VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

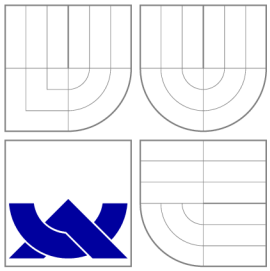
INFORMAČNÍ SYSTÉM NA CHYTRÉM TELEFONU PRO UČITELE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

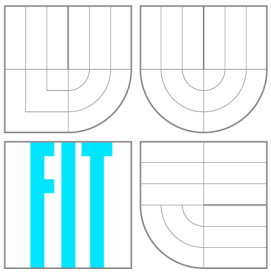
AUTOR PRÁCE
AUTHOR

VOJTĚCH ŠIMŠA

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INFORMAČNÍ SYSTÉM NA CHYTRÉM TELEFONU PRO UČITELE

INFORMATION SYSTEM ON SMARTPHONE FOR TEACHERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VOJTĚCH ŠIMŠA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2014

Abstrakt

Cílem této bakalářské práce je vytvořit informační systém pro učitele efektivně použitelný na mobilních zařízeních. Návrh umožňuje využití na základních a středních školách, zohledněna je rozdílnost požadavků různých stupňů vzdělávání i jednotlivých učitelů. V průběhu textu je popsán základní přístup k tvorbě aplikací pro platformu Android a jeho praktická aplikace v tomto projektu.

Abstract

The aim of this work is to create an information system for teachers, that would be effectively useable on mobile devices. The system is designed to be used at primary and secondary schools and reflects the diversity of demands for different levels of education and individual teachers. The text describes the basic approach to programming for the Android platform and its practical application in this project.

Klíčová slova

Android, informační systém, učitel, mobilní aplikace, SQLite

Keywords

Android, information system, teacher, mobile application, SQLite

Citace

Vojtěch Šimša: Informační systém na chytrém telefonu pro učitele, bakalářská práce, Brno, FIT VUT v Brně, 2014

Informační systém na chytrém telefonu pro učitele

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D.

.....
Vojtěch Šimša
19. května 2014

Poděkování

Vedení Základní školy a Mateřské školy Brno, Merhautova 37 za podporu a tamnímu pedagogickému sboru za vstřícný přístup.

Obsah

Kapitola 1

Úvod

Moderní technologie jsou dnes využívány ve všech odvětvích lidské činnosti. Vyjímkou není ani školství, kde interaktivní tabule a jiné nástroje nabírají stále větší význam. Přesto je převážná většina záznamů ve školství vedena pouze v papírové formě. Toto je zčásti způsobeno chybějícími nebo zastaralými legislativními předpisy, ale i v současné době představuje vhodný systém pro správu a vyhodnocování informací nezanedbatelnou výhodu, minimálně pro samotné učitele - rutinní činnosti, jako je například výpočet výsledných známek, tvoří značnou část jejich pracovní náplně.

Cílem této práce je vytvořit takový systém na platformě Android, podle [4] v současné době nejrozšířenější mobilní platformě vůbec. Má představovat efektivní nástroj pro udržení komplexního přehledu o žácích a třídách základních a středních škol a usnadnit tak učitelům jejich každodenní práci.

Kapitola 2

Android

Android je open source platforma známá především z oblasti mobilních zařízení (chytré telefony, tablety, navigace), ačkoli se objevují návrhy jeho použití i v jiných oblastech, například chytré domácí spořebiče [7]. Kromě samotného operačního systému zahrnuje také middleware[1] a základní aplikace. Základními aspekty jsou nízké nároky na hardware a přenositelnost.

2.1 Historie

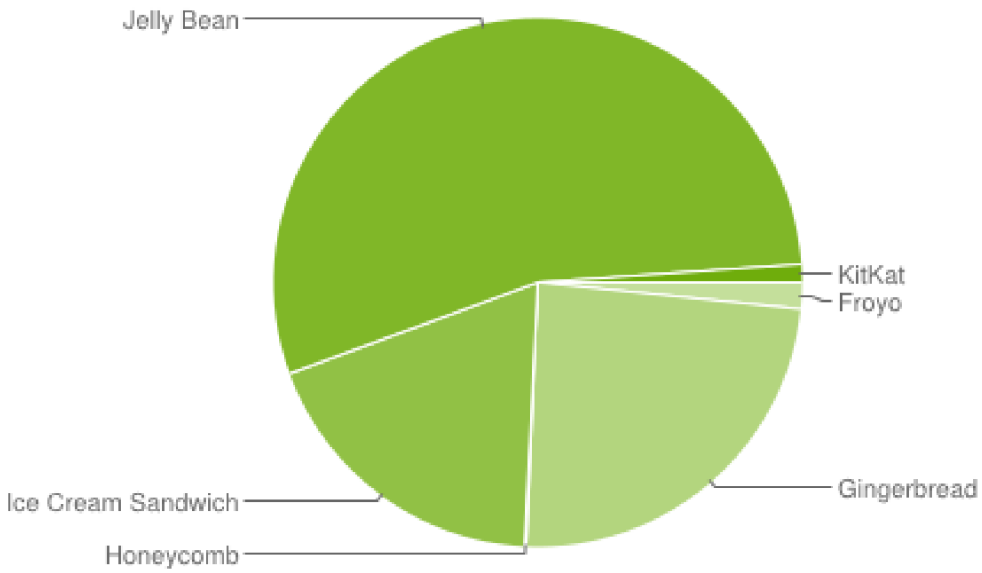
Obsah této kapitoly je čerpán především z [9] a [6]. Nepříliš známá společnost Android Inc. byla založena v roce 2003 a následně 2005 odkoupena společností Google Inc. Tým Google pod vedením Andyho Rubina, jednoho ze zakladatelů původní Android Inc., vyvinul platformu založenou na linuxovém jádře a v září 2007 získal několik patentů na poli mobilních technologií. V listopadu téhož roku bylo založeno konsorcium *Open Handset Alliance* (OHA), složené výrobců mobilních telefonů, telekomunikačních operátorů a technologických firem pod záštitou Google, které se zasloužilo o další vývoj. Dnes zahrnuje více než 80 firem (původně 34). První mobilní telefon s operačním systémem Android byl HTC Dream, představený v roce 2008 s nepojmenovanou verzí Android 1.0.

2.2 Verze

Počínaje 1.5 jsou významné verze pojmenovány v abecedním pořadí podle oblíbených amerických zákusků. Každá aktualizace přináší nové API[2] rozšiřující funkcionalitu předešlého. Tento fakt přináší nutnost hledání kompromisu mezi lepší funkcionalitou a nejnovějšími trendy v programování na jedné straně a počtem přístrojů, na kterých bude možné aplikaci spustit, na straně druhé. V současné době se jako nejvýhodnější jeví podporovat verze od 2.3.3 Gingerbread výše, čímž je pokryto 98,3 % trhu podle [10] Proti podpoře nižších verzí mluví, kromě jejich nízkého zastoupení, i často hlášené problémy se stabilitou databázové knihovny SQLite. Další výrazněji zastoupená verze je až 4.0.3 Ice Cream Sandwitch s API verze 15, tedy 5 verzí od 2.3.3 Gingerbread, což by mohlo znamenat značný rozdí ve funkcích, samotná verze Gingerbread však tvoří přes 24 % trhu. Dalším faktem je že těchto 24 % představují především starší a levnější zařízení a lze předpokládat, že státní instituce

[1] Softwarová vrstva, která zprostředkovává funkce operačního systému aplikacím

[2] API - Application Programming Interface - protokol komunikace mezi softwarovými komponentami



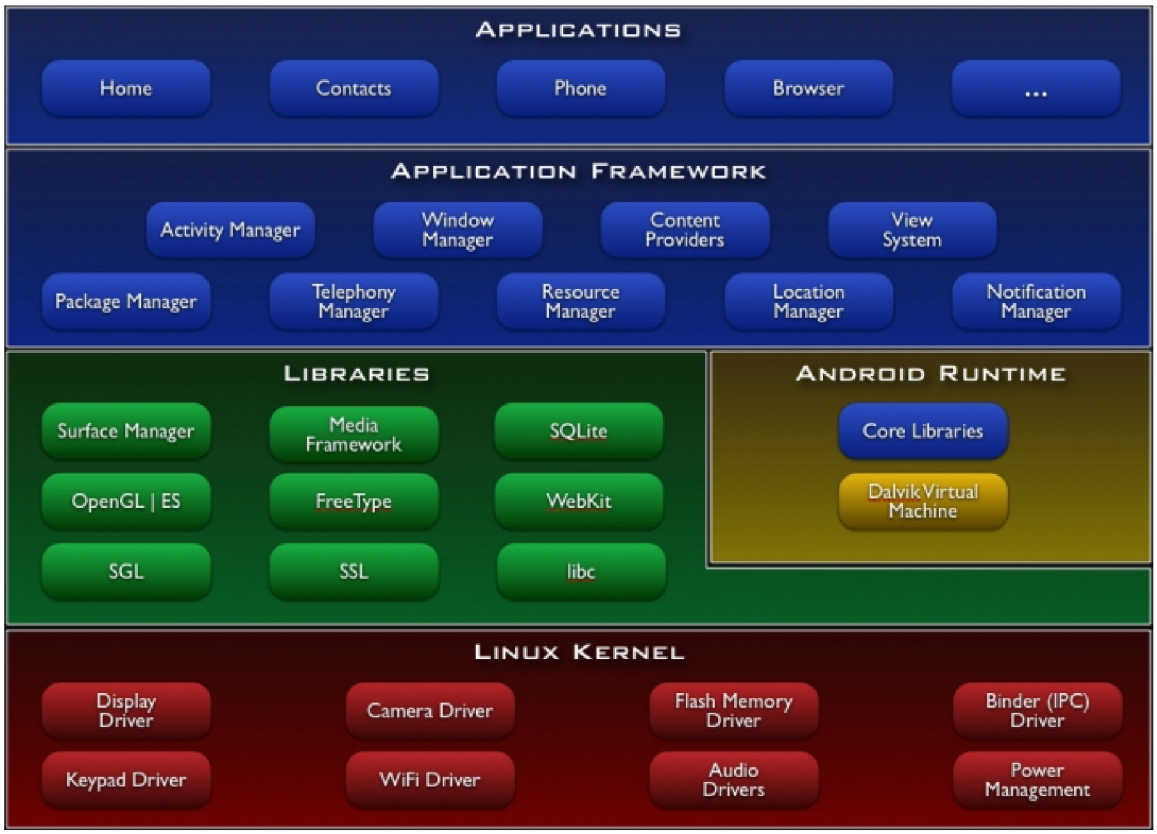
Obrázek 2.1: Zastoupení verzí Androidu k 2. 12. 2013 podle [10]

s omezeným rozpočtem, jakou škola je, by v případě nákupu více kusů dala přednost právě těmto modelům.

2.3 Architektura

Architektura systému Android je rozdělena do pěti vrstev zobrazených na obrázku 2.2.

- **Aplikace** Nejvyšší vrstva architektury, kterou tvoří aplikace používané uživateli. Implicitně obsahuje například telefonní seznam, přehrávač médií, internetový prohlížeč atd., další aplikace je možné doinstalovat přes obchod Google Play.
- **Aplikační framework** Vrstva architektury, která zprostředkovává aplikacím přístup k hardwarovým zdrojům zařízení. Cílem je jednoduché znovupoužití existujících komponent aniž by byla ohrožena bezpečnost jednotlivých aplikací. Mimo jiné zahrnuje:
  - **poskytovatele obsahu:** prostředek sdílení dat mezi aplikacemi,
  - **správce zdrojů:** přiděluje zdroje jednotlivým aplikacím na základě jejich požadavků a reálných možností systému a
  - **správce aktivit:** řídí životní cyklus aktivity a přepínání mezi nimi.
- **knihovny** Systém Android poskytuje řadu knihoven, přístupných skrze aplikační framework. Zastoupeny jsou, mimo jiné:
  - **SQLite:** odlehčený databázový systém
  - **WebKit:** knihovna pro práci s internetem
  - **OpenGL ES:** platformně nezávislá 3D grafická knihovna
  - **SGL:** knihovna základního 2D zobrazení



Obrázek 2.2: Vrstvy architektury systému Android podle [10]

  - **Systémová knihovna jazyka C:** implementace standardní systémové knihovny jazyka C vyladěná pro vestavěná zařízení
  - **Knihovny médií:** knihovny pro pořizování a přehrávání audiovizuálních záznamů, včetně obrázků

- **Běhové prostředí** Vrstva obsluhující běh aplikací. Každá aplikace běží ve svém vlastním procesu a s vlastní instancí virtuálního stroje *Dalvik*. *Dalvik* nahrazuje klasický JVM[3] známý z PC a je jednou z nejvýznamnějších částí systému Android. Oproti JVM, které jsou založeny na zásobníkové architektuře, používá Dalvik registrovou architekturu a je optimalizován pro minimální paměťovou náročnost. Spouští soubory ve formátu .dex, což jsou zdrojové kódy jazyka Java zkompilované příslušným nástrojem do 16tibytového instrukčního kódu. Samotný virtuální stroj má přístup k systémovému jádru za účelem správy běžících vláken a nízkoúrovňového přístupu do paměti.

- **Linuxové jádro** Nejnižší softwarová vrstva založená na monolitickém linuxovém jádru verze 2.6. Obstarává základní systémové služby jako je správa paměti, správa procesů, síťové prostředky a práce s ovladači hardware. Tato vrstva vytváří abstrakci hardware pro zbytek systému a uživatelská aplikace by k ní neměla získat přímý přístup.

[3] Java virtual machine, virtuální stroj Javy

2.4 Vývoj aplikací

Tato kapitola, založená především na [1] a [10], pokrývá teoretické základy nutné pro programování aplikací pro operační systém Android.

2.4.1 Přístupy k vývoji

Jako každý současný mobilní operační systém, Android umožňuje volbu mezi třemi základními přístupy k tvorbě aplikací. Na zvoleném přístupu záleží konkrétní nároky aplikace na zařízení i jeho okolí a není možné univerzálně určit který je nejlepší, protože váha výhod a nevýhod každého je závislá na určení dané aplikace.

- **Webový vývoj:** Umožňuje velmi rychlý a snadný vývoj a nasazení aplikace. Webové aplikace navíc bývají nezávislé na platformě, na které jsou spouštěny, takže umožňují vysokou přenositelnost. Problémem je, že není možné je distribuovat přes oficiální kanály jako je *Google Play* a mají vvýrazně omezený přístup k funkcím zařízení. Dalším negativem je nutnost připojení k internetu, které je dnes již rozšířené, ale stále není možné jej považovat za samozřejmost.

- **Nativní vývoj:** Klasický přístup k tvorbě aplikací. Aplikace je spuštěna lokálně v nejvyšší vrstvě operačního systému, může mít (v případě Androidu zprostředkovaný) přístup ke všem funkcím systému a je možné ji distribuovat skrze oficiální distribuční kanály. Nevýhodou je omezený výkon a zdroje současných přenosných zařízení, což limituje možnosti spouštěných aplikací. Aplikace také nejsou přenosné, což zvláště v případě společností se snahou o co největší pokrytí trhu znamená zvyšování nákladů na vývoj. Největší (a pro tento projekt zásadní) výhodou je, že aplikace není závislá na žádném vybavení dodávaném třetí stranou, čímž se omezuje možnost neovlivnitelné chyby.

- **Hybridní vývoj:** Snaží se kombinovat výhody obou zbývajících přístupů. Využívá nástroj označovaný jako *wrapper* nebo *zapouzdřovač*, mezi nejznámější patří například IBM Worklight [8]. Tento nástroj umožňuje spustit aplikaci vytvořenou pomocí HTML5, CSS a JavaScript jako nativní. Oproti klasickému nativnímu vývoji tento přístup umožňuje práci s jazyky vyšší úrovně. Aplikace sama je přenositelná mezi různými operačními systémy, zapouzdřovač však není, proto zvláště při přechodu na novou verzi API (viz. 2.2) vznikají značné komplikace. Další problém může představovat závislost rychlosti aplikace na výkonu webového nebo JavaScript jádra. Takto vytvořené aplikace je možné distribuovat pomocí *Google Play*

Pokud by bylo možné předpokládat stoprocentní pokrytí přístupnou sítí internet, jevil by se webový vývoj jako optimální - výpočetně náročné úkony by byly prováděny na vzdáleném serveru a přenosné zařízení by mohlo sloužit pouze jako terminál. Tento předpoklad bohužel není splněn, protože pokrytí sítí WiFi dosud není na školách standardem a parametry mobilního připojení bývají příliš závislé na telefonním operátorovi, ať už se jedná o rychlost nebo cenu. Hybridní vývoj by byl pravděpodobně rychlejší, avšak za cenu zpomalení výsledné aplikace, což u už tak poměrně nevýkonných přenosných zařízení představuje zásadní problém. Pro další práci byl proto zvolen nativní vývoj.

Android je otevřená platforma, u které počet a kvalita aplikací záleží na ochotě společností a programátorů psát pro ni. Aby tuto ochotu podpořil, poskytuje Google soubor nástrojů známý jako *Android SDK*, což je zkratka *Android Software Developement Kit*.

Jedná se o balíček umožňující vývoj a testování aplikací určených pro Android na operačních systémech běžných pro PC, tedy bez nutnosti fyzického přístupu k zařízení s každou verzí API kterou má výsledná aplikace podporovat. Největší podpora je poskytována pro vývojové prostředí *Eclipse* v podobě zásuvného modulu *Android Development Tools*, podobné moduly však vznikají i pro ostatní vývojová prostředí. Základním jazykem, ve kterém jsou psány nativní aplikace pro Android, je Java, ačkoli za pomoci *Android NDK*[4] je možné psát aplikace i v C/C++.

2.4.2 Struktura aplikace

Z hlediska zdrojových souborů je každý projekt uspořádán do specifické stromové struktury adresářů, mezi jejíž základní prvky patří:

- **AndroidManifest.xml:** popisuje aplikaci a její komponenty,
- **build.xml:** skript pro kompilaci a instalaci aplikace,
- **assets/:** statické soubory nutné pro použití aplikace (např. fonty, ...),
- **bin/:** složka pro uchování binárních souborů zkompilované aplikace,
- **gen/:** složka pro uchování souborů vygenerovaných překladačem,
- **libs/:** složka obsahující knihovny třetích stran,
- **src/:** složka se zdrojovými kódy aplikace,
- **res/:** složka pro uchování prostředků (např. ikony, návrhy grafických rozhraní, informace o barvách)

Softwarové komponenty aplikace jsou pak následující:

- **Aktivita** Základní kámen aplikace, obdoba okna na stolním počítači. Je to samostatná část řešící logiku celé aplikace. Životní cyklus aktivity je znázorněn na obrázku 2.3.

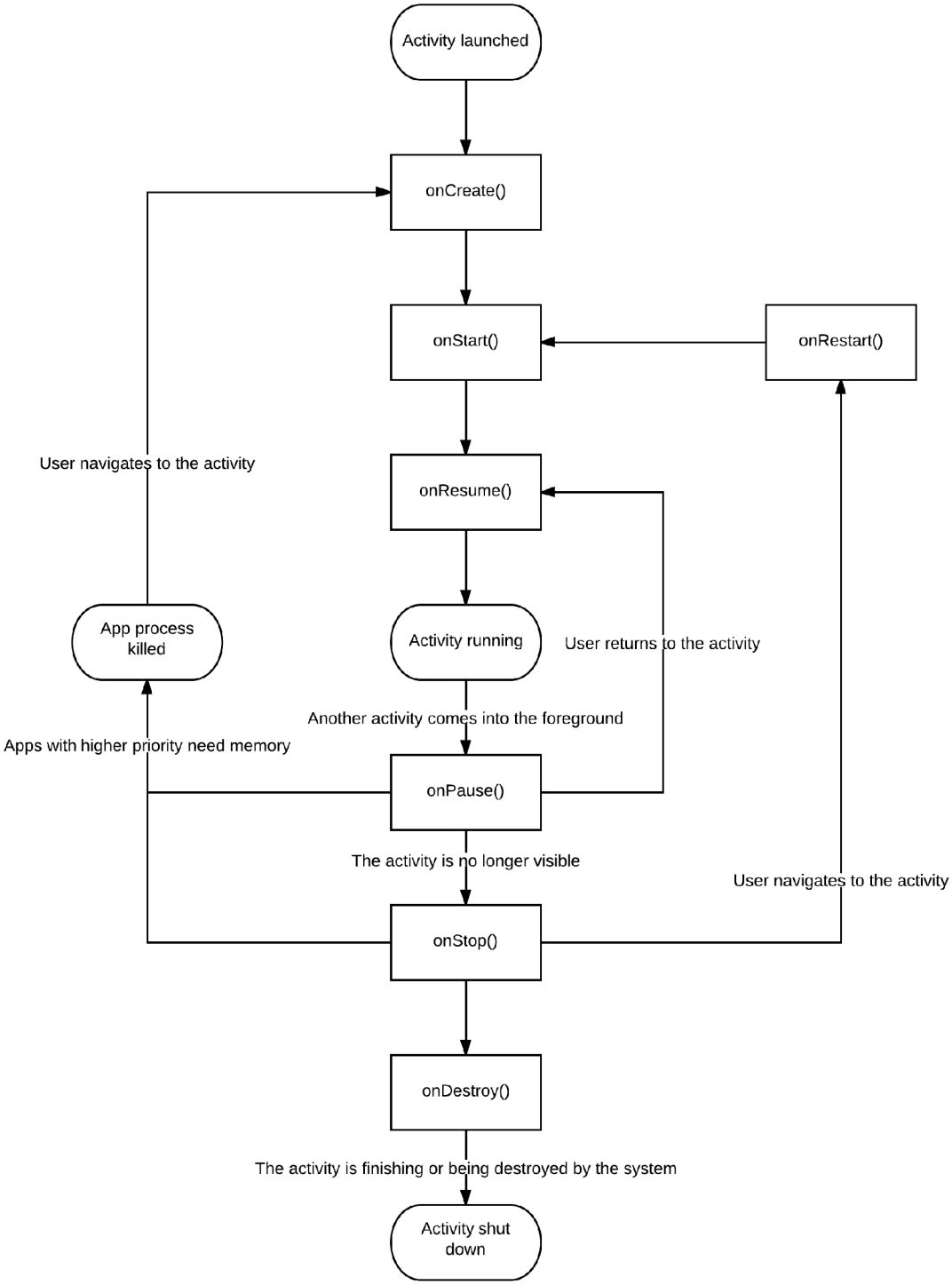
  Aktivita se vždy nachází v jednom z následujících stavů:

  - **aktivní:** Aktivita je spuštěna a na popředí, je možná plná interakce s uživatelem.
  - **pozastavená:** Aktivita je spuštěná a viditelná, na popředí se však nachází nějaké upozornění. Interakce mezi uživatelem a aktivitou není možná.
  - **zastavená:** Aktivita je spuštěná a běží, ale v popředí se nachází jiné aktivity. Jediná interakce s uživatelem probíhá pomocí upozornění.
  - **mrtvá:** Aktivita nebyla spuštěna nebo byla násilně ukončena.

- **Služby** Komponenty běžící na pozadí nezávisle na aktivitách, zajišťují běh dlouhodobých procesů. Příkladem může být přehrávání hudby, které se nezastaví při přepnutí aktivity přehrávače. Služba má vlastní životní cyklus.

- **Poskytovatel obsahu** Komponenta zajišťující přístup k datům a jejich zabezpečení, aniž by daná aktivita měla přímý přístup k těmto datům.

[4] Android Native Development Kit, více na `http://developer.android.com/tools/sdk/ndk/index.html`



Obrázek 2.3: Životní cyklus aktivity podle [10]

2.4.3 Uložení dat

Android podporuje tři formy perzistentního uložení dat - lokální v paměti zařízení, lokální na vyjímatelné kartě a vzdálené.

Vzdálené uložení vyžaduje přístup ke vzdálenému serveru, ať už se jedná o server specifický pro danou aplikaci nebo obecný, tzv. *cloud*[5]. Vzhledem k faktu, že vytvářená aplikace je koncipovaná pro použití offline, je z tohoto důvodu nevhodné.

Vyjímatelná karta, obvykle formátu *MicroSD*, poskytuje na poměry mobilních zařízení velké množství prostoru za cenu jednorázového nákladu na její zakoupení. Objevují se však dva hlavní problémy. V současné době méně závažný, do budoucna však potenciálně zničující, je odklon Google od používání vyjímatelných karet ve prospěch cloudových úložišť (v [3] je tento problém zdůrazněn v kontextu her, pro aplikace využívající databáze však platí stejně - omezený prostor vnitřní paměti se v posledních letech nezvětšuje, množství aplikací které jej využívají však ano). V současné době a v kontextu vyvíjené aplikace závažnější je pak problém bezpečnosti. Soubory na kartě jsou obecně přístupné všem programům na zařízení, a v případě vyjmutí karty také jakýmkoli jiným. Tímto způsobem uložená databáze tedy buď musí být šifrovaná, což zpomaluje její odezvu a klade další nároky na zařízení, nebo nesmí obsahovat žádné citlivé údaje.

Lokální paměť zařízení je pro aplikaci snadno přístupná a vždy přítomná, zároveň však velmi omezená co do rozsahu. V případě použití databáze *SQLite* navíc systém Android poskytuje základní zabezpečení databáze, kdy jediné způsoby jak zobrazit data mimo příslušnou aplikaci je pomocí tzv. *rootu* zařízení[6], což je zákrok oficiálně nepodporovaný, nebo pomocí specializovaných nástrojů na PC. Z těchto důvodů byla vnitřní paměť zvolena jako primární úložiště aplikace.

[5] více na `http://en.wikipedia.org/wiki/Cloud_computing`

[6] více na `http://www.androidmarket.cz/android/co-je-to-root-a-jak-jej-ziskat/`

Kapitola 3

Požadavky na systém

Pro úspěšný návrh systému je v první řadě nutné znát požadavky na jeho funkcionalitu, tedy co zákazník vlastně očekává. Tato kapitola byla vytvořena na základě požadavků dodaných pedagogickým sborem pro první stupeň Základní školy a Mateřské školy Brno, Merhautova 37. Požadavky ostatních škol a učitelů se mohou lišit, neměly by však být zásadně odlišné. Řešení drobných odchylek je pak popsáno v kapitole 4.

Obecně lze očekávané funkce systému rozdělit na funkce související se známkováním žáků, shrnuté v sekci 3.1, absencí v sekci 3.2 a informacemi o žákovi v sekci 3.3.

3.1 Známkování

Jedním ze základních kamenů aplikace by mělo být známkování žáků a možnosti automatického zpracování statistik. Zvláště počítáním průměrů tráví učitelé ke konci pololetí značnou část pracovní doby i volného času a všem by nějaká forma automatizace pomohla, jindy by bylo vhodné srovnání aktuálních známek konkrétního žáka s třídním průměrem - například při jednání s rodiči na třídní schůzce. Základní požadavky na funkce známkování byly shrnuty jako:

- **Datum:** Každá známka musí mít uvedené datum udělení z důvodu zpětné kontroly.
- **Kategorie:** Každá známka musí patřit do nějaké kategorie, například „pololetní písemná práce“ nebo „zkoušení“. Názor, zda mají různé kategorie mít různou váhu, případně jakou konkrétně, je nejednotný, proto by váha měla jít nastavit individuálně pro každou kategorii podle potřeb konkrétního učitele. Stejně tak kategorie v tělesné výchově budou jiné než v českém jazyce, musí proto být možné přidávat a ubírat kategorie v jednotlivých předmětech.
- **Popis:** Kromě kategorie (například „ústní zkoušení“) musí mít známka i konkrétní popis za co byla udělena (např. „Přemyslovci a Slavníkovci“). Tento popis by měl být relativně krátký, ale nelze očekávat konkrétní formát zápisu - i počet slov se různí učitel od učitele.
- **Průměry:** V každém předmětu musí být možné spočítat průměr v daném pololetí pro konkrétního žáka. Tento průměr musí, v případě že jsou různé, reflektovat váhu jednotlivých známek. Dále by bylo vhodné moci zobrazit i aktuální průměr třídy.
- **Návaznost:** V každém předmětu by měla být možnost zobrazit známku z předchozího klasifikačního období, i v případě že se technicky jedná o jiný předmět (např. přechod

Vlastivěda - Dějepis). Tato známka slouží pouze pro orientaci a nesmí mít vliv na výpočty výkonu žáka v novém období.

Kromě vypsaných položek musí systém umět pracovat i s položkami samozřejmými, jako je dělení známek podle předmětů a tříd, jejich výpis a podobně.

3.2 Absence

Druhý základní kámen aplikace. Stejně jako u známkování, i u absencí musí učitelé vytvářet statistiky převážně ručně.

U absencí nejsou tak markantní rozdíly mezi požadavky jednotlivých učitelů jako u známkování, o to větší rozdíly však jsou mezi stupni školní docházky. Na prvním stupni základní školy platí, že pokud je určitá hodina v rozvrhu označená jako „Český jazyk“, znamená to, že je započítána do absence v českém jazyce, i v případě že by například učitel onemocněl a byla suplována matematika. Toto však již neplatí na druhém stupni, kde jsou hodiny započítány podle reálně odučených předmětů, navíc je možné že hodina tzv. „odpadne“, tedy není odučena vůbec. Důsledky této nekonzistence jsou rozebrány v sekci 4.3.

Celkově se u absence sleduje méně znaků než u známek. Pro konkrétního žáka je nutné sledovat pouze ve které hodině byl v daném stavu a jaký stav to byl. Žák může být pouze přítomen, nepřítomen bez omluvy nebo nepřítomen s omluvou, jiné hodnoty nejsou přípustné. Oproti známkování zde lze také předpokládat vyšší množství operací změny nastavené hodnoty, protože rodiče často žáky omlouvají zpětně - učitel tedy žákův status „neomluven“, zapsaný v průběhu hodiny kdy chyběl, musí změnit na „omluven“ když přinese omluvenku. Pro účely statistik je zde sledován podíl hodin v nichž žák chyběl. Pokud tento dosáhne limitu, nemůže žák být z daného předmětu klasifikován. Tento limit se liší opět v závislosti na škole a stupni vzdělání žáka, musí proto jít nastavit individuálně. Systém by také měl zobrazovat upozornění u žáků kteří se k tomuto limitu blíží.

V této části je nutné implementovat vhodné rozhraní pro zadávání a pozdější změny indikátorů přítomnosti, a to jak po jedné hodině tak po větších časových úsecích. Absence kratší než jedna hodina bývají řešeny individuálně podle konkrétního trvání - položky „zaspal“ a „pozdní příchod - 10 minut“ bývají řešeny poznámkou, úseky delší než 30 minut pak bývají vyhodnoceny jako absence po dobu trvání celé hodiny. V obou případech bývají řešeny jiným způsobem, není proto nutné implementovat podporu pro „krátké“ absence. Ostatně jejich omlouvání by v praxi také bylo velmi složité, protože jednotlivé úseky by na sebe nenavazovaly, tudíž by pro každý byla nutná zvláštní omluvenka, což by prostor v žákovské knížce zaplnilo velmi rychle.

3.3 Informace o žákovi

Do této sekce spadají zbývající informace, které nemají vazbu na absenci a známky. Jedná se především o kontaktní údaje, historii žáka a kázeňská opatření. V této sekci lze předpokládat menší množství prováděných operací, změna bydliště každý týden není příliš pravděpodobná, jedná se však převážně o informace důvěrné, je proto vhodné je nějakým způsobem zabezpečit. Sledované informace jsou:

- **Osobní údaje:** jméno, příjmení, datum narození, místo, rodné číslo
- **Kontaktní údaje matky:** jméno, příjmení, zaměstnání, telefon, email

- **Kontaktní údaje otce:** jméno, příjmení, zaměstnání, telefon, email
- **Přihlášen do školy:** datum, případná předchozí škola
- **Odhlášen ze školy:** datum, případná nástupná škola
- **Lékař:** jméno, telefon
- **Trvalé obtíže:** popis případných trvalých zdravotních indispozic

Dále do této sekce patří kázeňská opatření, od poznámek po zhoršené známky z chování, a poznámky učitele (například „do pátku musí dodat sešit s domácími úkoly“). Tady se jedná o upozornění bez vazby na konkrétní předmět, sledované hodnoty jsou pouze datum a textový popis.

Kapitola 4

Návrh

Tato kapitola se zabývá konkrétním návrhem řešení problémů nastíněných v kapitole 3 a situacemi vzniklými z důvodu různorodosti požadavků. I tuto kapitolu lze rozdělit do sekcí absence 4.3, známkování 4.4 a informace o žácích 4.5, oproti předchozí zde však přibývá sekce zabývající se vazbami v systému a existujícími systémy s podobným určením. Schéma navrhované databáze je zobrazeno na obrázku 4.5. Vzhledem k různorodosti požadavků jednotlivých škol a učitelů je využití většiny tabulek volitelné.

4.1 Existující systémy

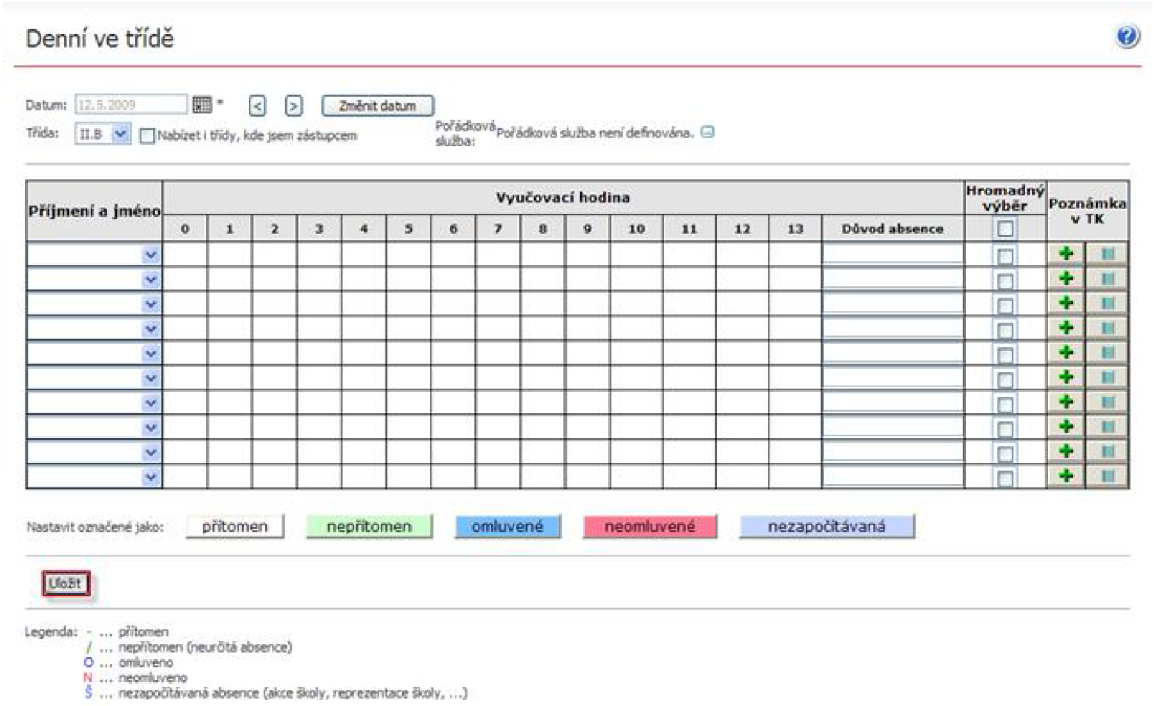
Informační systémy určené pro učitele na základních a středních školách jsou méně rozšířené než jejich vysokoškolské alernativy, přesto však existují. Mgr. Josef Basl v článku [2] uvádí dva hlavní zástupce, a to Škola OnLine[1] a systém Bakaláři[2]. V obou případech se jedná o rozsáhlé systémy umožňující spravovat většinu administrativy školy. Existují samozřejmě i další varianty, například systém společnosti *Matcomp s.r.o*[3], jejich principy jsou však podobné. Tyto systémy jsou vhodné spíše pro inspiraci než jako konkrétní vzory z důvodu odlišnosti platformy pro kterou jsou určeny, základní schéma je však možné převzít pouze s drobnými úpravami.

V případě Škola OnLine se jedná o webový informační systém, je tedy teoreticky možné jej použít i prostřednictvím přenosných zařízení. Systém však pro takový přístup není určen a je optimalizován pro použití s pomocí tradiční kombinace myši a klávesnice, uživatelský vstup z dotykového displeje proto není zrovna pohodlný. Výhodou, ačkoli zároveň i potenciálním bezpečnostním rizikem, je, že se jedná o hostovanou službu, škola tedy nemusí vlastnit webový server pro provoz vzdálené databáze.

Systém Bakaláři je lokální pro konkrétní školu, je proto nutné zřídit server pro běh serverové části programu. Tímto je omezen přístup třetích osob k informacím na serveru uloženým, zároveň však vzniká nutnost udržet server v chodu, spojená s finančními a personálními náklady. Klientská část je určena pouze pro systémy Windows. Dále systém Bakaláři nabízí sadu webových aplikací s omezeným přístupem, vhodných především pro žáky a jejich rodiče.

Oba zmíněné systémy jsou určeny především jako administrativní nástroje školy, nikoli učitele. Tomu odpovídá důraz kladený na tvorbu rozvrhů a suplování, evidenci vý-

[1] `http://portal.skolaonline.cz/Home.aspx`
[2] `http://www.bakalari.cz/`
[3] tthttp://www.matesova.cz/informacni-systemy_is-skoly



Obrázek 4.1: Zadávání absence v systému Škola OnLine

Obrázek 4.2: Zobrazení rozvrhu v systému Škola OnLine

půjček v knihovně a podobné aktivity. Dále sledují klasickou strukturu informačního systému určeného pro PC, tedy nástrojová lišta v horní části obrazovky, panel s upozorněními a nejčastěji využívanými zkratkami v levé části a samotná pracovní plocha ve zbývajícím prostoru. Tento přístup je pro mobilní aplikace poměrně nevhodný, jelikož plocha displeje

bývá menší a velikost tlačítek nutná pro efektivní práci větší než je běžné na PC. Příkladem problematického rozvržení obrazovky může být například obrázek 4.1, na kterém je zacena obrazovka pro zadávání absencí v systému Škola OnLine. Zde použitý přístup je možné použít i v mobilní aplikaci, velikost ovládacích prvků by však musela být přizpůsobena dotykovému ovládání. Zcela nevhodný prvek je pak volba žáka (pole *Příjmení a jméno*) pomocí výběru - jedná se o prvek nadbytečný a v dotykové variantě prostorově velmi náročný. Rozvržení sloužící především k zobrazení informací, nikoli jejich zadávání, je možné převzít snáze, například na obrázku 4.2 je zobrazení rozvrhu opět ze systému Šola OnLine. Výběr data je v této části nadbytečný, samotný rozvrh je však velmi dobře použitelný. Na druhou stranu je pravda, že konkrétně v případě rozvrhu by bylo obtížné vytvořit efektivní systém s jiným rozvržením.

4.2 Celkový návrh

Osvědčený přístup u mobilních aplikací je maximalizovat samotnou pracovní plochu s tím, že nástroje jsou převážně skryty. Většina akcí se zobrazuje v kontextu s označeným prvkem, tedy například při označení textového pole se uživateli zobrazí softwarová klávesnice, zatímco při označení pole s možností výběru je zobrazeno dialogové okno s konkrétními možnostmi. Ovládací prvky nezávislé na konkrétním kontextu byly do verze 3.0 Honeycomb (viz. 2.2) sdruženy pod (často hardwarovým) tlačítkem „menu“, v novějších verzích tento přístup není doporučován. Místo tohoto tlačítka je používán tzv. *ActionBar* (viz. [5]), zobrazený na přání uživatele vyjádřené konkrétním pohybem prstu po obrazovce. Na pracovní ploše jsou zobrazeny pouze ovládací prvky nutné pro zvýšení efektiivity práce, tedy zástupci nejčastěji používaných akcí, kde by skrytí přinášelo nepřijatelné zdržení. Tento přístup se jeví jako optimální i pro aplikaci informačního systému, kde jednotlivé obrazovky jsou buď rozcestníky s tlačítky pro další směrování, nebo zobrazovací plochy se skrytým ovládáním. Je nutné najít kompromis mezi přehledností plochy a dobou nutnou k provedení konkrétních operací, v praxi tedy zvážit která tlačítka na ploše být mají a která mají být skryta, toto však je záležitost související s konkrétními obrazovkami a není možné ji vyřešit obecně.

4.3 Absence

Tato část aplikace je datově pravděpodobně nejnáročnější. V minimální verzi je nutné sledovat počet zameškaných hodin a data těchto absencí, teoreticky by tedy stačilo, společně s vhodnou konstrukcí tabulek rozvrhu, vést záznamy o chybějících žácích ve formátu `jméno:datum:hodina`. Vzhledem k rozdílným přístupům k suplování na různých stupních (viz. kapitola 3.2) zde však nastává problém s určením konkrétního předmětu, ve kterém žák chyběl. Podobný problém vzniká pokud by pole `hodnota` obsahovalo označení předmětu místo čísla hodiny, z důvodu opakování předmětů v jednom dni. Tyto ukazatele je tedy nutné sledovat oba. Hlavní nárůst datového objemu je však způsoben nutností uchovávat záznamy i o přítomných žácích. Tento údaj je nutný jednak z kontrolního hlediska, aby učitel měl přehled jestli byl daný žák přítomen nebo jen „zapomněl zapsat“, a jednak z důvodu výpočtů absencí. Po konzultaci byl zvolen přístup výpočtu podílu zameškaných hodin v procentech jako výrazu 4.1 místo výrazu 4.2. Tento přístup umožňuje snazší zpracování změn hodinových dotací předmětů z důvodů suplování, státních svátků aj. a zároveň odbourává nutnost zadávat každý rok aktuální hodinové dotace pro všechny předměty. Nevýhodou je již zmíněný nárůst databáze a tvorba „planých poplachů“ - po první odučené

	M	ČJ	TV	...
Monika Antalová	-	-	-	
Jiří Hrdlička	!	/	/	
Pavel Vomáčka	/	/	?	
...				

Obrázek 4.3: Tabulka pro zadávání absence

hodině v referenčním období budou mít chybějící žáci 100% absenci. Problém planých poplachů lze vyřešit například kontrolou jestli $pritomen + nepritomen > 5$[4], kdy v případě negativního výsledku není varování zobrazeno. Další potenciální komplikace může vzniknout pokud se součet $pritomen + nepritomen$ u jednotlivých žáků v jednom předmětu liší. Tento stav značí že učitel některému žákovi zapomněl zapsat přítomnost v hodině a systém na něj musí automaticky upozornit při uzavírání referenčního období.

$$absence = \frac{nepritomen}{pritomen + nepritomen} * 100 \tag{4.1}$$

$$absence = \frac{nepritomen}{hodinovadotace} * 100 \tag{4.2}$$

Při maximálním vytížení má tabulka absencí přibližně 62560 záznamů na školní rok, této hodnoty však bude zřídka dosaženo. Předpokládá 34 žáků ve třídě - maximum po udělení všech vyjímek (viz. [11]), 46 vyučovacích hodin - maximum na sportovním gymnáziu (viz. [11]), a všechny státní svátky a prázdniny jsou počítány jako běžné dny. V dnešní době běžný průměr je 22 žáků na třídu a maximální týdenní počet povinných vyučovacích hodin je 35-40 pro žáky gymnázií a VOŠ.

Z pohledu uživatele se jako optimální jeví kombinace přístupu známého z klasické třídní knihy a obou systémů zmíněných v 4.1, tedy přístupu přes datum, s přístupem přes kartu žáka. Přístup přes datum znamená, že se uživateli po zadání třídy a data zobrazí tabulka s absencemi jednotlivých žáků v hodinách daného dne, jako je zobrazena na obrázku 4.3. Zvolený tvar databáze, zobrazený na obrázku 4.5 umožňuje automaticky vyplnit předměty podle zadaného vzoru nebo, pokud je pro danou hodinu již zadána absence, podle předmětu pro který ji uživatel nastavil. Konzistenci databáze je v tomto případě nutné ošetřit programově aby všichni žáci měli v daný čas stejný předmět i v případě, že je předmět změněn v průběhu zadávání. Toho lze dosáhnout kontrolou a případnou úpravou hodnot v databázi při změně zadávaného předmětu. Přístup přes kartu žáka je obdobný s tou vyjímkou, že je zobrazena pouze absence daného žáka. Problém dělení třídy, například ve výuce jazyků nebo tělesné výchově, není nutné řešit, protože daný učitel vede informace pouze o žácích které reálně učí. Po doteku v polích předmětu bude zobrazena nabídka změny, dotek v průsečíku předmětu a jména umožní změnit stav žáka v dané hodině. Zde byly zvoleny symboly „-“ pro nepřítomen - omluven, „/“ pro přítomen, „!“ pro nepřítomen - neomluven a „?“ pro neznámý stav, kdy záznam pro dané pole ještě nebyl zadán. Z důvodu uživatelské přívětivosti je vhodné ve výsledné aplikaci pole s těmito symboly zvýraznit vhodnými barvami. Oproti příkladu na obrázku 4.1 je zde méně ovládacích prvků, jména jsou přednastavena systémem a hodnoty v polích nejsou nastavovány zvláštním tlačítkem.

[4]Hodnota 5 byla zvolena záměrně. Jedná se o hodnotu dostatečně nízkou aby varování neztrácelo smysl a zároveň dostatečně vysokou aby byl co nejvíce omezen počet falešných varování.

4.4 Známkování

Tabulka známek nebude zdaleka tak rozsáhlá jako tabulka absencí, protože lze předpokládat že počet známek každého žáka pro jeden předmět nepřesáhne 40. Karta předmětu musí umožnit správu, tedy přidávání, odebírání i úpravy, kategorií známek udělovaných v tomto předmětu, stejně jako udělování konkrétních známek žákům. Prakticky by po výběru předmětu měl učitel mít možnost výběru žáka, kterému známku přidělí, není však nutné zde vytvářet samostatný formulář udělení známky. Po výběru žáka by však systém měl automaticky předvyplnit kontext předmětu, ve kterém budou informace o žákovi zobrazeny.

Různorodost známkování v závislosti na učiteli odráží především používání vah jednotlivých známek, zatímco někteří zastávají názor „známka jako známka“, jiní používají propracovaný systém „malých jedniček“ a „černých puntíků“. Systém musí umožňovat oba přístupy, zároveň však nesmí obtěžovat nutností zadávat nevyužívané parametry. V nastavení systému by proto měla být možnost nastavit fixní váhu všech dále udělených známek na hodnotu 1. Učitelé, kteří by různé váhy známek chtěli použít jako prostředek motivace žáků, pak musí sami zvážit jestli ctějí udělit „malou známku“ s přímým vlivem na výsledné hodnocení a vytvořit kategorii s příslušnou vahou, nebo „bod aktivity“, sloužící pouze pro jejich orientaci (více v sekci 4.5).

4.5 Informace o žákovi

Tato sekce zahrnuje především kartu žáka a výběr žáka ze seznamu. Databázově není příliš náročná, bude však nejspíše zabírat nejvíce kódu programu. Samotný výběr žáka musí být umožněn přímo ze seznamu, případně zúženého zadáním třídy do které patří, a z grafického zobrazení rozesazení v dané hodině.

Výběr ze seznamu je klasickou prací s referenční databází a je univerzální, výběr podle pozice vyžaduje více práce ze strany uživatele. Jedná se především o fakt, že rozložení učeben je různé, nelze proto vytvořit univerzální vzor pozic jednotlivých žáků - uživatel musí vytvořit „mapu“ pro každé rozesazení pro které chce grafický výběr použít. Využití této metody výběru v neznámé třídě, například při suplování, je potenciálně problematické také proto, že „mapa třídy“ nemusí reflektovat aktuální stav, pokud si žáci vzájemně vymění místa. Existují však učitelé kteří by tuto metodu výběru ocenili, je proto vhodné ji implementovat. V této konkrétní implementaci je rozesazení reprezentováno tabulkami `rozesazeni` a `pozice` v databázi podle 4.5. Tyto tabulky přiořazují každému žákovi pozici v daném rozesazení. Pozice ve třídě reprezentují jednotlivá místa, nikoli lavice, čímž je řešen problém různě velkých lavic - lavice pro dva žáky je modelována dvěma pozicemi vedle sebe. Tento přístup může vést ke snížení čitelnosti výsledného zobrazení, je proto doporučeno modelování s prokládáním volnými pozicemi, kde mezera na průchod mezi lavicemi je znázorněna neobsazenou pozicí. V obrázku 4.4 jsou zobrazena rozesazení bez prokládání, s prokládáním a s prokládáním s asymetrickými lavicemi, čísla za jménem reprezentují hodnoty `posX` a `posY` v tabulce. Karta žáka slouží k zobrazení informací o žákovi v zadaném kontextu. Tento kontext může být manuálně nastaven nebo zděděn, například při přechodu na kartu žáka z karty předmětu. V případě že je kontext obecný, jsou zde zobrazeny základní informace o žákovi, s možností zobrazení osobních údajů. Tato možnost musí být chráněna heslem z důvodu bezpečnosti. Pokud je nastaven kontext předmětu jsou zde zobrazeny aktuální známky s možností výstupu ve formě grafu a aktivita žáka. V každém případě jsou zde zobrazeny nejnovější poznámky a případná upozornění, jako třeba přiblížení k nepřípustné hodnotě absence.

jméno11	jméno12	jméno13	jméno14	jméno15	jméno16
jméno21	jméno22	jméno23	jméno24	jméno25	jméno26
jméno31	jméno32	jméno33	jméno34	jméno35	jméno27

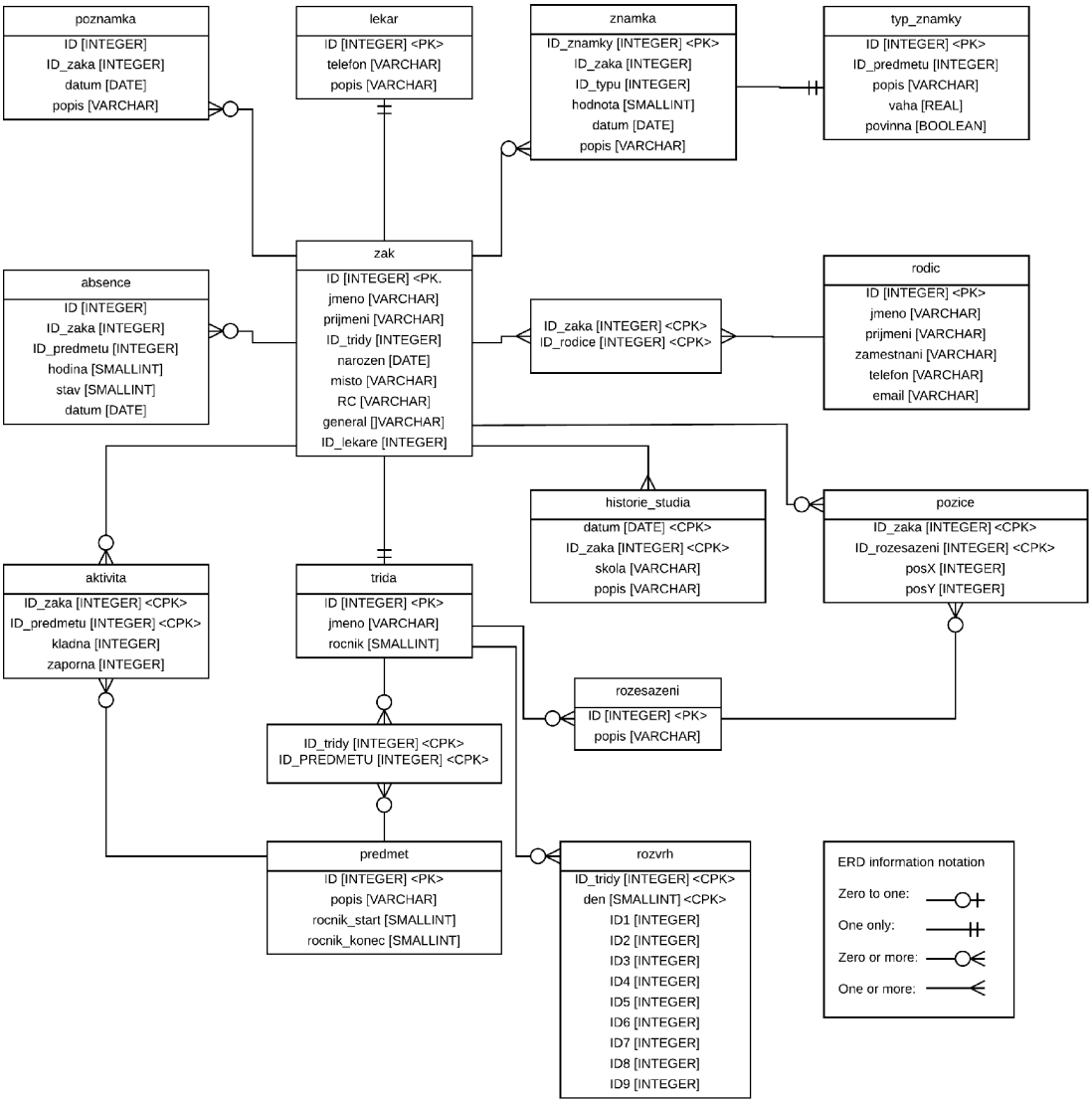
jméno11	jméno12	-	jméno14	jméno15	-	jméno17	jméno18
jméno21	jméno22	-	jméno24	jméno25	-	jméno27	jméno28
jméno31	jméno32	-	jméno34	jméno35	-	jméno37	jméno38

jméno11	jméno12	-	jméno14	-	jméno16	jméno17
jméno21	jméno22	-	jméno24	-	jméno26	jméno27
jméno31	jméno32	-	jméno34	-	jméno36	jméno37

Obrázek 4.4: Zobrazení rozesazení

Aktivita žáka je alternativní prostředek pro zaznamenávání jeho aktivity v hodinách. Je uchovávána v tabulce `aktivita` a nemá přímý vliv na hodnocení žáka. Uchovává se pouze aktuální počet kladných a záporných bodů, je proto možné jich přidělit velké množství bez zvýšení datového objemu databáze. Doporučené využití je jako klasický „černý puntík“ například za zapomenutý sešit.

Ochrana heslem byla předmětem konzultace s pedagogy a byla shledána dostačující. Cílem je odradit nenechavé žáky, nikoli profesionální zloděje dat. Co se týče přímého přístupu k datům databáze je zcela neefektivní. Tento přístup lze realizovat pomocí *rootu* zařízení nebo nástroje *Android Debug Bridge*, v obou případech však vyžaduje pokročilé technické znalosti a dlouhodobý přístup k zařízení. Majitelé zařízení, na kterých byl *root* proveden, by však měli zvážit možnost dodatečného šifrování pokud chtějí ve svém zařízení uchovávat citlivá data.



Obrázek 4.5: Původní, univerzální návrh databáze

Kapitola 5

Implementace

Struktura aplikace pro systém Android je poměrně specifická. Na obecné úrovni lze říci, že platí rovnost „jedna obrazovka - jedna třída“, kde třídy rozšiřují základní třídu *Activity*. V tomto systému byla zavedena třída *FirstScreen*, která plní funkci úvodní obrazovky a zároveň implementuje funkce využívané větším množstvím ostatních tříd. Zbývající třídy rozšiřují právě ji, čímž je zajištěn přístup k příslušným funkcím, efektivita jejich údržby i implementace základních společných prvků vyžadovaných systémem Android. Další sdílený objekt je třída *DataBlock* vytvořená podle návrhového vzoru Singleton[1], obsahující sdílená data aplikace specifická pro konkrétní běh.

5.1 Zpětná kompatibilita

Podle návrhu je systém možné spustit na Androidu verze 2.3.3 Gingerbread (API verze 10) a vyšší. Tato verze nepodporuje některé dnes standardní komponenty a využití jejich ekvivalentů není považováno za vhodné, příkladem může být *Action bar* a starší *Menu*. Tento rozdíl byl zčásti odstraněn použitím knihovny podpůrné knihovny appcompat v7, standardního nástroje podle [10].

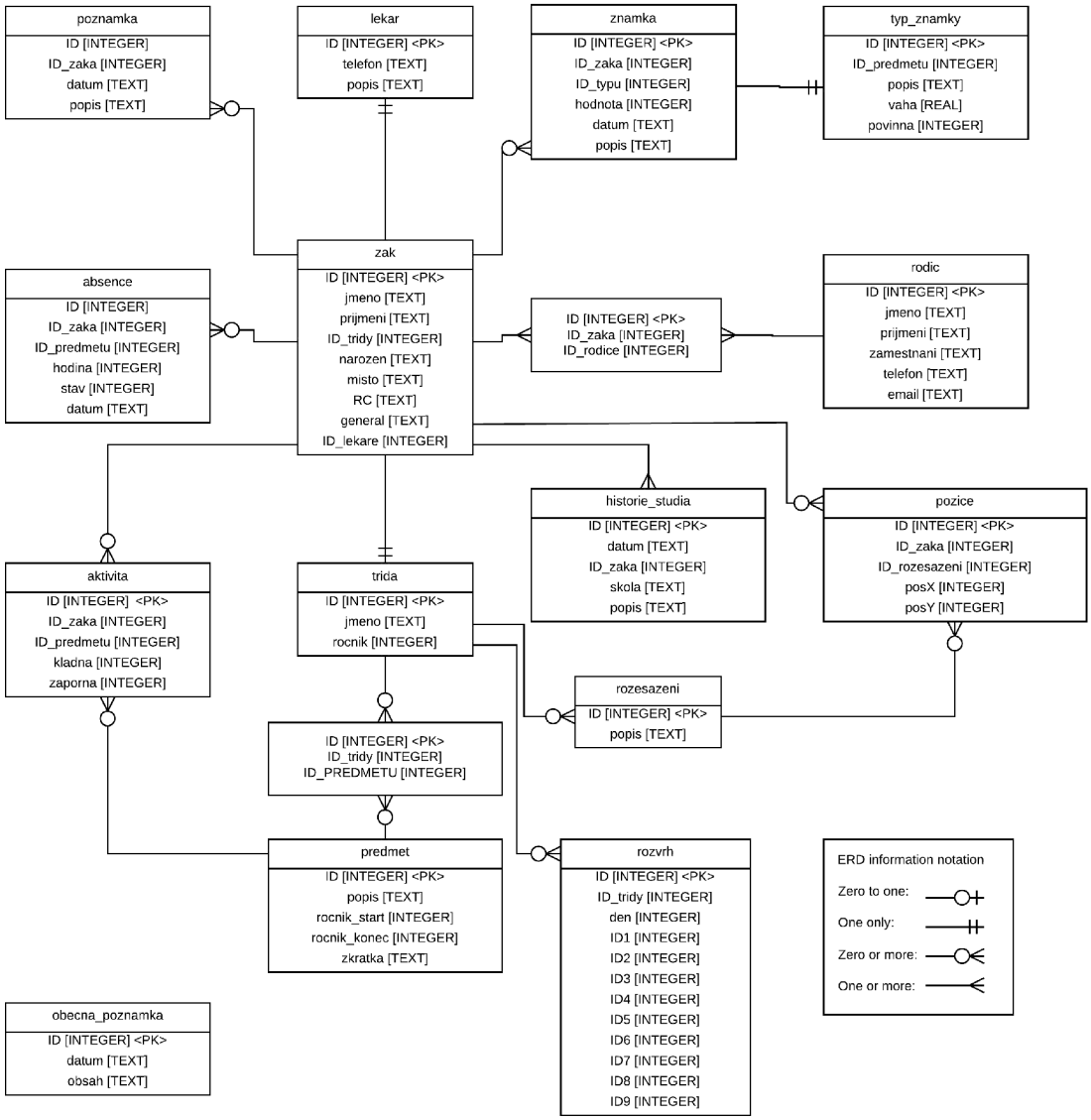
Tato knihovna poskytuje implementaci velkého množství funkcí moderních prvků, jejich vzhled však odpovídá staršímu standardu a je, z dnešního pohledu, zastaralý. Některé prvky pak nejsou implementovány vůbec, v těchto případech pak byly použity méně dekorativní ale funkční ekvivalenty. Příkladem je prvek *Wheel*, který je sice součástí složeného prvku *DatePicker* implementovaného od API verze 1, samostatně je však k dipozici až ve verzi 16 a v podpůrné knihovně obsažen není, musel proto být nahrazen prvkem *Spinner*.

5.2 Databáze

Původní návrh byl určen pro obecnou databázi a jako takový je pro zbytečně podrobný. Databáze *SQLite* veškerá data ukládá v podobě textových řetězců a velikost jednotlivých datových typů tak není pevně stanovena - odpadá tedy dělení na *smallint*, *bigint* a podobně. Prakticky byly využity typy *integer*, *real* a *text*, sloužící aplikaci k omezení zadávání neplatných hodnot.

Další nutnou změnou bylo zavedení samostatných primárních klíčů ve všech tabulkách databáze. Databáze *SQLite* sice umožňuje použití složených primárních klíčů, její imple-

[1] Česky jedináček nebo unikát, zaručuje že daná třída bude mít v programu jedinou instanci a poskytuje k ní globální přístupový bod.



Obrázek 5.1: Návrh databáze použitý ve výsledném projektu

mentace však v takové tabulce neumožňuje práci s více než jednou položkou - dotaz „vypiš všechny předměty asociované s třídou X“ by v databázi podle schématu 4.5 nebylo možné zpracovat. Výsledné schéma je na obrázku 5.1.

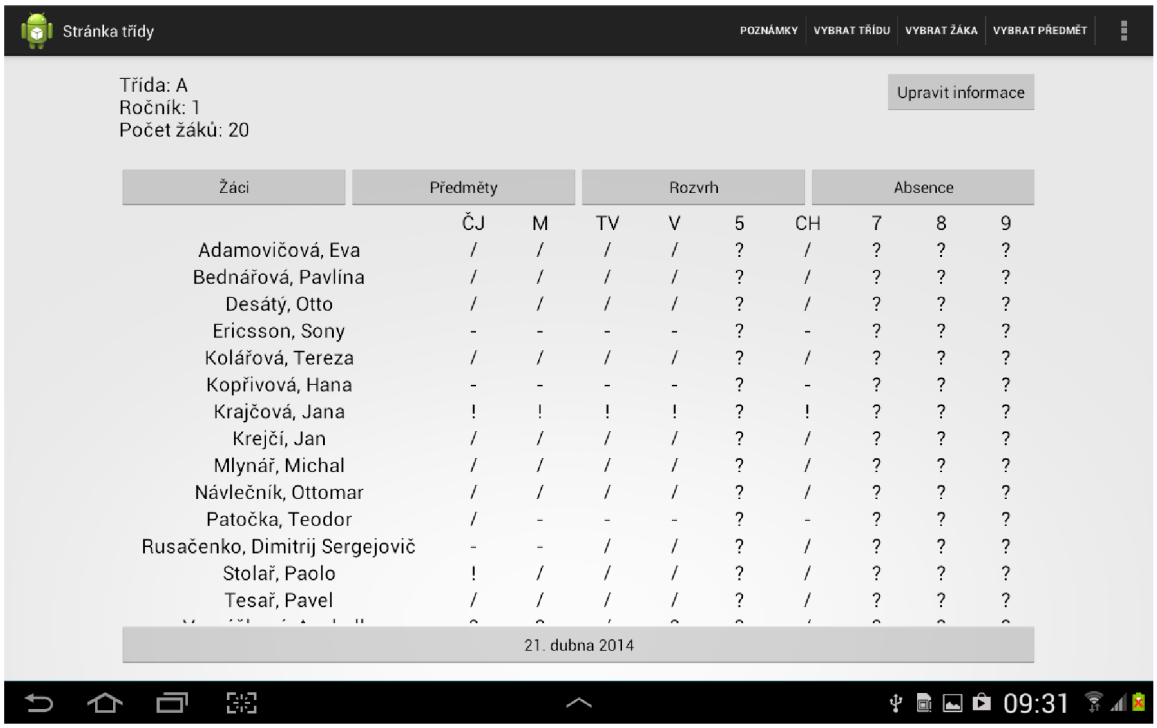
Jak bylo zmíněno v sekci 4.3, systém počítá s pravidelnou kontrolou konzistence databáze. Tato kontrola, zvláště ke konci klasifikačního období, může být časově relativně náročná, není tedy vhodné provádět ji příliš často. Na druhou stranu provést ji pouze jednou na konci klasifikačního období je také nežádoucí vzhledem k faktu, že některé hodnoty v systému by do té doby mohly být chybné. Proto byla v nastavení vytvořena možnost určit četnost těchto kontrol. V základu je nastavena kontrola jednou denně, která je zároveň nejmenším možným intervalem. Shora je interval omezen pouze maximální hodnotou typu Integer, předpokládá se však že v takovém případě bude zvolena možnost spouštět

kontroly pouze manuálně. Automatická kontrola probíhá v rámci volání funkce *onCreate()* třídy *FirstScreen*, která je základem všech tříd v systému - prakticky tedy kontrola proběhne při prvním volání funkce *OnCreate()* (spuštění, přechod mezi stránkami systému) v daném dni. Tento přístup zajišťuje provedení kontroly i v případě, že aplikace nebude přes noc vypnuta, samotný dotaz jestli už se má kontrola provést je pak tak krátký, že na výkon aplikace nemá zásadní vliv. Kontrolu je možné spustit i manuálně pomocí příslušné položky v nabídce *Správa obsahu*.

V případě že kontrola konzistence databáze odhalí nesrovnalosti, je v nabídce zobrazena další položka, vedoucí na stránku s výpisem chyb. V některých případech (například nesrovnalost v předmětu, kdy některý žák má v absenci pro konkrétní hodinu uveden jiný předmět než zbytek třídy) je navržena možnost automatické opravy, uživatel však musí tuto volbu potvrdit.

5.3 Uživatelské rozhraní

Pro organizaci prvků uživatelského rozhraní byla použita kombinace schémat *RelativeLayout* a *LinearLayout*. Teoreticky by bylo možné využít pouze schéma *RelativeLayout*, tento přístup by však vedl k příliš nepřehlednému zdrojovému kódu. Výhradní použití schámatu *LinearLayout* je běžnější, zejména ve starších aplikacích, v současné době se však od jeho využití ustupuje z důvodu vyšší náročnosti aplikace - xml strom prvků obrazovky je příliš rozvětvený, jeho vykreslování trvá déle a vzrůstá i paměťová náročnost aplikace.



Obrázek 5.2: Stránka třídy v módu Absence na zařízení s úhlopříčkou 10.1 palce

Na obrázku 5.2 je zachycena obrazovka zadávání absence pro fiktivní třídu 1.A. Základní schéma je shodné pro stránky třídy, žáka i předmětu, liší se převážně rozsah jednotlivých

Obrázek 5.3: Stránka třídy v módu Absence na zařízení s úhlopříčkou 4 palce

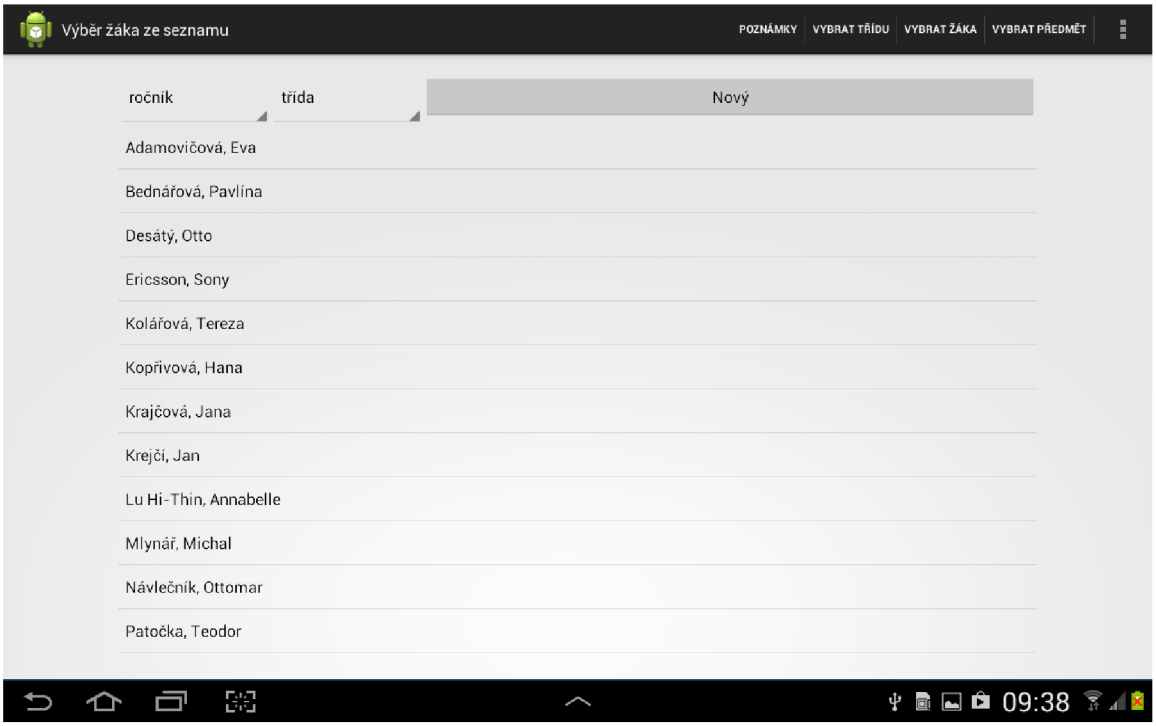
polí.

Obrazovka je z hlediska programu rozdělena do dvou základních bloků. V prvním bloku se nachází základní informace o daném subjektu, slouží především k rychlé orientaci uživatele například po přerušení práce s aplikací. Tlačítko na pravé straně slouží k modifikaci těchto informací, v případě že je subjektem žák je zde také možnost zobrazení kompletního výpisu. Tyto dodatečné informace mohou mít důvěrný charakter, proto není vhodné jejich zobrazení na základní (nezabezpečenou) obrazovku. Pod základními informacemi se nachází blok základních ovládacích prvků pro daný subjekt, ovlivňující jaké informace budou zobrazeny dále. Položky v tomto bloku jsou překreslovány pouze při změně orientace a v případě automaticky generovaných (zde počet žáků) při změně hodnoty.

V druhém bloku jsou pak zobrazeny informace na základě aktuální volby uživatele, v obrázku 5.2 absence žáků. Na spodní hraně obrazovky je pak blok ovládacích prvků souvisejících s danou volbou - v případě absence jediný, změna referenčního data.

Obrazovka výběru subjektu (zde žáka) ze seznamu je zachycena na obrázku 5.4. U horní hrany obrazovky se nachází blok s parametry pro zůžení výběru a s tlačítkem pro přidání nového subjektu, zbytek obrazovky pak zabírá seznam subjektů odpovídajících zadaným kritériím.

Veškeré prvky uživatelského rozhraní byly nastaveny tak, aby zabíraly maximální dostupný prostor. Cílem bylo umožnit ovládání i na zařízeních s menší úhlopříčkou, na velkých úhlopříčkách se proto mohou zejména některá tlačítka zdát příliš velká nebo příliš vzdálená od sebe. Použití je možné na zařízeních od přibližně čtyř palců, pro úhlopříčky do šesti palců je vhodné použití dotykového pera. Menší úhlopříčky se ukázaly jako nevhodné k zobrazení takového množství informací. Na obrázcích 5.3 a 5.5 jsou zachyceny obrazovky ze zařízení Sony Xperia M, tedy přístroje se čtyřpalcovým displejem. Zejména v případě absence je zřejmé, že se jedná o minimální možnou hodnotu.



Obrázek 5.4: Výběr studenta ze seznamu na zařízení s úhlopříčkou 10.1 palce

Obrázek 5.5: Výběr studenta ze seznamu na zařízení s úhlopříčkou 4 palce

5.4 Singleton *DataBlock*

Volby uživatele při používání aplikace ovlivňují nejen aktivitu, na kterou uživatel přechází, ale také ostatní aktivity na které by potenciálně přejít mohl. Příkladem může být volba data, kde datum nastavené v absenci je zároveň nastaveno i v ostatních částech systému. Android má pro komunikaci v aplikaci vytvořen nástroj *Intent*, ten je však určen pro komunikaci mezi dvěma aktivitami, nikoli pro nastavování globálních hodnot. V aplikaci je proto vytvořena třída *DataBlock*, která přes své rozhraní umožňuje centrální uložení klíčových proměnných.

Hlavní položkou je odkaz na databázi, s níž aplikace pracuje, a podtřídy obsahující proměnné se jmény sloupců databáze. Tento výčet umožňuje správu schématu databáze na jednom místě, zvyšuje čitelnost a usnadňuje případné změny v kódu dotazů. Dále je zde vedeno aktuálně nastavené datum, vybraná třída, žák a předmět a indikátor hlášení chyby v databázi.

Kapitola 6

Testování

Testování mobilní aplikace lze obecně rozdělit na dvě složky, programovou a uživatelskou. Tyto složky se liší co do rozsahu (a do jisté míry významu) v závislosti na konkrétním typu aplikace.

6.1 Testování programu

Tato sekce obsahuje kontrolu chování aplikace v extrémních případech, měření výkonu, spotřeby baterie a podobně. Podle [10] je pro tento typ aplikace, tedy informační systém nad lokální databází, vhodné testovat reakce na změny orientace a rychlost čtení a zápisu do databáze.

Základní reakcí systému Android na změnu orientace je ukončení a následné znovuspuštění aktivity běžící v popředí. S tímto systém počítá a veškeré hodnoty uložené v paměti zařízení jsou obnoveny. Problém nastává v případě změny orientace s otevřeným dialogovým oknem - v takovém případě je okno zavřeno a neuložené informace ztraceny. Toto je standardní chování, pro uživatele je však obtěžující. V příslušných záložkách je také nutné znovu zadat heslo, což rovněž může být nepříjemné, jedná se však o bezpečnostní prvek - toto znovuspuštění aktivity nelze odlišit od spuštění z jiného zdroje.

Rychlost čtení a zápisu je ovlivněna mnoha faktory, mimo neovlivnitelných, jako jsou odezva samotné paměti a výkon procesoru zařízení, strukturou SQL dotazů a velikostí prohledávané databáze. Za účelem testování byla vytvořena databáze rozsahem odpovídající modelové třídě pátého ročníku základní školy o přibližně 30 000 záznamech. Pozorovatelné zpoždění nastalo pouze v případech kontroly integrity databáze (viz 5.2) a výpočtů souvisejících s ukončením klasifikačního období, tyto procesy však nejsou tak časté a ze své podstaty jsou složitější než je běžné.

6.2 Uživatelské testování

Do této sekce spadají průzkumy mínění uživatelů. V případě tohoto projektu byla aplikace pro účely testování poskytnuta pedagogům Základní školy a Mateřské školy, Brno, Merhautova 37, kteří následně odpovídali na otázky v dotazníku. Testování se zůčastnilo 18 učitelů, z toho 15 z prvního stupně a 3 z druhého stupně. Následující hodnocení jsou v rozsahu 1-5 jako ve škole, tj. 1 = nejlepší, 5 = nejhorší.

- **Přehlednost aplikace:** 3x1, 8x2, 7x3, 0x4, 0x5.

- **Přehlednost zadávání údajů:** 2x1, 8x2, 8x3, 0x4, 0x5.
- **Jsou sledované údaje o žácích dostatečné?:** 15x ano, 3x ne.
- **Používali byste tuto aplikaci pokud by byla v rámci školy dostupná?:** 7x1, 5x2, 5x3, 1x4, 0x5.
- **Za jakou cenu byste byl/a ochotna si ji koupit sám/sama?** 5x nekoupil/a bych ji, 5x do 20kč, 7x do 50kč, 1x do 100kč, 0x více než 100kč

Odpověď na první otázku úzce souvisí s otázkou druhou, výhrady byly převážně stejné. Jednalo se o systém přiřazování žáků, kde je nutné žáka napřed vytvořit a teprve pak jej lze přiřadit do třídy. Chybějícími statistikami bylo ve dvou případech čtvrtletní hodnocení a v jednom případě detailnější informace o rodičích žáků.

Kapitola 7

Závěr

Cílem práce bylo vytvořit informační systém pro učitele základních a středních škol, vhodný pro použití na mobilních zařízeních, využívajících operační systém Android.

Kapitoly 3 a 4 se zabývají konkrétními požadavky učitelů na tento systém, kapitoly 2 a 5 se pak zabývají obecným přístupem k programování aplikací pro systém Android a řešením jednotlivých hlavních sekcí programu.

Ukázalo se, že vzhledem k rychlosti vývoje mobilních technologií existuje poměrně málo aktuální tištěné literatury, byly proto použity především digitální zdroje a publikace. V tomto oboru však právě tyto zdroje obsahují nejrelevantnější informace a aktuálně doporučované postupy.

Protože v záplavě dnešních aplikací je obtížné zaujmout natolik, aby si potenciální uživatel alespoň přečetl popis, bylo by vhodné do případného dalšího vývoje zahrnout novou grafickou úpravu, z funkčního hlediska pak možnost synchronizovat data s centrálním serverem.

Literatura

[1] Allen, G.: *Android 4: Průvodce programováním mobilních aplikací*. Computer Press, 2013, iSBN 9788025137826.

[2] Basl, J.: Informační systémy škol: specifická oblast využití manažerských informačních systémů. *Ikaros [online]*, ročník 12, 2006 [cit. 2014-01-16], `http://www.ikaros.cz/informacni-systemy-skol-uziti-manazerskych-informacnich-systemu`.

[3] Dotson, C.: The Hills Are Greener: Death to SD Cards? `http://www.androidrundown.com/blog/hills-greener-death-sd-cards/`, 2013-07-10 [cit. 2014-01-18].

[4] Janessa Rivera, R. v. d. M.: Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. `http://www.gartner.com/newsroom/id/2665715?fnl=search`, 2014-02-13 [cit. 2014-04-27].

[5] Main, S.: Say Goodbye to the Menu Button. *Android Developers Blog [online]*, 2012 [cit. 2014-01-18], `http://android-developers.blogspot.cz/2012/01/say-goodbye-to-menu-button.html`.

[6] Staff, V.: Android: A visual history. `http://www.theverge.com/2011/12/7/2585779/android-history`, 2011-12-07 [cit. 2014-01-08].

[7] Works, T.: Samsung Premium House to Inspire a Future Home Life with Innovative Appliances and Timeless Design at IFA 2013 Berlin. `http://global.samsungtomorrow.com/?p=27183#sthash.hf49eyKK.dpuf`, 2013-09-04 [cit. 2014-01-08].

[8] WWW stránky: IBM Worklight. `http://www-03.ibm.com/software/products/en/worklight`, [cit. 2014-01-11].

[9] WWW stránky: Android. `http://www.android.com`, [cit. 2014-02-10].

[10] WWW stránky: Android Developers. `http://developer.android.com`, [cit. 2014-02-10].

[11] ČESKO: Školský zákon č. 561/2004 Sb. 2008.

Dodatek A

Obsah CD

- elektronická verze písemné zprávy a návod k instalaci ve formátu pdf
- zdrojový kód písemné zprávy
- zdrojové kódy Informačního systému
- přeložená aplikace Informačního systému