokud se mezi dvěma řádky s hodnotami vyskytne prázdný řádek, pak řádek sice bude mít nastaven identifikátor, ale do XML souboru se nezapisuje.

Na obrázku číslo 16 je zobrazen ukázkový příklad, u kterého je v levé části screenshot dat zobrazených v aplikaci a v pravé části je způsob uložení dat v XML souboru.

V příkladu element *sheetData* obsahuje element ***row***. Ten má nastaven atribut *r* s hodnotou 1, tedy id řádku je rovno jedna.

	A	B
1	TV skladem	80
2		

```
<sheetData>
  - <row r="1" x14ac:dyDescent="0.25" spans="1:2">
      - <c r="A1" t="s" s="1">
            <v>0</v>
        </c>
      - <c r="B1" s="1">
            <v>80</v>
        </c>
    </row>
</sheetData>
```

Obrázek 16

Řádek může obsahovat libovolný počet sloupců. Sloupce v řádku jsou ukládány pomocí elementu s názvem *c*.

Element c

Slouží k uložení jedné konkrétní buňky listu. Každá buňka má svůj jednoznačný identifikátor, což je průnik sloupce a řádku, kde se nachází, viz. kapitola Sheets. Hodnota, kterou buňka ukládá je vložena do elementu *v*. Pokud je obsah buňky řetězec, pak se do elementu *v* neukládá přímo tento řetězec, ale index řetězce. Řetězec se uloží do souboru *sharedStrings.xml*. Index je tedy pořadí přidaného řetězce v XML souboru.

Způsob uložení řetězce v souboru *sharedStrings.xml* je podrobně popsán v kapitole Sharedstrings.xml. Aby aplikace poznala, že v dané buňce je uložen řetězec, musí být přidán do elementu *c* atribut *t* s hodnotou *s*, viz. obrázek číslo 17 a to v části index hodnoty, kde je buňka s jednoznačným identifikátorem C1 a obsahuje hodnotu 0, tedy index hodnoty řetězce v souboru *sharedStrings.xml*.

Pokud v elementu *c* není nastaven atribut *t* s hodnotou *s,* pak se hodnota načte přímo z elementu *v*. Na obrázku číslo 17 v části přímá hodnota je v elementu *v,* uloženo číslo 30, což je hodnota, která bude vypsána při spuštění aplikace.

Hodnota do elemenetu *v* může být vložena přímo, nebo nepřímo výpočtem. Pokud první znak při vkládání obsahu buňce je roven znaku =, pak bude hodnota buňky vypočtena z funkce následující za znakem =. Funkce pro výpočet hodnoty se uloží do elementu *f,* jež je součástí elementu *c*. Příklad způsobu uložení funkce v buňce je zobrazen na obrázku číslo 17 v části přímá hodnota. V případě, že je stejná funkce aplikována na více buněk a liší se pouze v hodnotě prvků, které budou ve funkci využity, pak se funkce v buňce, kde byla prvně použita, oindexuje a buňkám, které využívají tuto funkci, se do elementu *f* přidá namísto duplicitní funkce vytvořený index.

Přímá hodnota

```
<c r="D1">
    <f>SUM(A1+B1)</f>
    <v>30</v>
</c>
```

Index hodnoty

```
<c r="C1" t="s">
    <v>0</v>
</c>
```

Obrázek 17

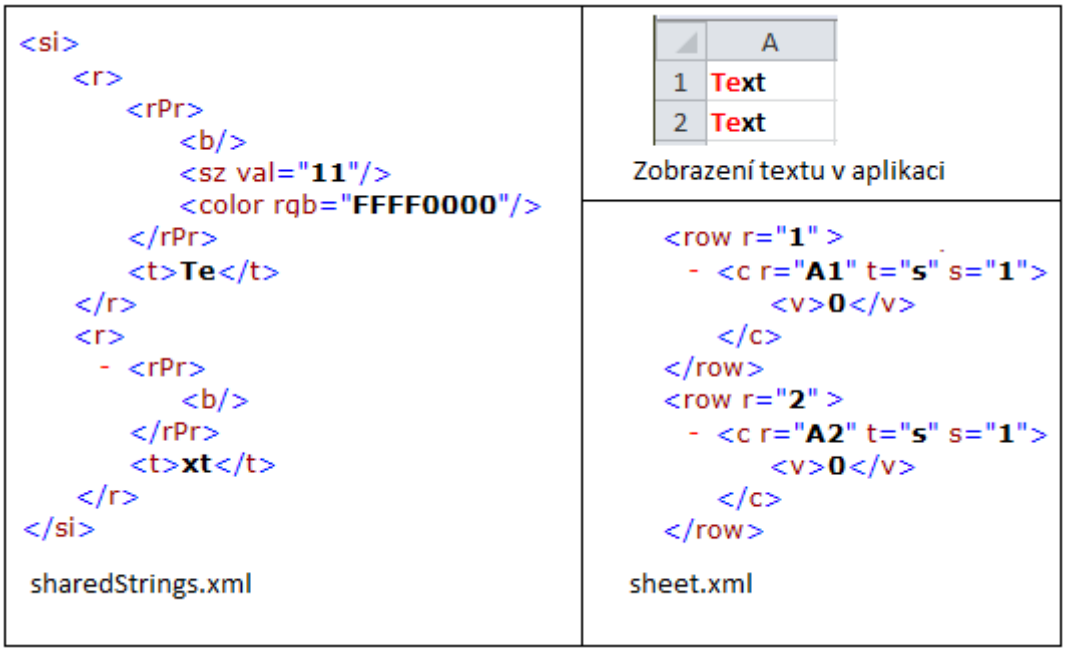
3.2.4 **Sharedstrings.xml**

Do tohoto souboru se vkládají řetězce, které jsou použity ve všech listech aplikace. Důvod uložení řetězců do speciálního souboru namísto uložení přímo do buňky *c* elementu v *sheetData* souboru sheet.xml je ten, že data (řetězce) se v mnoha případech v buňkách opakují a je nežádoucí mít v souboru redundantní data. Proto se vkládanému řetězci při vložení do buňky přidělí číselný index, značící jeho pořadí v souboru *sharedStrings.xml* a tento index se vloží do elementu *v* buňky, kde je text vložen.

Postup při vložení řetězce je tedy ten, že je nejprve zkontrolováno, zda již tento text není uložen v souboru *sharedStrings.xml,* pokud je, zjistí se jeho pořadí v souboru *sharedStrings.xml* a opět se namísto textu vloží do elementu *v* hodnota indexu. V případě, že se text v souboru nevyskytuje, pak se další řetězec přidán do souboru *sharedStrings.xml.*

Jestliže je na řetězec aplikován font, měnící například barvu části řetězce, pak se tyto informace o aplikovaném fontu ukládají společně s daným textem buňky také do souboru sharedStrings.xml.

Na obrázku číslo 18 je zobrazena v pravé vrchní části aplikace s textem, uloženým v buňce A1 a dále v buňce A2. Text v buňce A1 je stejný jako text v buňce A2. Při vložení textu do buňky A1 se řetězec uložil do souboru *sharedStrings.xml*. V souboru zatím nebyl žádný jiný řetězec, proto vložený řetězec má hodnotu indexu rovnou nule. Do elementu *v* elementu *c* v souboru *sheet.xml* pro buňku A1 byl vložen namísto hodnoty *text* jeho index, tedy 0. Při vložení stejného textu do buňky A2 již není vytvořen v souboru *sharedStrigns.xml* stejný element, jelikož by vznikla duplicita, které má být právě pomocí souboru *sharedStrings.xml* zabráněno, proto je pouze načten index element s totožným textem a tento index je nastaven i buňce A2.



Obrázek 18

Každý řetězec ukládaný do souboru *sharedStrings.xml* je rozdělen na tolik části, kolik je aplikováno různých fontů na tento řetězec. Tyto podřetězce se uloží do elementu *r* (run). Vlastnosti aplikovaného fontu jsou uloženy v elementu *rPr* a podřetězec je uložen v elementu *t*.

Na obrázku číslo 18 je font pro podřetězec *Te* z řetězce *Text* uložen v elementu *rPr*. Je zde nastaveno tučné písmo pomocí elementu *b*, velikost písma 11 pomocí elementu *sz* s atributem *val* a v poslední řadě barva textu na hodnotu FFFF0000, což je červená barva.

Na druhý podřetězec *xt* je aplikován font, který aplikuje podřetězci tučné písmo a to pomocí element *b*.

3.2.5 Styles v SpreadsheetML

V excelovské aplikaci lze obsah buněk formátovat vícero způsoby, tedy pomocí:

- Stylů
- Témat
- Přímého formátování

Styly

Všechny definované styly aplikace se ukládají do souboru *styles.xml*. V tomto souboru se nastavují informace, které ovlivňují číselný formát buňky, barvu, font písma, který bude na buňku aplikován a orámování vybrané buňky.

V případě modifikace stylu buňky se vloží atribut *s* s hodnotou čísla stylu do *sheet.xml* souboru pro modifikovanou buňku. Číslo stylu je udáváno pozicí *xf* elementu rodičovského elementu s názvem *cellXfs* v souboru *styles.xml*, kam se tyto styly ukládají.

Při vytvoření nového stylu se přidá do elementu *cellXfs* v souboru *styles.xml* nový element s názvem *xf,* jenž má nastaveny následující atributy

- borderId
- fillId
- fontId
- numFmtId
- xfId

BorderId

Každý soubor obsahuje element *borders,* jehož prvky jsou elementy *border,* číslovány od 0 výše. Jednotlivé elementy *border* nastavují typy ohraničení pro levou, pravou, vrchní a spodní část buňky. Atribut *borderId* tedy udává index elementu *border*, který bude pro daný styl v *xf* elementu aplikován.

FontId

Pro každý styl definovaný pomocí *xf* elementu musí být určen font, který bude aplikován na buňku s daným stylem, to se provádí pomocí atributu *fontId*, který určuje element *font* rodičovského elementu *fonts.* Element font nastavuje buňce například velikost písma, styl písma a další vlastnosti, týkající se úpravy fontu písma.

FillId

Určuje pozici *fill* elementu v rodičovském elementu *fills* ze souboru *styles.xml*. Číslování *fill* elementu je opět od 0 a výše. Pomocí *fill* elementu se nastavuje výplň buňky.

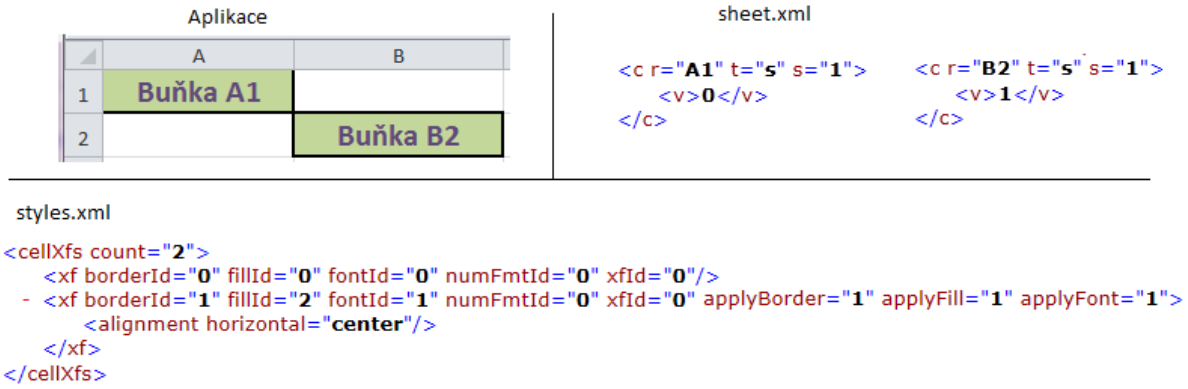
NumFmtId

Slouží k definování výběru *numFmt* elementu z elemetu *numFmts* souboru *styles.xml*. V elementu *numFmt* se nastavuje typ informací, které se v buňce pro tento *numFmt* element můžou vyskytnout. Jako příklad lze uvést element *numFmt*, který umožňuje vkládat do buňky pouze hodnoty, které splňují tvar den/měsíc/rok. Typ zápisu tohoto příkladu je následující:
<numFmt formatCode="d/m/yy;@" numFmtId="166"/>

3.2.5.1 Opětovné využití stylů

Libovolný styl definovaný v souboru *styles.xml* lze aplikovat na více buněk. Důvod je úspora místa v XML souboru, kde je styl vytvořen. Pokud by byl styl vytvořen pro každou buňku, vznikaly by redundantní data tehdy, když by dvě či více buněk měly definován totožný styl.

Na obrázku číslo 19 je uveden konkrétní příklad, kam se styl vytvoří a jakým způsobem se aplikuje na konkrétní buňky. V levé vrchní části obrázku je screenshot z aplikace, ve které je do buněk A1 a B2 přidán text, na nějž je aplikován stejný styl. V souboru *sheet.xml* je pro tyto dvě buňky číslo stylu uloženo, a to konkrétně v elementu *c* pomocí atributu *s*, kde s má hodnotu rovnou jedna.



Obrázek 19

Aplikace tedy zná číslo stylu pro obě buňky, což je zmíněná jednička. Jak bylo popsáno výše, definice konkrétních stylů je uložena v souboru *styles.xml* pomocí elementu *xf* s pořadím čísla stylu plus jedna, jelikož styly se číslují od nuly.

Je tedy načten element *xf* s pořadím 2. Element má nastaven atribut *borderId* roven 1, tedy pro daný styl se načte druhý *border* element z rodičovského elementu *borders* v souboru *styles.xml.*

```
<border>
  - <left style="medium">
        <color indexed="64"/>
    </left>
  - <right style="medium">
        <color indexed="64"/>
    </right>
  - <top style="medium">
        <color indexed="64"/>
    </top>
  - <bottom style="medium">
        <color indexed="64"/>
    </bottom>
    <diagonal/>
</border>
```

Obrázek 20

Načtený *border* je zobrazen na obrázku číslo 20. Ten nastavuje pomocí atributu *left, right, top*, *bottom* a *diagonal* typ ohraničení a to pomocí atributu *style* a dále určuje barvu ohraničení, to provádí pomocí element *color*.

V dalším kroku je načten element *font* pro daný styl. V elementu *xf* je pro daný styl definován font pomocí atributu *fontId*, kde má nastavenou hodnotu rovnou 1, je tedy načten druhý element *font* z elementu *fonts*. Načtený element font je zobrazen na obrázku číslo 21.

```
<font>
    <b/>
    <sz val="16"/>
    <color theme="7" tint="-0.249977111117893"/>
    <name val="Calibri"/>
    <family val="2"/>
    <charset val="238"/>
    <scheme val="minor"/>
</font>
```

Obrázek 21

Daný element *font* nastavuje textu následující vlastnosti:

- tučné písmo – pomocí element *b*
- velikost písma rovnou 16 – pomocí atributu *sz* s atributem *val*
- barvu písma
- jméno fontu písma – pomocí elementu *name* s atributem *val* rovným hodnotě Calibri

  a další vlastnosti

Po načtení font elementu se načte další element, určují výplň buňky. Element se jmenuje *fill.* Je uložen v elementu *fills* souboru *styles.xml*. Element *fills* obsahuje více elementů *fill*, proto je načtena hodnota atributu *fillId* v *xf* elementu daného stylu, udávající pořadí elementu *fill*, který bude aplikován na dané buňky mající nastaven tento styl. Pro hodnotu *fillId* rovnou hodnotě dva, viz obrázek číslo 19, je načten element *fill* s pořadím číslo 3, kvůli indexování elementů od 0. Načtený *fill* element je zobrazen na obrázku číslo 22.

```
<fill>
  - <patternFill patternType="solid">
        <fgColor theme="6" tint="0.39997558519241921"/>
        <bgColor indexed="64"/>
    </patternFill>
</fill>
```

Obrázek 22

Daný *fill* element nastavuje pro buňku následující vlastnosti:

- barvu pozadí
- barvu písma

Buňkám se A1 a B2 se nastaví vlastnosti, načtené z elementů: *fill, font, border a numFmt,* jak je i konkrétně zobrazeno na obrázku číslo 19 v levé horní části.

3.2.5.2 Přímé formátování

Přímé formátování lze na buňku aplikovat pouze tehdy, pokud má buňka nastaven libovolný styl. Tento typ formátování slouží ke změně stylu v podřetězci, na který je aplikován primární styl buňky. Pokud primární styl nastavuje celému řetězci, umístěném v buňce například tučné písmo a pouze jeden znak nemá být tučný, pak se řetězec musí rozdělit na tři nové části, první a poslední část budou mít zděděn styl, který je nastaven primárně pro celou buňku, tedy styl s tučným písmem. Prostřední styl se od první a poslední části bude lišit v tom, že nebude mít nastavenu vlastnost pro zapnutí tučného písma.

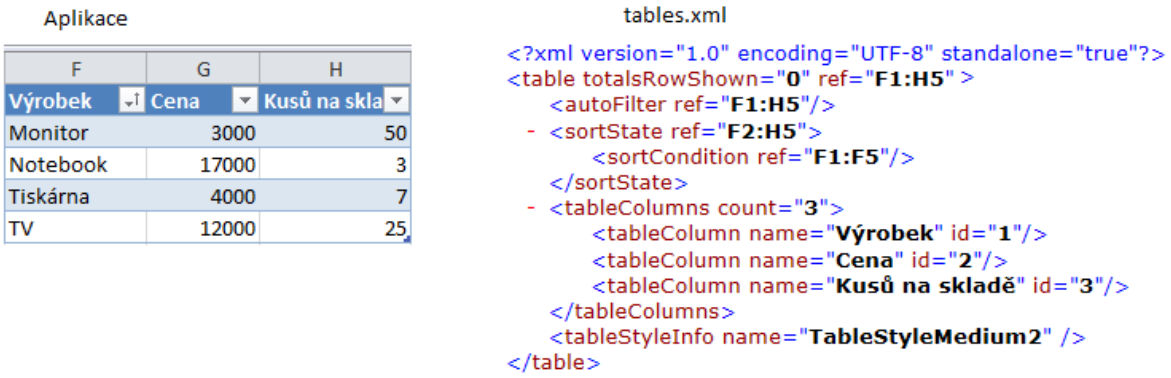
Tyto nově vzniklé řetězce se uloží do souboru *sharedStrings.xml*. V případě, že buňka neobsahuje řetězec, ale obsahuje číselnou hodnotu, pak nelze na tuto číselnou hodnotu aplikovat více stylů, buňka buďto musí mít například definován styl tučného písma na všechny znaky, nebo na žádný, ale nelze nastavit, aby půlka čísla měla nastaven styl tučného písma a druhá nikoliv. Je to způsobeno tím, že číselné hodnoty se do elementu *v* rodičovského elementu *c* dané buňky v souboru *sheet.xml* vkládají přímo, nikoliv odkazem indexu na soubor *sharedStrings.xml,* jak je tomu u řetězců.

3.2.6 Tabulky v SpreadsheetML

Tabulky slouží k formátování, seřazení a analyzování dat, které jsou v nich umístěny. Tabulky lze vytvářen buďto přímo z dat, které jsou již v listu obsaženy, nebo lze data pro tabulku vypočítat podle předem známé funkce. Data vyskytující se v tabulce, nejsou uloženy v žádném speciálním souboru, ale jsou uloženy přímo v *c* elementech daného *sheet.xml* souboru. Informace o tabulce jsou uloženy ve speciálním souboru *table.xml*.

V tomto speciálním souboru jsou uloženy rozměry tabulky, počet jejich sloupců, kde u každého sloupce je uvedeno jeho jméno a jednoznačný identifikátor. Dále je v souboru uložen název tématu, který bude na danou tabulku aplikován. Konkrétní téma je však uloženo v souboru *theme.xml*.

V souboru je dále uvedeno, zda-li, a pokud ano, tak které sloupce se mají seřazovat, ať už vzestupně, či sestupně. Pokud tabulka obsahuje vypočtené informace, pak musí soubor *table.xml* obsahovat elementy, pomocí kterých aplikace zjistí, že informace v tabulce byly vypočteny, nikoliv zadány ručně.



Obrázek 23

Na obrázku číslo 23 je v levé části zobrazena aplikace s tabulkou, která má tři sloupce se jmény: Výrobek, Cena a Kusů na skladu. V pravé části obrázku je XML soubor, sloužící k uložení informací o této tabulce.

4 Implementace

Vytvořený program je navrhnut pro mobilní telefon s operačním systémem Android. Je rozdělen na dvě hlavní části.

První část programu, nazývaná Word, umí pracovat s dokumenty formátu *Docx* definované standardem OOXML. Dokumenty lze v této části programu otevírat ke čtení, dále editovat text a jeho styly zobrazení a následně změny, které byly na textu provedeny, uložit.

Druhá část program, nazývaná Excel, umí otevřít ke čtení či editování dokumenty formátu *Xlsx* ze standardu OOXML. Změny provedené při editaci lze uložit, jak do stávajícího souboru, jenž je editován, tak do nového souboru.

Oba programy, tedy Word i Excel umožňují vytvářet nové dokumenty. Po zapnutí programu je u každé části vytvořen prázdný dokument, do kterého uživatel může ihned psát a změny uložit.

4.1 Word aplikace

Po zapnutí programu Word aplikace načte vzhled, který je definován ve třídě TextEditor. Veškeré další funkční prvky a třídy jsou volány z této třídy.

Po načtení vzhledu vytvoří aplikace všechny struktury, sloužící k uložení objektů, používaných v dokumentu.

Aplikace slouží k:

- čtení obsahu dokumentu
- přidávání textu do dokumentu
- mazání textu v dokumentu
- editaci stylů textu

4.1.1 Objekty dokumentu

Dokument WordprocessingML standardu OOXML je založen na ukládání odstavců pomocí elementů *w:p* a v nich *w:r* elementů nazvaných runs, viz kapitola: Runs element. V rámci vytvořeného programu byly pro uložení informací o elementech *w:p* a *w:r* vytvořeny speciální třídy nazvané:

- Paragraf.java
- WR.java

Paragraf.java

Třída reprezentuje objekt *w:p* ze standardu OOXML. V programu je jeden *w:p* element odstavec textu. U každého tohoto odstavce je třeba uložit jeho jednoznačný identifikátor, pozici začátku odstavce, všechny jeho *w:r* elementy, které odstavec obsahuje. Ty jsou uloženy ve speciálním ArrayListu s názvem *a_wr_elements.* Dále třída ukládá ostatní elementy, které se objevily v elementu *w:p.*

WR.java

Třída reprezentuje objekt *w:r* ze standardu OOXML. Element *w:r* musí být vždy součásti odstavce. Popis *w:r* element lze nalézt v kapitole: Runs element. Aplikace pro každý *w:r* element eviduje tyto vlastnosti:

- Číslo odstavce, do kterého *w:r* náleží
- Délka textu ve *w:r*
- Pozice prvního znaku textu v aplikaci
- Pozice posledního znaku v textu
- Text *w:r* elementu – text obsažený ve *w:t* elementu
- Jednoznačný identifikátor *w:r* element – čísluje se od 0 výše
- Ostatní elementy ve *w:r* element

4.1.2 Rozbalení dokumentu

Dokument formátu *Docx* je zazipovaný soubor skládající se z XML souborů. Při práci s dokumentem je tedy potřeba tento dokument rozbalit a uložit v zařízení s OS Android na paměťové médium. Při rozbalení zazipovaného souboru jsou vytvořeny 2 složky jménem *origDocx*a *NewDocx*, kde ve složce *origDocx* jsou uloženy originální XML soubory z archivu dokumentu a ve složce *newDocx* jsou uloženy také originální soubory z archivu i s úpravami, které na těchto souborech byly provedeny.

4.1.3 Princip zpracování informací v programu

Při tvorbě nového či editaci načteného dokumentu se v aplikaci vytvoří ArrayList jménem *a_all_paragraf.* Pokud je načten dokument, pak se do ArrayListu *a_all_paragraf* přidají všechny elementy *w:p*, které daný dokument obsahuje. *W:p* elementy jsou uloženy v dokumentu jménem

document.xml. Metoda načítající elementy *w:p* současně pro každý odstavec volá metodu, sloužící k načtení jeho *w:r* elementů.

Pro čtení XML souboru a jeho editaci je v aplikaci vytvořena třída jménem *Zpracuj_Document_xml.java*

Důvod, proč se v paměti vytváří objekty s ArrayListem odstavců a jejich runs elementů je ten, že při editaci textu by bylo časově i výkonnostně náročné editovat obsah XML souboru, ve kterém je text uložen, proto jsou objekty načteny do paměti. Veškeré změny, které jsou na objektech uložených v paměti provedeny, musí mít ekvivalentně uloženy i do XML souboru.

4.1.3.1 Zpracuj_Document_xml.java

Tato třída podporuje čtení a editaci XML prvků uložených v souboru. Jsou v ní předvytvořeny metody, které aplikace využívá pro zpracování dokumentu. Stručný seznam metod:

- getAllParagraf() – načte všechny odstavce v dokumentu
- getAllR(int idParagraf) – načte všechny *w:r* elementy odstavce
- removeParagraf(int idParagraf) – smaže odstavec
- createParagraf(int idParagraf) – vytvoří paragraf s předaným id, odstavců za tímto nově vytvořeným odstavcem změní jejich id – zvětší jej o jedna
- addOrChangeTextIn_wt – přídá či změní text ve *w:r* elementu

4.1.4 Načtení textu z dokumentu do aplikace

Po uložení všech odstavců a *w:r* elementů z dokumentu *document.xml* do paměti zařízení, kde je aplikace spuštěna, lze tyto načtené objekty zobrazit. K zobrazení obsahu jsou využity prvky z knihoven Android.

Text je načten konkrétně do objektu EditText jménem *contentEdit*, který umožňuje jak zobrazení textu, tak také jeho editaci. Přidávání textu do EditTextu probíhá tak, že se v cyklu procházejí všechny odstavce, uložené v paměti v ArrayListu jménem *a_all_paragraf*. Pro každý procházený odstavec je v EditTextu vytvořen nový řádek. Pokud takto procházený odstavec obsahuje jeden či více *w:r* elementů uložených v ArrayListu *a_wr_element,* pak je pro každý *w:r* element načtena jeho instance a získán text, který je v tomto objektu uložen. Tento získány text je následně přidán do EditTextu *contentEdit*. Popsaným způsobem se přidá text pro všechny odstavce, uložené v paměti.

4.1.5 Editace textu dokumentu

Editace textu v dokumentu je velmi složitá operace. Po editování textu je měněn obsah objektů načtených do paměti, nikoliv přímo XML soubor, viz kapitola: Princip zpracování informací v programu. V následujících podkapitolách budou postupně vysvětleny problémy, které během editace textu nastávají.

Přidání jednoho znaku

Přidání jednoho znaku je v editaci nejjednodušší operace. Aplikace při vložení znaku vyhledá podle pozice v EditTextu odstavec, do kterého byl znak vložen. Pokud vyhledaný odstavec obsahuje ArrayList s *w:r* elementy, pak se přímo podle pozice přidání znaku nalezne i *w:r* element. Tomuto nalezenému *w:r* elementu se změní text, který obsahuje na nový text včetně přidaného znaku. U modifikovaného elementu se změní následně jeho délka textu, počáteční, případně koncová pozice.

V případě, že načtený odstavec neobsahuje žádný *w:r* element, pak musí být pro přidávaný znak tento element vytvořen. Nový element má nastaven text přidávaného znaku.

Smazání jednoho znaku

Jestliže je z EditTextu smazán libovolný znak, provedou se následující operace. Nejprve je zjištěno, zda mazaný znak je přímo odstavec, pokud ano, pak se všechny jeho objekty, které obsahuje, překopírují do předchozího odstavce (pokud neexistuje předchozí odstavec, tak se vytvoří). Po překopírování se u všech nových *w:r* elementů změní jejich jednoznačný identifikátor. Nové id udává jejich současnou pozici v novém odstavci. U každého přidaného *w:r* elementu se dále mění číslo odstavce, do kterého patří. Změna čísla odstavce se dále provede i pro *w:r* elementy, které se nacházejí v odstavcích za smazaným odstavcem. I u nich je nutné přečíslování.

Pokud mazaný znak není odstavec, ale je součástí textu ve *w:r* elementu, tak se pro mazaný znak nalezne jeho odstavec, ze kterého se načte element *w:r,* kterému se změní text (z textu se smaže znak). Po změně textu se změní i velikost délky textu, kde se od aktuální délky odečte číslo jedna.

Smazání více znaků

Při mazání více znaků mohou nastat 2 případy. Oba jsou vysvětleny v následujících dvou podkapitolách.

Mazání více znaků z jednoho runs elementu

Případ, kdy jsou mazány znaky v rámci jednoho *w:r* elementu v libovolném odstavci. U modifikovaného elementu je změněn jeho text. Ten se změní tím způsobem, že se z něho odstraní znaky, které byly vymazány v EditTextu. *W:r* elementu se změní jeho délka textu.

Mazaní více znaků z dvou a více runs elementů

V tomto případě se nejprve změní text pro první a následně pro poslední vybraný *w:r* element. Prvním elementem se rozumí element, který stojí na začátku výběru mazaných znaků a posledním elementem se rozumí element, jenž stojí na konci mazaných znaků. Jestliže se mezi prvním a posledním *w:r* elementem nacházejí i jiné *w:r* elementy, pak budou smazány. Při mazání dvou a více elementů může nastat situace, že první vybraný a poslední *w:r* element nejsou součástí jednoho odstavce, ale každý z nich je umístěn v jiném odstavci. Pokud tento případ nastane, pak se musí všechny elementy od posledního elementu *w:r* (včetně něho), překopírovat do odstavce, kde se nachází první *w:r* element, konkrétně se *w:r* kopírují za tento první element, a to přesně v tom pořadí, v jakém se nacházejí v původním odstavci.

Po překopírování elementů dochází i k jejich přečíslování, kde každému nově zkopírovanému elementu je přiřazen jednoznačný identifikátor značící jeho aktuální pozici v odstavci. Dále je u těchto prvků změněno číslo odstavce, do kterého patří. Tato změna čísla odstavce se provede i u všech *w:r* elementů nacházejících se v odstavcích za posledním vybraným *w:r* elementem.

4.1.6 Editace stylů

Aplikace umožňuje editovat styl buďto pouze pro jeden znak, nebo pro libovolný počet znaků. Umožňuje následující změny stylů:

- nastavení či smazání tučného písma textu
- nastavení či smazání kurzívy textu
- nastavení či smazání podtržení textu
- změnu velikosti písma textu
- změnu barvy textu

Změna stylu pro jeden znak

Při změně stylu u jednoho znaku se načte pro modifikovaný znak jeho *w:r* element, do kterého patří. Aby bylo možné změnit styl u vybraného znaku, je potřeba načtený *w:r* element rozdělit na více částí. Na kolik částí se rozdělí, je určeno tím, který znak má být modifikován. Mohou nastat následující případy:

- Modifikace stylu prvního znaku textu z runs elementu
- Modifikace stylu mezi prvním až posledním znakem elementu
- Modifikace stylu posledního znaku elementu

Jelikož pro každou situaci vzniká jiný počet elementu, budou všechny tři body podrobně rozebrány v následujících podkapitolách.

Modifikace stylu prvního znaku textu z runs elementu

Při této situaci je *w:r* element, ve kterém se modifikuje znak, upraven. Bude mít délku textu rovnou jedné a textu je přiřazena hodnota, jež byla vybrána k modifikaci stylu. Po úpravě původního *w:r* vzniká nový *w:r* element, jenž bude mít hodnotu textu z původního *w:r* bez prvního znaku. Tento nově vzniklý element se přidá ihned za původní *w:r* element. Nově vytvořený styl má stejné vlastnosti, jaké měl nastaven původní *w:r* element před rozdělením. Po rozdělení je původnímu, nyní už upravenému *w:r* elementu změněn styl, který uživatel aplikace pro následující znak vybral. Může to být například změna velikosti písma daného znaku.

Modifikace stylu posledního znaku textu z runs elementu

Toto je obdobný příklad, jako je popsán v Modifikace stylu *prvního znaku textu z runs elementu*. Opět vzniká nový *w:r* element ze zbytku textu, který nebyl vybrán ke změně stylu, tento element má stejné stylové vlastnosti jako původní element. Následně je původnímu elementu změněn text na hodnotu vybraného znaku k modifikaci a přidán či odebrán styl nastavovaný pro tento znak.

Modifikace stylu mezi prvním až posledním znakem elementu

V tomto případě se vytvoří dva nové *w:r* elementy. První *w:r* element se vloží před původní element se znakem vybraným pro změnu stylu, tomuto elementu bude nastaven text z původního elementu od počátku až po znak vybraný k modifikaci. Tento element zdědí stylové nastavení původního elementu. Druhý nově vzniklý element taky zdědí nastavení stylů z původního elementu,

text u tohoto elementu je získán z textu původního elementu od znaku k modifikaci stylu až po konec textu. Druhý vytvořený element se vloží za původní element. Styl je změněn pouze u původního elementu podle toho, jaký styl uživatel pro znak zvolil. Původnímu elementu je změněn text na hodnotu vybraného znaku. Délka u původního elementu je také modifikována.

4.1.6.1 Změna stylu pro více znaků

U změny stylů pro více znaků mohou nastat dva případy. První z nich je případ, kdy se bude měnit styl více znakům v rámci jednoho *w:r* elementu. V druhém případě nastává situace, kdy se bude měnit styl znakům v rámci dvou a více *w:r* elementů.

Modifikace stylu pro více znaků v rámci jednoho runs

Pro vybrané znaky ke změně stylu se nalezne jejich *w:r* element, do kterého patří. Jestliže je pro modifikaci znaků vybrán celý element, pak se mu změní styl podle uživatelem nadefinovaného stylu v aplikaci.

Situaci, kdy jsou vybrány znaky od druhého znaku výše, až po konec elementu je obdobná, jako situace popsána v kapitole jménem: Modifikace stylu posledního znaku textu z runs elementu. Rozdíl je v tom, že na rozdíl od zmíněné kapitoly se nemění styl pouze u posledního znaku, ale mění se pro všechny vybrané znaky. Původní *w:r* element je zkrácen na část textu, která nebyla vybrána k modifikaci stylu. Za původním *w:r* elementem vzniká nový element, jenž má nastaven text vybraný k modifikaci stylu. Tento nově vzniklý element přebírá všechny stylové nastavení, jako má nastaven původní element a navíc je mu přidán styl, definován uživatelem v aplikaci.

Další situace je ta, že ke změně stylu je vybrána část slova mezi prvním a posledním znakem textu. V tom případě vznikají dvě nové slova. Situace je opět obdobná jako při změně stylu pro jeden znak, popsané v kapitole: Modifikace stylu mezi prvním až posledním znakem elementu. Rozdíl je pouze ten, že u původního slova není délka textu zkrácena na jeden znak, ale je upravena na délku textu vybraného k modifikaci, tomuto textu je nastaven styl, který byl vybrán uživatel v aplikaci.

Modifikace stylu pro více znaků v rámci dvou a více runs

Při modifikaci stylů u dvou a více runs elementů mohou nastat tyto případy:

- Je vybrán celý první *w:r* element a celý poslední *w:r* element
- Je vybrán celý první *w:r* element a část posledního *w:r* elementu.
- Je vybrána část prvního *w:r* elementu a celý poslední *w:r* element
- Je vybrána část prvního *w:r* elementu a část posledního *w:r* elementu

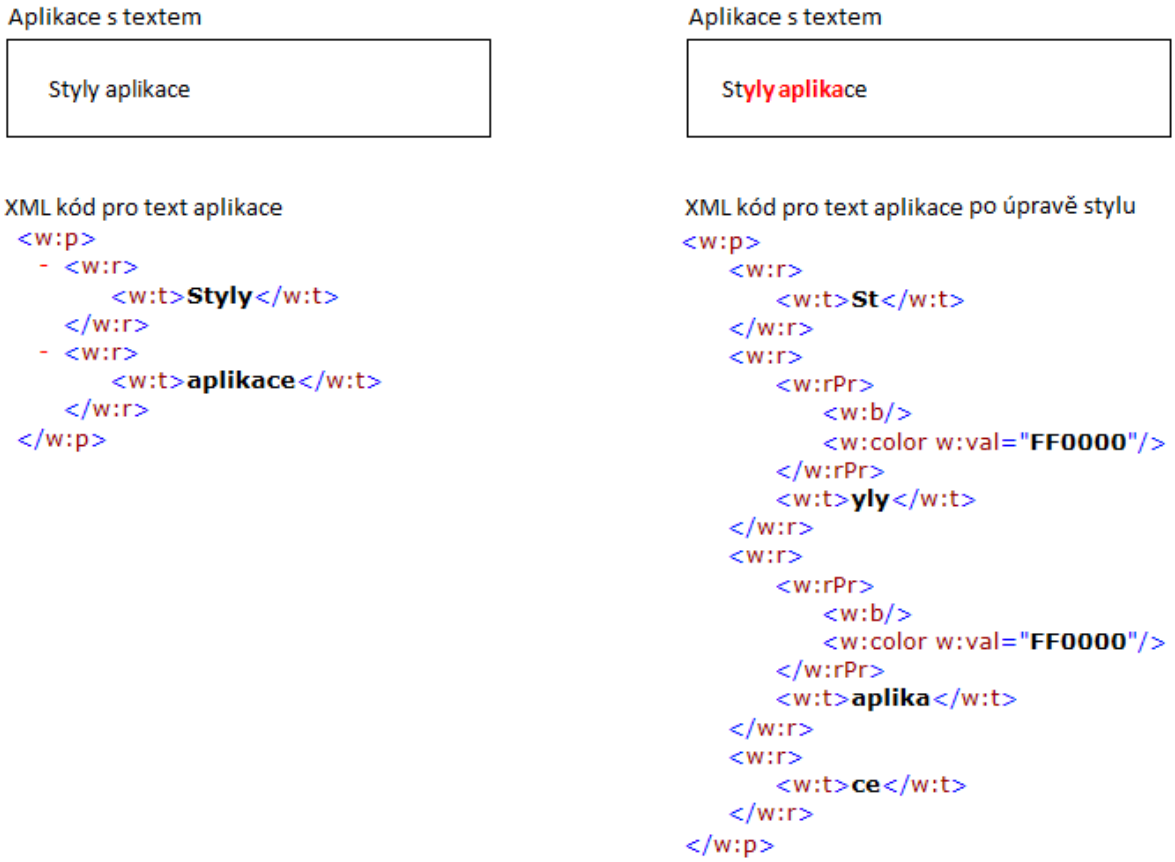
Pokud je vybrán celý první *w:r* element a celý poslední *w:r* element. Pak se změní styl oběma těmto *w:r* elementům a všem elementům nacházejících se mezi těmito vybranými elementy.

Jestliže je vybrán celý první *w:r* element a část posledního *w:r* elementu. Pak poslední element bude zkrácen na část, která byla vybrána ke změně stylu. Ze zbytku textu z posledního elementu, který nebyl vybrán ke změně stylu, vzniká nový *w:r* element. Ten dědí stylové nastavení posledního elementu. Následně je změněn styl definovaný uživatelem v aplikaci pro první až poslední *w:r* element.

Obdobná situace vzniká, pokud je vybrána část prvního *w:r* elementu a celý poslední *w:r* element. V tomto případě se před první slovo vytvoří nový *w:r* element, jemuž je nastaven text z prvního *w:r* elementu, který nebyl vybrán ke změně stylu. Nový element dědí stylové nastavení z prvního elementu. Následně je prvnímu elementu zkrácen text na hodnotu, která byla vybrána ke změně stylu. Všem elementům od prvního až po poslední element je změněn styl definovaný v aplikaci.

Poslední situace, která může při změně stylů pro více než jeden element nastat, je ta, když je vybrána část prvního *w:r* elementu a část posledního *w:r* elementu. Pak se před první element vytvoří nový *w:r* element s textem z prvního elementu, jenž nebyl vybrán ke změně stylu. Tento element dědí stylové nastavení z prvního elementu. Dále se za poslední element vytvoří nový element, jemuž je nastaven text, který nebyl vybrán v posledním slově ke změně stylu. Elementu je nastaven totožný styl posledního elementu. Dále je prvnímu a poslednímu element zkráceno na text, který byl pro tyto elementy vybrán v EditTextu ke změně stylu. V posledním kroku je změněn styl nadefinován uživatelem v aplikaci, konkrétně od prvního až po poslední element.

Na obrázku číslo 24 je zobrazena situace, kdy je měněn styl ve více *w:r* elementech. V levé části je zobrazen text aplikace, který je v XML souboru *document.xml* uložen pomocí dvou *w:r* elementů. První *w:r* element má nastaven text „*Styly“* a druhý element má nastaven text „*aplikace“*. V pravé části obrázku je text, u kterého byly aplikovány změny stylu. V XML souboru pro daný text přibyly dva nové *w:r* elementy, jelikož nastala situace, že byla vybrána část prvního *w:r* elementu a část posledního *w:r* elementu ke změně stylu. Způsob vzniku dvou nových elementů je popsán o odstavec výše.



Obrázek 24

4.1.7 Uložení změn

Po ukončení editace textu lze provedené změny uložit. Aplikace projde všechny XML soubory, které byly modifikovány a vytvoří nový dokument formátu *Docx* s předaným názvem.

4.2 Excel aplikace

Aplikace slouží k úpravě dokumentů ve formátu XLSX standardu OOXML. Po jejím zapnutí se nahraje vzhled, jenž je definován ve třídě *ExcelActivity.java*. Aplikace je jedna velká tabulka, skládající se z libovolného počtu řádků a sloupců. V základní verzi obsahuje 14 řádku a 5 sloupců. Sloupce i řádky lze do tabulky dynamicky přidávat. Tento počet je zvolen z důvodu lepšího výkonnostního běhu aplikace na zařízeních s OS Android.

Průnikem jednotlivých sloupců a řádků je buňka, do které lze vkládat text. V Aplikaci je pro tuto funkci zvolena komponenta z knihovny Androidu. Konkrétně je to komponenta EditText. V základní verzi tedy aplikace obsahuje tabulku se čtrnácti řádky, kde každý řádek má přidán pět

instancí komponenty EditText. Pokud však uživatel potřebuje více řádku než je základní počet 14, pak se do tabulky přidá požadovaný počet řádků, obsahující instance nových komponent EditTextu.

Každá buňka má svou instanci komponenty EditText, je to z důvodu, aby se přesně identifikovat a lokalizovat.

Sloupce mají nastaven identifikátor písmena anglické abecedy a řádky mají nastaveny číslice. Buňka, která je dána průnikem třetího řádku s pátým sloupcem má tedy nastaven identifikátor ve tvaru E3, jelikož E je pátý znak abecedy a 3 identifikuje 3 řádek.

Po inicializaci komponenty TableLayout, což je popsaná hlavní tabulka aplikace a buněk s komponenty EditText může uživatel aplikace editovat obsah jednotlivých buněk.

Aplikace slouží k:

- Čtení obsahu dokumentu
- Přidávání obsahu do buněk dokumentu
- Mazání obsahu buněk či jen jeho části

4.2.1 Struktura aplikace

Hlavní třídou aplikace je třída ExcelActivity.java. Tato třída obsahuje HashMapu jménem *hm_idBunkyAWrExcel*, která slouží k uložení jednoznačného identifikátoru buňky a jejího obsahu. Obsah pro jednotlivou buňku je ukládán do ArrayListu, který je naplněn instancemi třídy WR_Excel.java.

Pokud je v buňce vložena číselná hodnota, pak je pro tuto buňku vytvořena pouze jedna instance třídy WR_Excel.java. V případě, že buňka neobsahuje číselnou hodnotu, ale text, pak se pro danou buňku vytvoří takový počet instancí třídy WR_Excel, kolik má text obsažený v buňce nastavený počet stylů. V případě, že buňka obsahuje tři slova a každé slovo má nastaven jiný druh stylování, pak se do ArrayListu pro danou buňku vloží tři instance třídy WR_Excel.java, kde první instance bude mít nastaven text prvního slova, druhá instance text druhého slova a v poslední instanci bude text třetího slova. Pořadí přidání instancí do ArrayListu musí být zachováno.

WR_Excel.java

Tato třída slouží k uložení informací o obsahu dané buňky.

Ve třídě jsou evidovány následující informace:

- Zda je prvek číslo či řetězec
- Jednoznačný identifikátor *w:r* elementu
- Začáteční pozice textu

- Koncová pozice textu
- Text

4.2.2 Přidání textu do buňky

V této kapitolu bude popsán proces, který se děje po přidání obsahu do prázdné buňky. Aplikace nejprve zjistí pomocí metod v nich nadefinovaných, zda vkládaný obsah do buňky je řetězec nebo číslo.

Pokud je vkládaný obsah číslo, pak se vytvoří nová instance třídy *WR_Excel*, které se nastaví, že obsah proměnné text této instance je číslo, nikoliv řetězec a dále se proměnné text nastaví vložené číslo uživatelem do buňky. Tato instance se uloží následně do ArrayListu, který je uložen do HashMapy *hm_idBunkyAWrExcel,* kde identifikátor vloženého záznamu je právě id buňky s novým textem a hodnota je ArrayList s instancí *WR_Excel*.

Pokud vkládaný text je řetězec, nikoliv číslo, pak se opět pro tento řetězec vytvoří nová instance třídy *WR_Excel*, ovšem s tím rozdílem, že se u instance nastaví proměnné *isString* hodnota true, značící, že hodnota uložená v proměnné *text* je řetězec. Dále se do proměnné *text* vloží hodnota, přidaná uživatel do buňky. Vytvořená instance se uloží do Arraylistu, jenž je přidán do *hm_idBunkyAWrExcel.*

Hodnota o nově vložené buňce se uloží do ArrayListu, který se v případě uložení dokumentu projde a každou buňku i s jejím obsahem přidá do XML souboru *sheet.xml,* sloužící právě k uchování informací o buňkách.

4.2.3 Editace obsahu buňky

Pokud uživatel aplikace začne editovat text buňky, pak je pro tuto buňku načten ArrayList obsahující jeho *WR_Excel* instance. U načtených instancí je zjišťováno, zda je jejich obsahem číslo nebo řetězec. Dále je zjištěno, zda přidávaný znak je číslo nebo řetězec. Mohou nastat tři případy:

- Do buňky s číslicemi je vložena číslice
- Do buňky s číslicemi je vloženo písmeno
- Do buňky s řetězcem je vložen řetězec či znak

Do buňky s číslicemi je vložena číslice

Pro vkládaný znak se načte její instance *WR_Excel* podle pozice vkládaného znaku a změní se u ní obsah proměnné *text* na hodnotu s přidanou číslicí v buňce, tudíž celé číslo v buňce bude totožné s číslicí v proměnné *text.*

Do buňky s číslicemi je vloženo písmeno

Nejprve je načtena instance *WR_Excel* pro tuto buňku. U instance se změní hodnota proměnné *isString* z hodnoty false na hodnotu true, jelikož po přidání písmena již buňka nesplňuje podmínku, že její obsah je číslo. Dále se vloží hodnota textu, která je v buňce, do proměnné *text*. Do ArrayListu, který eviduje provedené změny v obsahu buněk se uloží záznam o tom, že v případě uložení dokumentu se v souboru *sheet.xml*, kde je buňka uložena, musí změnit obsah *v* elementu buňky rodičovského elementu *c* a dále nastavit tomuto rodičovskému elementu atribut *t=s*, značící, že obsah buňky je řetězec, nikoliv číslo.

Do buňky s řetězcem je vložen řetězec či znak

Nastane-li tento případ, pak se načte instance *WR_Excel* pro slovo v buňce, do kterého byl přidán znak. Pro načtenou instanci se u proměnné *text* změní hodnota textu tím způsobem, že do současné hodnoty, kterou proměnná s textem obsahuje je přidán znak, který byl uživatelem přidán v EditTextu. Opět se do ArrayListu evidující změny v obsahu buněk uloží záznam o tom, že obsah buňky byl změněn a musí být uloženy i do XML souborů, které tyto informace uchovávají.

4.2.4 Výpočty

Aplikace neumožňuje zadávat do buněk rovnice, které mají být dopočítány. Je to způsobené tím, že ve standadtu OOXML jsou popsány stovky funkcí. Tyto funkce však nejsou implementovány. Aby mohla aplikace využívat libovolné funkce, musela by je mít nejprve naimplementovány, což při tak velkém počtu funkcí není v rozsahu diplomové práce časově možné.

5 Závěr

Cílem práce bylo vytvořit aplikaci pro operační systém Android, která bude umožňovat čtení a editování dokumentů *Docx* a *Xlsx* standardu OOXML. Aplikaci jsem vytvořil a rozdělil na dvě části, kde první část umožňuje číst, editovat a vytvářet nové textové soubory. Editací je myšleno přidávání či mazání textu z jakékoliv části dokumentu a také změna formátování textu, například změna barvy či velikosti písma.

Druhá část aplikace umožňuje číst, vytvářet a editovat obsah dokumentů formátu *Xlsx*, neumožňuje však měnit hodnoty buněk, ve kterých se nachází výpočetní funkce. Důvod je ten, že specifikace SpreadsheetML definuje stovky výpočetních funkcí a v rámci této práce nebylo možné se se všemi funkcemi seznámit a následně je implementovat.

Aplikace není vhodná pro práci s většími dokumenty, jelikož musí načítat mnoho souborů XML obsažených v archivu ZIP dokumentu, což je v operačním systému Android velmi procesorově náročná operace.

V dalším bodu práce jsem analyzoval standard Open Office XML a podrobně popsal jeho dvě specifikace, sloužící právě k práci s dokumenty formátu *Docx*a *Xlsx*.

Součástí teoretické části práce bylo popsání již existujících programů pro práci s OOXML v operačním systému Android a následné porovnání těchto popsaných souborů, s mnou vytvořenou aplikací, toto jsem popsal v kapitole:

Práce obsahuje také návrh a popis principu vytvořeného programu.

Tato práce pro mě byla přínosná tím, že jsem se naučil vytvářet aplikace pro operační systém Android a dále jsem se zdokonalil v programování v jazyce Java, jelikož veškerá funkcionalita aplikace je naprogramována právě v tomto jazyce.

V práci jsem čerpal z odborné literatury a článků na internetu.

Literatura

[1] MICROSOFT. *Office - Office.com* [online]. 2007 [cit. 2012-05-02]. Dostupné z: http://office.microsoft.com/cs-cz/excel-help/formaty-office-open-xml-i-podrobne-informace-o-formatech-office-open-xml-RZ010243529.aspx?section=14

[2] *Documents To Go™*: *User Manual for Android* [online]. 2010 [cit. 2012-05-03]. Dostupné z: http://download.dataviz.com/pdf/manuals/dxtg_android_manual.pdf

[3] Iphone-dataviz-docs-to-go. *Mujmac.cz* [online]. 2011 [cit. 2012-05-02]. Dostupné z: 3. http://www.mujmac.cz/art/mm_rss/iphone-dataviz-docs-to-go-17-6-09.html?tisk=on

[4] QUICKOFFICE. Quickoffice_pro_android_5.0_guide [online]. 2010 [cit. 2012-05-03]. Dostupné z: http://support.quickoffice.com/index.php?_m=downloads&_a=viewdownload&downloaditemid=91&nav=0,16

[5] Office Open XML. *Wikipedie* [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://cs.wikipedia.org/wiki/Office_Open_XML

[6] ECMA TC45. *Office Open XML*: *Part 3: Primer* [online]. 2006 [cit. 2012-05-03]. Dostupné z: http://www.ecma-international.org/news/TC45_current_work/Office%20Open%20XML%20Part%203%20-%20Primer.pdf