Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

eLogika nad platformou Android

eLogika for Android System

2016 **Bc. Jiří Znoj**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jiří Znoj**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: eLogika nad platformou Android
eLogika for Android System

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je analýza, návrh a implementace systému eLogika pro tablety s OS Android 4.2 nebo novější a mobilní telefon se systémem Android 4.1 nebo vyšší.

Zásady pro vypracování:
1. Seznamte se s problematikou a metodami systémového návrhu a implementace pro Android.
2. Proveďte analýzu a návrh systémů.
3. Navrhněte rozhraní pro komunikaci se serverem systému eLogika.
4. Pomocí vytvořeného návrhu naimplementujte aplikace, které budou využity při výuce předmětu Matematická logika na VŠB-TU Ostrava.

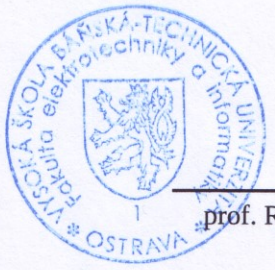
Seznam doporučené odborné literatury:

[1] Hellman, E.: Android Programming: Pushing the Limits, Wiley, 2013, ISBN: 978-1118717370
[2] Iversen, J., Eierman, M.:Learning Mobile App Development: A Hands-on Guide to Building Apps with iOS and Android, Addison-Wesley Professional, 2013, ISBN: 978-0321947864

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

Vedoucí diplomové práce: **Mgr. Marek Menšík, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016



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.

V Ostravě dne: 26. dubna 2016 .....

podpis studenta

Poděkování

Děkuji své rodině a přítelkyni za podporu při vytváření této diplomové práce. Dále děkuji svému vedoucímu Mgr. Marku Menšíkovi, Ph.D., za konzultace a Mgr. Jitce Snášelové, DiS., za korekturu textu.

Abstrakt

Cílem práce bylo zabývat se analýzou, návrhem a implementací systému eLogika pro tablety s OS Android ve verzi 4.2 nebo novější a mobilní telefony s OS Android ve verzi ne starší než 4.1. V úvodu je popsána motivace pro vznik android aplikací v rámci systému eLogika. Dále se práce věnuje analýze a návrhu jedné aplikace pokrývající zadané požadavky, v další kapitole pak návrhu pro komunikaci se serverem systému eLogika. Následující kapitola popisuje vše, co se týká implementace aplikace sloužící k výuce předmětu Matematická logika na VŠB-TUO. V závěru je popsáno celkové shrnutí vzniklé aplikace a diplomové práce.

Klíčová slova

Android; Android OS, Android Studio; aplikace; eLogika; analýza, návrh, komunikace, implementace, REST

Abstract

The aim of this thesis was to deal with analysis, design and implementation of the system eLogika for tablets with Android operating system version 4.2 or newer and mobile phones with Android operating system in version not older than 4.1. In the introduction it is described the motivation for creating android applications in the system eLogika. The text below introduces this work, focusing on analysis and design of the application, which covers given requirements. The following chapter deals with the design of communication of the application with systém server eLogika. The chapter after that one focuses on all stuff around implementation of the application serving the purpose of teaching subject Mathematic logic on VŠB-TUO. At the end of this thesis it is written the summary of the implemented application and diploma work.

Key words

Android; Android OS, Android Studio; application; eLogika; analysis, design, communication, implementation, REST

Obsah

Seznam použitých zkratek ..... - 9 -
Seznam ilustrací ..... - 11 -
Seznam tabulek ..... - 13 -
Úvod ..... - 14 -
1 Problematika a metody systémového návrhu a implementace pro Android ..... - 15 -
1.1 Systémový návrh ..... - 15 -
2 Analýza a návrh ..... - 17 -
2.1 UML ..... - 17 -
2.2 Role v systému a jejich případy užití ..... - 19 -
2.3 Kvalitativní (nefunkční) požadavky ..... - 25 -
3 Návrh rozhraní pro komunikaci se serverem systému eLogika ..... - 28 -
3.1 Systém eLogika ..... - 28 -
3.2 REST API ..... - 29 -
3.3 Metody navržené pro komunikaci se serverem systému eLogika ..... - 29 -
3.4 Volley ..... - 32 -
3.5 Logování aplikace ..... - 32 -
4 Implementace ..... - 33 -
4.1 Prostředí pro vývoj ..... - 33 -
4.2 Způsob vedení vývoje ..... - 33 -
4.3 Implementace jednotlivých případů užití ..... - 35 -
4.4 Bezpečnost offline testů ..... - 46 -
4.5 Jazykové mutace ..... - 46 -
4.6 Použité návrhové vzory ..... - 47 -
4.7 Android API ..... - 48 -
4.8 Knihovna vlastních funkcí ..... - 50 -
4.9 Šablony pro View ..... - 53 -
4.10 Knihovny třetích stran ..... - 54 -
4.11 Spuštění aplikace ..... - 54 -
5 Závěr ..... - 55 -
6 Použitá literatura ..... - 57 -

7 Seznam příloh..... - 61 -

Seznam použitých zkratek

Zkratka	Význam
ADB	Android Debug Bridge
AES	Advanced Encryption Standard
API	Application Programming Interface
APK	Android application package file
ARC	App Runtime for Chrome
ECB	Electronic Codebook
FullHD	Full High-Definition - rozlišení 1920x1080 pixelů
GB	Gigabyte
GOF	Gang of Four
GUI	Grafické uživatelské rozhraní
HAXM	Hardware Accelerated Execution Manager
HD	High-Definition – rozlišení 1280x720 pixelů
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
ID	Identifikátor
IP	Internet Protocol
JDK	Java Development Kit
JSON	JavaScript Object Notation
LMS	Learning Management System
MB	Megabyte
MD5	Message-Digest algorithm - hašovací funkce
MS	Microsoft
OS	Operační systém
PDF	Portable Document Format
PSČ	Poštovní směrovací číslo
QR	Quick Response
RAM	Random Access memory
REST	Representational State Transfer

SDK	Software development kit
SHA1	Secure Hash Algorithm 1 - hašovací funkce
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SW	Software
TFS	Team Foundation Server
UML	Unified Modeling Language
VGA	Video Graphics Array – rozlišení 640x480 pixelů
WCF	Windows Communication Foundation
WP	Windows Phone
WVGA	Wide VGA – rozlišení 800x480 pixelů
XML	eXtensible Markup Language

Seznam ilustrací

1 Třída Logging, která se zabývá logováním všech akcí v rámci aplikace ..... - 32 -
2 Diagram životního cyklu android aktivity [5] I
3 Diagram životního cyklu fragmentu [6] II
4 Diagram verzí OS Android, verzí API a počet zařízení k 14. 4. 2016 – výřez snímku obrazovky programu Android Studio 2.0, který lze stáhnout zde: [2] III
5 Ukázka programu SourceTree stažitelného zde: [19] IV
6 Ukázka TFS dostupného na adrese http://elogika.vsb.cz:8080/tfs/eLogika%20Collection/Android IV
7 Ukázka programátorské dokumentace JavaDoc V
8 Diagram ukázky klasifikace UML diagramů UML 2.5 [3] VI
9 Levá polovina třídního diagramu pro případ užití „Zavedení podmínek kurzu“ pro uživatele v roli garanta VII
10 Pravá polovina třídního diagramu pro případ užití „Zavedení podmínek kurzu“ pro uživatele v roli garanta VIII
11 Třídní diagram návrhového vzoru Šablona (Template method) IX
12 Třídní diagram návrhového vzoru Továrna (Factory method) X
13 Třídní diagram návrhového vzoru Jedináček (Singleton) XI
14 Třídní diagram Hierarchického adaptéru se třídou Item demonstrující návrhový vzor Kompozit (Composite) XII
15 Třídní diagram komunikace s knihovnou Volley XIII
16 Třídní diagram některých pomocných tříd aplikace eLogika XIV
17 Třídní diagram některých pomocných tříd aplikace eLogika XV
18 Diagram případů užití pro nepřihlášeného uživatele a studenta XVI
19 Diagram případů užití pro uživatele v roli tutora XVI
20 Diagram případů užití pro uživatele v roli garanta XVII
21 Diagram aktivit procesu přihlášení XIX
22 Aktivitní diagram pro ověření přihlašovacích údajů XX
23 Sekvenční diagram popisující přihlášení na termín a odhlášení z termínu uživatele v roli studenta XXI
24 Sekvenční diagram zachycující stažení materiálu. XXII
25 Sekvenční diagram popisující odevzdání aktivity uživatelem v roli studenta XXIII
26 Sekvenční diagram zachycující přihlášení do aplikace s výběrem školy a role XXIV
27 Levá polovina sekvenčního diagramu ke správě konzultačních hodin XXV
28 Pravá polovina sekvenčního diagramu ke správě konzultačních hodin XXVI
29 Levá polovina sekvenčního diagramu ke správě tutorů XXVII
30 Pravá polovina sekvenčního diagramu ke správě tutorů XXVIII
31 Diagram komunikace ukazující komunikaci s knihovnou Volley XXIX
32 Diagram komunikace pro případ užití „Konzultační hodiny“ pro uživatele v roli studenta XXIX
33 Diagram komunikace zachycující vytváření a editaci šablon XXX

34 Diagram komunikace pro zobrazování veřejných kurzů, přihlašování na ně a odhlašování z veřejného kurzu, na který je uživatel přihlášen XXXI
35 Diagram nasazení pro systém eLogika..... XXXII
36 Zleva: LG Optimus 2x (Android 4.4), Xiaomi Mi5 (Android 6.0), Lenovo Yoga 2 (Android 5.0) XXXIII
37 Nahoře jsou znázorněny emulátory Android Studia (Android N, Android 6.0, Android 4.1), dole je pak ukázka okna ARC XXXIV
38 Okno s hlášením chyby v OS Android 4.1..... XXXV
39 Okno s hlášením chyby v OS Android 5.0..... XXXV
40 Okno vypracovaného online testu XXXVI
41 Ukázka postranního menu aplikace..... XXXVI
42 ViewPager s ListView a ukázkou načítání SwipeRefreshLayoutu v rámci šablony list_view_swipe XXXVI
43 Ukázka Snackbaru překrývajícího ListView s aktivitami XXXVII
44 Volba času uvnitř knihovny TimePickerDialogOnButton XXXVII
45 Ukázka načítání testu a otázek XXXVII
46 Ukázka vypracování, vyhodnocení a zobrazení testu XXXVIII
47 Aktuality zobrazované uživateli v roli studenta XXXIX
48 SeparatedListAdapter XXXIX
49 Emailové zprávy..... XL
50 Výběr data uvnitř knihovny DataPickerDialogOnButton XL
51 Ukázka použití knihovny HierarchicalAdapter XLI
52 Ukázka použití knihovny TimePickerDialogButton_HMS..... XLI
53 Ukázka případu užití Správa kapitol pro uživatele v roli Garanta XLII
54 Ukázka použití knihovny SpecialCharDialogButton XLII
55 Ukázka rovrzhu pro uživatele v roli Garanta XLIII
56 Ukázka rozvrhu pro uživatele v roli studenta..... XLIII
57 Ukázka použití RecyclerView..... XLIV
58 Ukázka ActionBaru a filtrování pomocí loginu XLIV

Seznam tabulek

1 Scénář případů užití pro případ užití „Nahlásit chybu“ XVIII

Úvod

Systém eLogika je LMS systém, tedy redakční systém určený pro e-learning. Slouží k podpoře výuky s využitím informačních a komunikačních technologií. V současné době je systém eLogika přístupný skrze webový prohlížeč (webovou aplikaci) a mobilní aplikace pro Windows 8.1 a Windows Phone 8.1. Jednou z motivací vzniku aplikace eLogika pro OS Android je fakt, že se jedná o nejrozšířenější mobilní platformu.

Největší výhodou mobilní aplikace (oproti webové) je bezesporu možnost přístupu k vybraným částem aplikace bez nutnosti internetového připojení. Mobilní aplikace také snižují množství přenesených dat mezi serverovou částí a koncovým zařízením, což má za následek nižší zátěž serveru a tím i rychlejší odezvu systému. Optimalizace ve smyslu datového toku je velice důležitá z důvodu mnohdy slabého internetového připojení skrze mobilní data nebo kvůli omezení množství přenesených dat mobilním operátorem. Snížení přenosu dat skrze internetové připojení je dosaženo nejen použitím vhodného návrhu pro komunikaci, ale také například díky uživatelskému rozhraní, které není nutné ze serveru stahovat. Další výhodou uživatelského rozhraní koncového zařízení je jeho přizpůsobení se displeji koncového zařízení. Uživatelské rozhraní je často u mobilních aplikací složeno z prvků verze OS koncového zařízení, což napomáhá uživateli v rámci aplikace ke snadnější orientaci.

Jednou z výhod e-learningu je časová a geografická flexibilnost. Rozšíření eLogiky o mobilní zařízení je právě v této oblasti velkou výhodou. Mobilní zařízení jsou čím dál výkonnější a spousta uživatelů je dnes používá častěji než stolní či přenosné počítače. Mobilní telefony máme stále u sebe, a tak je možnost vzdělávat se doslova „na dosah ruky“.

Následující kapitoly této diplomové práce se zabývají krátkým představením komponent využívaných pro vývoj android aplikací (standardních, vlastních nebo knihoven třetích stran). V textu se vyobrazuje analýza a návrh aplikací s ukázkami analýzy a návrhu aplikace eLogika pro zařízení s OS Android. Dále jsou zde řešeny funkce aplikace a jejich implementace, některé použité návrhové vzory a další podkapitoly týkající se implementace. V příloze se pak nachází všechny metody používané pro komunikaci se serverem systému eLogika a v textu jsou některé z navržených metod popsány.

1 Problematika a metody systémového návrhu a implementace pro Android

1.1 Systémový návrh

Životní cyklus aktivity

Aktivita je třída, která zodpovídá za vyobrazení GUI uživateli a zpracování jeho interakcí se zobrazenými android komponentami. Jednotlivé aktivity jsou na sobě nezávislé a mohou se nacházet v různých stavech. Přechody mezi těmito stavy řídí OS Android a během změn stavů volá metody, jejichž výchozí chování lze v každé třídě dědící z třídy aktivity překrýt. Schéma s jednotlivými stavy, znázorněnými přechody a volanými funkcemi lze nalézt v příloze A na obrázku 2. [9]

Metoda onCreate() se volá po prvním spuštění aktivity, jejím znovuspuštění, při změně zdrojů nebo z důvodu potřeb OS (například nedostatku paměti). Vytváří se zde GUI a definují se proměnné s objekty potřebnými pro běh aktivity. Metoda onStart() se stará o zobrazování uživatelského rozhraní a volá se po skončení metody onCreate(). onResume() metoda je volána, pokud byla aktivita na pozadí a přechází zpět do popředí. onPause() je metoda, která se volá při přesunu běžící aktivity do pozadí. Zde je vhodné ukládat data, která ještě nebyla uložena a uložit by se měla. Volání onStop() metody je realizováno, pokud byla aktivita zastavena a uložena na zásobník, ze kterého může být ještě obnovena. V případě, že dojde k takovémuto obnovení, je zavolána metoda onRestart(). Metoda onDestroy(), jak je patrné z výše zmiňovaného diagramu, je volána v případě, že byl ukončen její životní cyklus. [9]

Fragmenty

Fragmenty reprezentují chování nebo část chování uživatelského rozhraní v aktivitě. V jedné aktivitě může být zobrazeno více fragmentů – jeden fragment se například zobrazí při použití android zařízení „na výšku“ a v případě zobrazení „na šířku“ se zobrazí fragmenty dva. Fragmenty jsou takové odlehčené aktivity, které lze opakovaně použít. [8]

Fragmenty byly představeny v systému Android 3.0 pro podporu zobrazení na velkých zobrazovacích plochách, jakými jsou například tablety nebo telefony s vysokým rozlišením. Každý fragment by měl být modulární a znovupoužitelný v různých aktivitách. [8]

Životní cyklus fragmentu je znázorněn grafem v příloze A na obrázku 3. Fragment se může nacházet, stejně jako aktivita, ve 3 různých stavech. Obnoven (Resumed), pokud je fragment viditelný v běžící aktivitě, pozastaven (Paused), když je na pozadí aktivity, jež je stále viditelná, a zastaven (Stopped) je fragment tehdy, pokud není viditelný, pokud je aktivita zastavena nebo pokud se fragment nenachází v právě běžící aktivitě. [8]

Největší rozdíl mezi životním cyklem aktivity a životním cyklem fragmentu je ve způsobu jejich uložení na zásobníku. Aktivita je uložena na zásobník spravovaný operačním systémem, kdežto fragment je uložen na zásobník spravovaný aktivitou, a to jen v případě, že je to onou aktivitou explicitně vyžadováno. [8]

Aplikace bude mít mnoho různých způsobů rozložení prvků na obrazovce android zařízení – dále uváděných jako „okna aplikace“, která budou některá data a prvky uživatelského rozhraní sdílet. Aby se zamezilo častému vytváření takových prvků, kterými jsou například boční menu a horní panel aplikace, pak bude-li to možné, budou pro různá okna použity různé fragmenty uvnitř aktivity se sdílenými daty a některými prvky GUI.

Návrh vzhledu

Jako předloha pro vzhled android aplikace poslouží webová aplikace na adrese https://eLogika.vsb.cz, takže pro uživatele, kteří jsou zvyklí používat webovou aplikaci eLogika, bude snadné se zorientovat i v aplikaci eLogika pro android.

Při návrhu aplikace bude brán zřetel také na to, že pro uživatele, který vystupuje v roli studenta, bude primární použití aplikace v mobilním telefonu, tedy (nejčastěji) v režimu „na výšku“. Pro uživatele v roli garanta nebo tutora se bude naopak při návrhu aplikace předpokládat zobrazení na šířku, neboť výše zmínění budou aplikaci využívat v tabletu.

Material design

V roce 2014 [10] byl představen nový návrh vzhledu android aplikací – Material design, který byl poprvé uveden v OS Android 5.0. Google již (2016) aktualizoval své mobilní aplikace, aby splňovaly principy Material designu a k tomuto stylu se postupně přibližuje i s aplikacemi, které nejsou určeny výhradně pro platformu android (například některé prvky prohlížeče Chrome pro OS Windows nebo webová aplikace https://plus.google.com/).

Google stále pracuje na podpoře Material designu i pro zařízení se starší verzí OS Android, než je verze 5.0, a to pomocí knihoven „Support Library“. Více informací o těchto knihovnách lze nalézt zde: [11]

Při představení Material designu již aplikace eLogika pro android vznikala a nebylo ještě možné vzhled Material designu používat, neboť aplikace byla spustitelná i na starších verzích OS Android, než je verze 5.0. Postupně s vývojem aplikace se vyvíjela i knihovna pro android vývojáře v7 appcompat library [11] usnadňující vývoj aplikací, jež splňují principy Material designu. Především díky této knihovně jsou již některé části aplikace navrženy tak, že dodržují principy, doporučení, využívají styly anebo používají komponenty navržené ve stylu Material designu. Detailnější informace o pravidlech, principech, komponentách a stylech lze nalézt zde: [12].

Aplikace si v rámci diplomové práce neklade za cíl dodržet výše zmíněná pravidla Material designu, ale seznámit se s nimi a při tvorbě aplikace se jimi inspirovat.

2 Analýza a návrh

2.1 UML

UML neboli Unified Modeling Language (unifikovaný modelovací jazyk) je jazyk sloužící ke specifikaci, vizualizaci, tvorbě a popisu částí softwaru. Jedná se o vizualizace pomocí různých diagramů, které lze napojit na různé programovací jazyky. [13]

Diagramy jazyka UML, podle normy UML 2.5, lze rozdělit do 2 kategorií – Strukturální diagramy a diagramy chování. Další dělení lze pak vypozorovat z obrázku 8, přičemž modrou barvou jsou vyznačeny diagramy, které nejsou částí oficiální klasifikace UML 2.5 diagramů. Více o UML 2.5 diagramech lze nalézt zde: [14]

Strukturální diagramy

Strukturální diagramy slouží k náhledu na statickou část systému v různých úrovních jeho abstrakce spolu s vazbami mezi těmito jednotlivými úrovněmi. Diagramy patřící do kategorie strukturálních diagramů jsou koncepty navrhovaného systému. [14]

Třídní diagramy

Třídní diagramy se snaží zachytit navrhovaný systém z pohledu jeho třídy, obsahu tříd a statických vazeb, které třídy mezi sebou mají. Třída je v diagramu zachycena jejím názvem, vlastnostmi – atributy a pak chováním – metodami.

Vlastnosti a chování třídy bývají doplněny znakem viditelnosti k ostatním třídám. Soukromé (vlastnosti a chování) se označují znakem „-“, veřejné znakem „+“, chráněné znakem „#“ anebo „~“, což značí viditelnost v rámci balíčku. Dále je možné uvádět u atributů tříd jejich typ, násobnost, implicitní hodnotu a další vlastnosti či omezení. U metod popisované třídy se uvádí její návratový typ a argumenty včetně jejich typů s možností uvedení jejich implicitní hodnoty. V případě potřeby je možné uvést další vlastnosti či omezení dané metody.

Vazby mezi třídami jsou nazývány asociacemi. Tyto asociace mohou být pojmenovány. U tříd v rámci asociací může být uvedeno, v jaké roli třída do vztahu (k jiné třídě) vstupuje. Násobnost asociace udává v diagramu informaci o tom, kolik objektů dané třídy se bude vztahu účastnit. Buď se uvádí přesné číslo, nebo interval minimálního až maximálního počtu objektů ve vazbě. Libovolný počet zastupuje znak „*“ a znak „+“ nese informaci, že alespoň jeden objekt dané třídy je ve vztahu vždy přítomen.

V příloze B jsou přiloženy některé z diagramů použitých při návrhu struktury aplikace.

Diagramy nasazení

Diagramy nasazení slouží k vyobrazení architektury systému, jakožto rozložení softwarových artefaktů do cílů implementace. [14]

Artefakty jsou položky vzniklé vývojem - např. spustitelné soubory, knihovny, databáze, konfigurační soubory. Cíle implementace jsou uzly, kterými bývají buď hardwarová zařízení,

nebo spustitelný software. Tyto uzly mohou být vzájemně propojeny přes komunikační kanály. [14]

Diagram nasazení pro systém eLogika lze nalézt v příloze H.

Diagramy chování

Na rozdíl od strukturálních diagramů se diagramy chování nesoustředí na statické části systému, ale na jeho chování za běhu. Jedná se o popis jednotlivých stavů v systému v závislosti na čase. [14]

Diagram případů užití

UML diagram případů užití, také často vystupující pod názvem „use case diagram“, je jedním z typu diagramů chování. Diagramy případů užití jsou tvořeny aktéry a případy užití. Aktérem může být uživatel systému nebo třeba i jiný systém, který s popisovaným systémem nějakým způsobem přichází do kontaktu. Případy užití jsou akce, které může aktér se systémem vykonávat.

Spolu s diagramem případů užití se často uvádí scénáře případů užití a dohromady pak tvoří model případů užití. Scénáře případů užití detailně popisují jednotlivé případy užití, neboť o nich samotný diagram případů užití mnoho neříká. Use case scénář je složen z několika částí, které podrobně popisují jednotlivé případy užití. Těmito částmi jsou obvykle aktéři, pak předpoklady, které musí být splněny před samotným průběhem případu užití, průběh s jednotlivými kroky, možný alternativní průběh a výjimky.

Diagramy případů užití pro aplikaci eLogika lze nalézt v příloze B, ukázku scénáře případu užití pak v příloze D.

Diagramy aktivit

Diagramy aktivit slouží k popisu průběhu nějaké činnosti v rámci systému (aktivity) v závislosti na čase. Jedná se o sekvenci dílčích činností (aktivit), přechodů mezi nimi, podmínek blíže určujících výběr následující aktivity. Jde tedy o bližší popis chování vybrané části systému.

Počáteční stav je značen bodem, do kterého nevede žádný přechod. Přechody udávají, jaká událost bude následovat za stavem, ze kterého je přechod vyznačen. Přechody jsou zaznamenány ve tvaru šipky, takže je zcela jednoznačný směr toku událostí. Další stavy diagramu se nazývají aktivitami, což jsou dílčí činnosti popisované aktivity a mohou být blíže specifikovány dalšími diagramy aktivit. Rozhodovací blok slouží k jednoznačnému určení následujícího stavu diagramu, pokud je potřeba rozhodnout se v rámci přechodu dle nějaké podmínky. Tok se může rozdělit do paralelních běhů, které se opět sjednotí. V rámci diagramu aktivit se uvádí ještě jeden přechod, který se nazývá koncovým stavem, je vyznačen bodem, který má kolem sebe prstenec a nevychází z něj již žádný další přechod. Jednoznačně je tak určen konec popisované aktivity.

Ukázky diagramů aktivit lze nalézt v příloze E. Nachází se zde diagram ukazující proces přihlášení a diagram pro ověření přihlašovacích údajů, což je jedna z aktivit předchozího

diagramu. Jedná se tedy o blíže specifikovanou část aktivitního diagramu ukazujícího proces přihlašování.

Sekvenční diagramy

Sekvenční diagramy popisují komunikaci mezi objekty se zachycením časové posloupnosti událostí. Objekty mezi sebou komunikují prostřednictvím zpráv.

V rámci diagramu jsou vodorovně vypsány objekty a svisle se vyznačuje časový průběh. Čára vycházející svisle ze znázorněných objektů značí časovou délku jejich existence vzhledem k existenci dalších vyznačených objektů. Na této svislé čáře pak může být vyznačena přibližná doba nějaké činnosti, kterou v rámci události, kterou sekvenční diagram popisuje, objekt vykonává.

Zprávy mohou být synchronní, což znamená, že objekt čeká na odpověď a další činnost bude provádět až po jejím obdržení. Synchronní zprávy se značí plnou čarou a plným trojúhelníkem na straně, kam šipka ukazuje. Odpověď na zprávy obecně se značí šipkou čárkovanou. V rámci odpovědi může být vrácena nějaká návratová hodnota. Asynchronní zprávy, kdy objekt na odpověď nečeká a může dále pokračovat v nějaké činnosti, se značí jednoduchou šipkou s plnou čarou (nejedná se o plný trojúhelník).

V sekvenčních diagramech je možné zaznamenat cykly, podmínky ovlivňující část průběhu události v rámci diagramu, vytváření a možný zánik objektů. Všechny části diagramu lze komentovat pomocí poznámek.

Několik sekvenčních diagramů použitých k návrhu systému lze nalézt v příloze F.

Diagramy komunikací

Diagram komunikací je velmi podobný sekvenčnímu diagramu s tím rozdílem, že při popisu komunikace mezi objekty zachovává jejich logické rozložení a pořadí zasílání zpráv je určeno číslem (udávajícím pořadí zasílané zprávy). Pravidla pro interpretaci synchronních zpráv, asynchronních zpráv a poznámek jsou stejná jako v sekvenčních diagramech.

Některé diagramy komunikace sloužící k popisu vybraných částí systému lze nalézt v příloze G.

2.2 Role v systému a jejich případy užití

V systému může uživatel vystupovat ve 4 různých rolích podle funkce, kterou může v daném kurzu mít. Uživatel dále bude vystupovat podle své role jako nepřihlášený uživatel, jako student, tutor anebo jako garant. Diagramy případů užití pro všechny role lze nalézt v příloze B a více informací o diagramech případů v kapitole Diagram případů užití.

Nepřihlášený uživatel

Veřejné kurzy

Nepřihlášený uživatel zobrazí seznam veřejných kurzů. Pro každý veřejný kurz si může zobrazit jeho podrobný popis a v případě nenaplněné kapacity se na něj přihlásit.

Při zapsání do veřejného kurzu je nutné vyplnit přihlašovací údaje.

Student

Aktuality

Studentovi se zobrazují aktuality. Pokud má aktualita nějaké přílohy, tak se zobrazí a student si je může stáhnout a zobrazit.

Přihlásit se na termín

Student může zobrazovat termíny skupin aktivit, termíny testů a termíny aktivity. U každého termínu lze zobrazit jeho detaily a následně je pak možné se na jednotlivé termíny přihlásit či se z nich odhlásit. Přihlášení a odhlášení je proveditelné pouze pro termíny, které jsou pro danou operaci v daném čase aktivní.

Vypracovat test

Student je schopen v aplikaci zobrazit testy, zobrazené testy spustit nebo uložit pro pozdější vypracování offline (bez připojení k internetu). Uložení testů je možné pouze pro testy, které jsou takto označeny.

Vypracované testy

Student má možnost nechat si vypsat výsledky vypracovaných testů a v jednotlivých testech pak zobrazit správné a chybné odpovědi, pokud to daný test umožňuje. V případě, že se jednalo o papírový test, je možné nahlásit podezření, že byl test špatně vyhodnocen.

Hodnocení kurzu

Student je schopen si zobrazit celkové hodnocení kurzů, na které je přihlášen. U jednotlivých kurzů jsou pak také vypsána jednotlivá hodnocení.

Aktivity

Aktivity jsou rozděleny na „aktuálně vypsané" a „odevzdané a vypracované".

U aktuálně vypsaných aktivit si student zobrazuje jejich detail a pak zde vkládá řešení vybrané aktivity. Pokud byla aktivita již odevzdaná, tak ji zde může stáhnout a zobrazit.

U odevzdaných a vypracovaných aktivit student zobrazuje jejich detailní popis, odevzdané řešení nebo odevzdané řešení mění za řešení nové.

Rozvrh

Studentovi se zobrazí rozvrh, na který je přihlášen. Student má také možnost se v aplikaci přihlásit do vybraného rozvrhu a přidat do něj vlastní předmět.

Teorie

Student si zobrazuje seznam kapitol. Pro každou kapitolu se mu pak vypisuje seznam podkapitol (seznam kapitol ve vybrané kapitole), seznam materiálů a seznam cvičných testů pro vybranou kapitolu, které lze rovnou vypracovat.

Konzultační hodiny

Studentovi se zobrazí vypsané konzultační hodiny a je mu umožněno se na ně přihlásit. Z přihlášených konzultačních hodin se student v případě potřeby odhlásí.

Profil

Student si upravuje údaje, které jsou o něm vedeny v systému (příjmení, e-mail, jazyk, ulici, číslo popisné, PSČ, město, titul za jménem, titul před jménem, fotografii, heslo)

Veřejné kurzy

Student zobrazuje seznam veřejných kurzů včetně jejich detailů. Ke každému zobrazenému kurzu se může přihlásit, nebo se z něj odhlásit.

Nahlásit chybu

Kdekoli v systému je studentovi umožněno nahlásit chybu.

Garant

Zavedení podmínek kurzu

Pokud je uživatel přihlášen jako garant, pak může v aplikaci přidávat skupiny aktivit a také je pak editovat či mazat. Garant mění minimum/maximum kurzu a zobrazuje celkový souhrn výsledků kurzu. V tomto souhrnu se zobrazují studenti daného kurzu a u jednotlivých studentů jejich jednotlivé výsledky. Každý výsledek lze upřednostnit a zjistit výsledky za jednotlivé skupiny aktivit.

Garant zobrazuje skupiny aktivit zvlášť pro prezenční a zvlášť pro kombinovanou formu studia. U každé skupiny aktivit lze, pokud je uživatel přihlášen jako garant, vypsat seznam termínů, termíny upravovat a u každého termínu vypsat přihlášené studenty, historii přihlášení, přihlašovat studenty na termín, jednotlivé studenty odhlašovat, tisknout seznam studentů, tisknout QR kódy studentů nebo studentům přihlášeným na daný termín napsat zprávu.

Pro každou skupinu aktivit je možné přidat termín, vypsat souhrn výsledků skupiny aktivit, kde lze pro jednotlivé studenty vypsat všechny jejich výsledky a vybrané výsledky pak upřednostnit.

Správa tutorů

Uživateli v roli garanta je umožněno vytvářet nové tutory pro vybraný kurz, a to buď zcela jako nové uživatele, nebo z uživatelů již existujících. Existující uživatelé jsou vyhledáváni podle zadaného příjmení (nebo jeho části) a vybraným jsou přiřazena práva tutora. Všechny tutory daného kurzu lze zobrazit, editovat jejich údaje nebo je z kurzu odebrat.

Seznam tříd a jejich studentů

Třídy vybraného kurzu jsou dvojího druhu. Buď se jedná o třídy kombinované formy studia, nebo o třídy formy prezenční. Třídy kombinované formy studia obsahují podtřídy pro jednotlivé tutoriály.

Lze vytvořit novou třídu jak pro prezenční, tak pro kombinovanou formu studia. Při vytváření tříd kombinované formy se rovnou vytváří podtřídy s konkrétními daty a detaily jednotlivých tutoriálů. Každou třídu lze také editovat či odstranit.

Pro všechny třídy garant zadává časové omezení pro změnu tříd – období pro přihlašování a pro odhlašování ze tříd.

Ke každé třídě lze přiřadit studenty, jednotlivé studenty ze třídy odebrat, zobrazit všechny studenty dané třídy nebo studentům odeslat hromadnou e-mailovou zprávu.

Správa testů a aktivit

Garant zobrazuje pro vybranou skupinu aktivit aktivity a testy, které obsahuje. Dále má možnost vytvářet nové aktivity, existující editovat či mazat.

V aplikaci bude garantovi umožněno přidat nový termín, zobrazit seznamy termínů jednotlivých aktivit, tyto termíny upravovat a u každého z nich vypsat přihlášené studenty. Garant je schopen v aplikaci studenty na termín přihlašovat, odhlašovat anebo si historii přihlašování nechat vypsat. Uživatel, jakožto garant, má možnost si stáhnout a zobrazit seznam studentů vybraného termínu ve formátu pdf nebo je schopen analogickým způsobem získat pdf soubor se seznamem QR kódů studentů daného termínu. Uživatel v roli garanta může studentům přihlášeným na vybraný termín napsat zprávu. U každého termínu vybrané aktivity si lze nechat vypsat řešitele, zobrazit jejich řešení a udělit jim hodnocení.

Pro testy vybrané skupiny aktivit bude garantovi umožněna obdobná funkčnost, jako je tomu u aktivit. Tedy vytvářet nové, stávající editovat a mazat.

Pro každý test je garant schopen vytvořit nový termín, editovat stávající nebo zobrazit seznam termínů vybraného testu. Pro každý termín lze vypsat přihlášené studenty, studenty na termín přihlašovat, odhlašovat anebo si historii přihlašování nechat vypsat. Garant má možnost si stáhnout a zobrazit seznam studentů vybraného termínu ve formátu pdf nebo je schopen analogickým způsobem získat pdf soubor se seznamem QR kódů studentů daného termínu. Uživatel v roli garanta může studentům přihlášeným na vybraný termín napsat zprávu. Pro každý termín je garantovi umožněno nechat si vypsat vygenerované testy nebo test vygenerovat.

Správa otázek

Garant zobrazuje otázky podle vybrané kapitoly, dále může blíže specifikovat kategorii a zařazení testů (Hlavní, Cvičný nebo Ukázkový), ve kterých se otázka objevuje. Otázky lze také zobrazovat pouze nepřekontrolované, pouze překontrolované nebo všechny. Je možné vytvářet otázky nové a existující mazat.

Správa kategorií

Uživateli v roli garanta je umožněno zobrazovat kategorie vybrané kapitoly. Každou z takto zobrazených kapitol je možné editovat nebo mazat. Garant samozřejmě může vytvářet také kategorie nové.

Správa šablon

Garant zobrazuje šablony, může vytvářet šablony nové, stávající mazat či editovat. Při editaci nastavuje počet bloků vybrané šablony, bloky dále edituje a vybírá pro ně otázky.

Správa kapitol

Kapitoly a následně jejich podkapitoly a podkapitoly podkapitol garant zobrazuje, může je upravovat, mazat, zobrazovat a přiřazovat k jednotlivým podkapitolám materiály, přiřazené materiály stahovat, zobrazovat a také odstraňovat.

Pokud jsou k některé kapitole přiřazeny cvičné testy, lze je zde zobrazit a vypracovat.

Aktuality

U aktualit je garant kromě jejich zobrazování, stahování a zobrazování jejich příloh také schopen nové aktuality vytvářet, mazat, upravovat a přidávat k nim nové přílohy nebo existující přílohy odebírat.

E-mailové zprávy

Uživatel v roli garanta zobrazuje odeslané e-mailové zprávy, kontroluje jejich status (jestli byly korektně odeslány), může zobrazovat jejich obsah s dalšími informacemi, záznamy o odeslaných e-mailech mazat a nové e-mailové zprávy vytvářet. Novou e-mailovou zprávu může garant odeslat buď vybraným třídám, nebo studentům vybrané třídy, které lze filtrovat podle jejich příjmení či loginu.

V každé nové e-mailové zprávě lze zadat předmět, text e-mailové zprávy a libovolný počet příloh.

Konzultační hodiny

Konzultační hodiny garant zobrazuje jako seznam konzultačních hodin, kde lze jednotlivé hodiny editovat, mazat nebo vytvářet nové.

Obdobně jako seznam konzultačních hodin je možné zobrazit historii konzultačních hodin, kde lze navíc například pro periodicky opakované konzultační hodiny zobrazit jednotlivé hodiny. Pro všechny konzultační hodiny má garant možnost nechat si vypsat studenty na jednotlivé konzultační hodiny zapsané. Vybraným nebo všem studentům z obsazených konzultačních hodin je pak možné odeslat e-mailovou zprávu.

Garantovi je také umožněno zobrazit si nejbližší obsazené konzultační hodiny se stejnými možnostmi, s jakými je možné zobrazit historii konzultačních hodin.

Logování

Garantovi je umožněno zobrazovat logy aplikace pro zadaný login a vybrané časové období. Jedná se o logy přístupů, logy online testů a logy všech akcí.

Nahlásit chybu

Nahlášení chyby probíhá analogickým způsobem jako u uživatele v roli studenta.

Profil

Případy užití pro operace týkající se profilu jsou stejné jako u uživatele v roli studenta.

Tutor:

Zavedení podmínek kurzu

Pokud je uživatel přihlášen jako tutor, pak může v aplikaci přidávat skupiny aktivit. Tutor mění minimum/maximum kurzu a zobrazuje celkový souhrn výsledků kurzu. V tomto souhrnu se zobrazují studenti daného kurzu a u jednotlivých studentů jejich jednotlivé výsledky. Každý výsledek lze upřednostnit a zjistit výsledky za jednotlivé skupiny aktivit.

Tutor zobrazuje skupiny aktivit zvlášť pro prezenční a zvlášť pro kombinovanou formu studia. Pro každou skupinu aktivit má tutor možnost vypsat souhrn jejích výsledků, kde lze pro jednotlivé studenty vypsat všechny jejich výsledky a vybrané výsledky upřednostnit.

Správa testů a aktivit

U správy testů a aktivit má uživatel v roli tutora stejné možnosti jako uživatel v roli garanta.

Správa otázek

Tutor zobrazuje otázky podle vybrané kapitoly, dále může blíže specifikovat kategorii a zařazení testů (Hlavní, Cvičný nebo Ukázkový), ve kterých se otázka objevuje. Otázky lze také zobrazovat pouze nepřekontrolované, pouze překontrolované nebo všechny. Je možné vytvářet otázky nové a existující mazat.

Seznam tříd a asociace se studenty

Tutor má možnost vytvořit novou třídu jak pro prezenční, tak pro kombinovanou formu studia. Vytváření tříd probíhá stejným způsobem jako u garanta. Tutor také může měnit časové omezení pro změnu tříd.

Ke každé třídě lze přiřadit studenty, jednotlivé studenty ze třídy odebrat, zobrazit všechny studenty dané třídy nebo studentům odeslat hromadnou e-mailovou zprávu.

Správa skupin aktivit

Správa skupin aktivit je u uživatele v roli tutora pouze jiný název pro Zavedení podmínek kurzu.

Správa kategorií

Uživateli v roli tutora je umožněno zobrazovat kategorie vybrané kapitoly.

Správa šablon

U šablon má uživatel v roli tutora stejné možnosti jako uživatel v roli garanta.

Správa kapitol

Kapitoly a následně jejich podkapitoly a podkapitoly podkapitol tutor zobrazuje, zobrazuje také u jednotlivých podkapitol materiály, přiřazené materiály může stahovat, zobrazovat a také odstraňovat.

Pokud jsou k některé kapitole přiřazeny cvičné testy, tak je možné je zde zobrazit a vypracovat.

Aktuality

U aktualit má uživatel v roli tutora stejné možnosti jako uživatel v roli garanta.

E-mailové zprávy

U e-mailových zpráv má uživatel v roli tutora stejné možnosti jako uživatel v roli garanta.

Konzultační hodiny

U konzultačních hodin má uživatel v roli tutora stejné možnosti jako uživatel v roli garanta.

Logování

U logování má uživatel v roli tutora stejné možnosti jako uživatel v roli garanta.

Nahlásit chybu

Nahlášení chyby probíhá analogickým způsobem jako u uživatele v roli studenta.

Profil

Případy užití pro operace týkající se profilu jsou stejné jako u uživatele v roli studenta.

2.3 Kvalitativní (nefunkční) požadavky

Verze androidu

Požadavek na minimální verzi androidu v koncovém zařízení byla vedoucím práce stanovena na Android ve verzi 4.1, což odpovídá verzi API 16 a vyšší. V současné době (14. 4. 2016) je díky tomuto kritériu podporováno přibližně 94,8 % všech zařízení se systémem android, což je přibližně o 10 % více, než tomu bylo v polovině roku 2015.

Aplikace je psána pro minimální verzi API 14 a pokrývá tím pádem v současnosti (14. 4. 2016) přibližně 97,3 % všech zařízení se systémem android. Nižší verze než požadovaná byla zvolena proto, že obsahuje veškeré rozhraní použité v implementaci. Informace o přibližném počtu zařízení spolu se závislostmi verze API na verzi androidu lze vyčíst také z výřezu snímku obrazovky programu Android Studio, který je přiložen v příloze na obrázku 4.

Verze operačního systému mají tu vlastnost, že ta novější podporuje rozhraní verzí nižších. Výjimku tvoří funkce označené jako „deprecated“, které nejsou doporučeny z důvodu bezpečnosti nebo že jejich použitelnost v novější verzi již není doporučena.

Požadavky na koncové zařízení

Aplikace bude mít bezproblémový chod na zařízeních, která jsou definována níže. Obecně však pro systém android platí, že zařízení s vyšším rozlišením displeje, větší pamětí RAM a do této doby (2016) i novější verzí OS Android umožňuje chod aplikací napsaných pro zařízení s nižšími HW a SW nároky. Testování tedy bude probíhat i na slabších zařízeních s nižším rozlišením, než je požadováno, aby byl bezproblémový chod skutečně garantován. Na zařízeních s nižším rozlišením, než je uvedeno jako minimální pro vybrané zařízení, nemusí být viditelné všechny prvky uživatelského rozhraní.

Mobilní telefon

Aplikace bude používána na mobilních telefonech v roli studenta. Minimální požadavky na takovéto zařízení byly vedoucím práce stanoveny následujícím způsobem: operační systém android ve verzi 4.1 nebo novější, 2 GB paměti RAM a FullHD rozlišení displeje.

Aplikace bude testována v průběhu vývoje na mobilních telefonech s konfiguracemi:

- Xiaomi MI2S (Android 4.1, 2GB RAM, HD rozlišení)
- Nexus 4 – emulátor Android Studia (Android 4.1, 1.5 GB RAM, 1280x768)
- Xiaomi MI2S (Android 4.4, 2GB RAM, HD rozlišení)
- Xiaomi MI2S (Android 5.0, 2GB RAM, HD rozlišení)
- Xiaomi MI2S (Android 5.1, 2GB RAM, HD rozlišení)
- LG Optimus 2X (Android 4.4, 512MB RAM, WVGA rozlišení)
- Nexus 6 (Android 5.1, 3GB RAM, rozlišení 2560x1440)
- Nexus 5X – emulátor Android Studia (Android 6.0, 1.5GB RAM, 1920x1080)
- Xiaomi Mi5 (Android 6.0, 3GB RAM, FullHD rozlišení)
- ARC doplněk pro prohlížeč Chrome [42]

Tablet

Aplikace v roli garanta a tutora je zamýšlena pro tablety. Požadovaný operační systém pro bezproblémový chod je android 4.2 nebo novější, minimální velikost paměti jako u mobilního telefonu a rozlišení displeje také FullHD. Velikost úhlopříčky displeje tabletu byla s vedoucím diplomové práce diskutována a následně byla určena doporučená hodnota 10 palců. Při splnění těchto podmínek je aplikace snadno ovladatelná, použitelná a všechny texty jsou dobře čitelné.

Testovanými zařízeními v průběhu vývoje jsou:

- Lenovo Yoga Tablet 2-10 (Android 5.0, 2GB RAM, rozlišení 1920x1200, 10 palců)
- Nexus 9 – emulátor Android Studia (Android N, 1.5GB RAM, 2048x1536, 9 palců)
- Nexus 5X v režimu na šířku emulovaný Android Studiem (Android 6.0, 1.5GB RAM, 1920x1080, 5 palců)
- Nexus 10 (Android 5.0, 2GB RAM, 2560x1600, 10 palců)
- ARC doplněk pro prohlížeč Chrome [42]

Jazykové mutace

Výchozím jazykem aplikace bude angličtina. Dalším jazykem je čeština a aplikace bude připravena na přidávání dalších jazyků v případě potřeby.

Bezpečnost testů uložených pro offline režim

Uložené testy pro offline vypracování budou v zařízení šifrovány.

3 Návrh rozhraní pro komunikaci se serverem systému eLogika

3.1 Systém eLogika

Systém eLogika je LMS systém - redakční systém určený pro e-learning.

E-learningem rozumíme studium s podporou elektrotechnických zařízení a internetu. [1]

LMS (Learning Management System) neboli řídicí výukový systém, je systém, který umožňuje e-learningové vzdělávání. Slouží ke kontrole výuky v průběhu semestru, obsahuje výukové kurzy a pro ně pak podpůrné materiály, umožňuje udělování a zobrazování hodnocení a je složen ze spousty dalších funkcí usnadňujících výuku a zlepšujících její kvalitu za přispění informačních a komunikačních technologií.

Výhodou systému eLogika jakožto e-learningové aplikace (oproti klasickým výukovým metodám) je bezesporu přístup k výukovým materiálům skrze internet. Oproti běžným internetovým stránkám může být přístup k materiálům omezen například pouze studentům daného kurzu, což ocení především pedagogové. Další výhodou eLogiky jistě je, že učivo lze rozdělit do kapitol s cvičnými testy, což umožňuje studentům vzdělávat se a procvičovat si probíranou látku v libovolném čase. V rámci systému eLogika je možné studentům předávat důležité informace a studenti ví, kde takové informace naleznou. Jistě jako výhody můžeme dále zmínit možnost vypracování testu v některém z předem zadaných termínů bez nutnosti fyzické účasti na konkrétním místě, možnost odevzdání úkolů, přihlašování na různé termíny, přihlašování se na konzultační hodiny a další činnosti spojené s výukou. Vše je soustředěno na jednom místě přístupném skrze internet.

Další velkou výhodou systému eLogika, která nemusí být na první pohled zjevná, je možnost sběru dat o uživatelích. Lze sledovat, co je pro studenty hůře zvládatelné, a co naopak snadné. Toto lze vypozorovat ze záznamů o studentech a jejich používání systému eLogika. To, jak studenti zobrazují studijní materiály, vypracovávají testy, jaké mají výsledky a další statistické údaje, slouží ke zpětné vazbě probíhajících kurzů a vede k přizpůsobení výuky studentům, a to tak, že se zaměří na části, které pro ně byly hůře zvládatelné. Z takto získaných dat se vytváří zpětná vazba studentům, kteří se tak dozví, kde nejvíce chybují a na kterou část učiva se mají zaměřit.

Systém eLogika má také své nevýhody, kterými jsou například počáteční investice do implementace a potřebného vybavení, závislost na technice, nevýhodnost průběhu výuky pro některé studenty vyžadující individuální přístup (například zrakově postižení studenti), nutnost seznámení se s novým systémem jak studenty, tak i pedagogy. [15]

Aplikace eLogika pro Android spadá do m-learningu. Písmeno „m“ zastupuje mobilní zařízení a otvírá tak možnosti e-learningového studia bez nutnosti neustálého přístupu k internetu. Snižuje se tak datový přenos pro klientskou i serverovou část a pro uživatele odpadá při některých činnostech nutnost internetového připojení. V ničem jiném se m-learning oproti e-learningu zásadně neliší. [16]

3.2 REST API

REST (Representational State Transfer) je komunikační architektonický styl, který je charakterizován jednotným a snadným přístupem k datům skrze HTTP, případně HTTPS protokol. Komunikace probíhá mezi serverem poskytujícím rozhraní REST API a klientem, který využívá služeb serveru. Tato komunikace je bezestavová, takže se jedná pouze o dotazy ze strany klienta a odpovědi na dotazy ze strany serveru. REST umožňuje využít mezipaměť (cache) pro uchování odpovědí serveru. Data nemusí být získána přímo serverovou částí aplikace, ale serverová část může sloužit jen jako prostředník mezi poskytovatelem dat a klientem. REST volitelně umožňuje serveru zaslat kód klientovi, který kód vykoná. Tím je přesunuta část logiky ze serveru na klienta. [17]

Pro komunikaci mezi serverovou a klientskou částí aplikace eLogika se využívá formát JSON, což je datově úsporný formát snadno strojově zpracovatelný zrovna tak, jako je i dobře čitelný lidmi. Jedná se o defaultní jazyk architektury REST.

Formát JSON může být dvojího typu. Buď se jedná o datovou strukturu typu slovník – tedy seznam dvojic, kdy první prvek ze dvojice je klíč a druhý hodnota, nebo se jedná o seznam hodnot – pole, vektor, seznam. [18]

V rámci HTTP nebo HTTPS komunikace se uvnitř aplikace eLogika používají metoda GET k získávání dat ze serveru a metoda POST k odesílání dat zpět na server. Dokumentace všech metod použitých pro komunikaci se serverem eLogika lze nalézt v příloze J.

Celý systém eLogika je zachycen na diagramu nasazení v příloze H. WCF (Windows Communication Foundation) je framework umožňující tvorbu servisně orientovaných aplikací. WCF komunikuje s klientem webové aplikace a částečně i s Windows 8.1 a Windows Phone 8.1 skrze zprávy ve formátu XML – skrze protokol SOAP.

3.3 Metody navržené pro komunikaci se serverem systému eLogika

Aplikace běžící v prostředí OS Android komunikuje se serverovou částí systému eLogika výhradně skrze REST API – více vizte: 3.2. Dokumentace všech metod použitých v rámci této komunikace se nachází v příloze J a jejich použití v rámci kontextu aplikace je popsáno v kapitole 4.3.

Adresa pro komunikaci se serverem sytému eLogika může probíhat dvěma způsoby – buď skrze HTTP protokol, adresu serveru http://elogika.vsb.cz (http://158.196.141.100) a port 8090, nebo skrze šifrovaný HTTPS protokol, adresu serveru https://elogika.vsb.cz (https://158.196.141.100) a port 23590. Ve výchozím stavu se využívá komunikace přes HTTPS a případnou změnu je nutné provést ve zdrojovém kódu.

Metody uvedené v příloze J volané za účelem komunikace aplikace a její serverové části se konkatenují (toto slovo neznám…) s výše uvedenou adresou pro komunikaci (včetně použitého protokolu a příslušného portu), řetězcem „/api“ a sekce, ve které se volaná metoda nachází. Metoda je volána s příslušnými parametry podle použité metody komunikačního protokolu. Kompletní adresa volané metody tedy vypadá například takto:

https://elogika.vsb.cz/api/CourseConditions/currentaktivita?courseInfoId={courseInfoId}&userId={userId}

Informace o tom, že se jedná o GET metodu, je předána knihovně Volley, skrze kterou komunikace probíhá. Více informací o komunikaci skrze tuto knihovnu lze nalézt v kapitole 3.4. Server pro tento dotaz vrací pole navržených JSON objektů s následujícími parametry:

- "Smazana" – příznak značící, jestli je aktivita smazaná
- "NazevSA" – název skupiny aktivit
- "Nazev" – název aktivity
- "Maximum" – maximum bodů z aktivity
- "Povinny" – příznak, jestli je aktivita povinná
- "NazevTermin" – název termínu, ve kterém se aktivita nachází
- "AktivniOD" – čas, od kdy je aktivita aktivní
- "AktivniDO" – čas, do kdy je aktivita aktivní
- "Popis" – slovní popis dané aktivity
- "VzorVyslSouborNazev" – název souboru se vzorovým výsledkem aktivity
- "PopisSouborNazev" – název souboru pro popis aktivity
- "VzorVyslSouborContType" – typ souboru se vzorovým výsledkem aktivity
- "PopisSouborContType" – typ souboru pro popis aktivity
- "VyberDoData" - datum a čas, do kdy je možné si aktivitu vybrat
- "Vybiratelnost" – příznak možnosti výběru aktivity
- "IsEditable" – příznak značící, jestli je aktivita dále editovatelná
- "IDAktivita" – id aktivity
- "IDTermin" – id termínu
- "PocetPokusu" – počet pokusů na vypracování dané aktivity
- "VzorVysl" – slovní popis vzorového výsledku aktivity
- "DoporHodnoceni" – doporučené hodnocení aktivity
- "Minimum" – minimum bodů potřebných k úspěšnému zvládnutí aktivity
- "OmezeniStudentu" – maximální počet studentů, kteří mohou aktivitu vypracovat
- "TypOmezeni" – „OmezeniStudentu“ se může týkat pouze třídy – hodnota „0“, nebo celého kurzu – hodnota „1

Pro získání informací potřebných k úplnému zobrazení všech aktuálně vypsaných aktivit je potřeba ještě zavolat další metody, a to z důvodu snížení počtu přenášených dat ze serveru na klienta. Data obsahují soubory spojené s jednotlivými aktivitami. V případě, že je potřeba získat soubor se vzorovým výsledkem vybrané aktivity, zavolá se metoda J 23d). Jestliže uživatel aplikace požaduje stažení a otevření souboru s popisem vybrané aktivity, v aplikaci se nejprve zavolá metoda J 23e) a následně server vrátí odpověď s daty, která se uživateli zobrazí. Každý zobrazený soubor v rámci aplikace eLogika je zároveň stažen do zařízení s informací, že se tomu tak stalo, takže uživatel nemusí tato objemná data stahovat vícekrát než jednou.

Jako příklad navržených POST metod uvedu metody pro odesílání e-mailových zpráv. Tyto zprávy lze odeslat všem studentům vybraného kurzu, vybraným třídám, vybraným studentům zvolené třídy, vybraným z přihlášených studentů na zvolený termín (aktivity, skupiny aktivit nebo testu), vybraným přihlášeným studentům na některou z periodicky opakovaných konzultačních hodin nebo vybraným studentům z těch, kteří se přihlásili skrze systém eLogika na některou konkrétní vypsanou konzultační hodinu.

K samotnému odesílání e-mailových zpráv byly navrženy dvě metody. Metoda J 33c) pro odesílání e-mailových zpráv na vybrané e-mailové adresy a metoda J 33d), která namísto e-mailových adres používá pole s identifikátory tříd.

E-mailové zprávy by se všechny daly odesílat pomocí metody J 33c), ale v rámci snížení objemu přenesených dat mezi serverovou částí aplikace eLogika a koncovým zařízením s OS Android byla navržena metoda druhá - J 33d). Tím se zamezí získávání e-mailových zpráv pro všechny vybrané třídy, což by znamenalo vytvoření nové metody na straně serveru, která by pro vybrané identifikátory tříd vrátila všechny e-mailové adresy studentů, kteří do těchto tříd patří, nebo by to znamenalo opakované volání serveru (pro jednotlivé identifikátory tříd) a získávání e-mailových zpráv postupně. Získávání e-mailových adres se tedy děje na straně serveru a dále se vykonávání těchto dvou výše zmíněných metod na serverové straně nemění.

K odesílání e-mailových zpráv všem studentům v rámci kurzu byl zvolen jiný přístup z výše navrhovaných. Byla navržena metoda J 1z), která pro zadaný kurz vrátí e-mailové adresy všech studentů kurzu a na takto získané adresy je pak odeslán e-mail za asistence metody J 33c). Tento přístup byl zvolen proto, že tu není tak nutná úspora dat ze dvou důvodů. Prvním je ten, že pokud bychom chtěli odeslat e-mailovou zprávu všem studentům kurzu v jedné metodě, můžeme to udělat odesláním e-mailové zprávy pomocí metody J 33d) – tedy všem třídám v rámci vybraného kurzu. Druhým důvodem je ten, že odesílání zpráv celému kurzu není tak častou operací, neboť zahrnuje studenty jak prezenční, tak kombinované formy. Odesílání zpráv všem studentům kurzu jedné z forem studia je realizováno odesláním e-mailové zprávy na všechny třídy dané formy.

Odesílání zpráv vybraným studentům zvolené třídy, vybraným z přihlášených studentů na zvolený termín (aktivity, skupiny aktivit nebo testu), vybraným přihlášeným studentům na některou z periodicky opakovaných konzultačních hodin nebo vybraným studentům z těch, kteří se přihlásili skrze systém eLogika na některou konkrétní vypsanou konzultační hodinu, probíhá tak, že jsou nejprve získány jejich e-mailové adresy a ty jsou pak použity k volání metody J 33c). V čase odesílání zprávy takovým studentům jsou již v rámci aplikace jejich e-mailové adresy známy z jiných metod, které sloužily k získání jednotlivých studentů a k možnosti vybrat si z nich ty, kterým bude e-mailová zpráva odeslána.

Metody J 33c) a J 33d), pokud vše proběhne tak, jak má, vrací odpověď ve tvaru slovníkového typu – tedy ve tvaru klíč: hodnota – `{„Result": „true"}`. Pokud z nějakého důvodu metoda na straně serveru neproběhne korektně, tak server odešle odpověď `{„Result": „false"}`. Uživatel je o úspěšnosti volané metody informován.

3.4 Volley

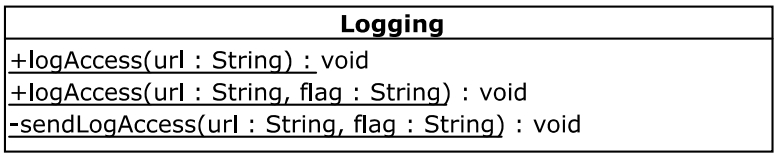
Volley je knihovna OS Android, která se stará o síťovou komunikaci mezi klientem a serverem. Obsahuje plánovač síťových požadavků, vytváří asynchronní spojení mezi klientskou a serverovou částí aplikace eLogika, zajišťuje paměťovou koherenci (stejná data v různých pamětech zařízení), podporuje prioritizaci dotazů na server, umožňuje zrušení dotazu na server, obsahuje nástroje k ladění aplikací a je snadno přizpůsobitelná aplikaci, která ji využívá. Knihovna již obsahuje implementaci pro zpracování odpovědí serverové části v podobě řetězce, obrázku, JSON objektu a JSON pole. [22]

Jak byla knihovna Volley přizpůsobena pro požadovanou práci se systémem eLogika, je možné vyčíst v diagramu na obrázku 15 v příloze B. Jakým způsobem je knihovna Volley v aplikaci využívána, ukazuje diagram na obrázku 31, který je umístěn v příloze G.

3.5 Logování aplikace

Aplikace je logována stejně jako aplikace webová. Loguje se přihlašování, výběr školy a role, zobrazení profilu, veřejné kury, aktuality, přihlášení se na termín, vše, co se týká testu, aktivit včetně jejich správy a téměř všechny akce provedené v aplikaci jsou doprovázeny logováním.

K logování se využívá metoda popsaná v dokumentaci J 1r) a jednoduchý třídní diagram třídy Logging, která se o logování stará, je k nahlédnutí na obrázku 1.



Obrázek 1: *Třída Logging, která se zabývá logováním všech akcí v rámci aplikace*

Metoda logAccess je volána z různých míst aplikace s parametrem url, který pro zobrazené okno aplikace vkládá url adresu, která odpovídá url adrese webové aplikace, aby bylo možné provádět další operace s logy jednotně pro všechna zařízení v rámci eLogiky. Před samotným voláním metody REST API - J 1r) umístěné v metodě sendLogAccess třídy Logging se vkládá parametr „OS“ značící, že se jedná o operační systém Android, což odděluje aplikaci eLogika pro android od dalších aplikací eLogika. „Flag“ obsahuje dodatečné informace k url adrese – pokud je okno webové aplikace rozděleno do více oken v rámci mobilní aplikace. Další přidávané parametry jsou popsány v dokumentaci u metody J 1r).

4 Implementace

4.1 Prostředí pro vývoj

Pro vývoj aplikace bylo využito Android Studio ve verzích 0.8.0 – 2.0.0. Android studio lze stáhnout zde: [2]. Bylo zajímavé sledovat vývoj Android studia. V prvních zkoušených verzích byl software značně nestabilní a obsahoval mnoho chyb. Postupem času se stabilita zvyšovala a přibývaly nové funkce. V poslední verzi – tedy ve verzi 2.0.0 je například velmi zajímavá nová funkce s názvem „Instant Run“. Při vývoji se pro otestování změn provedených v kódu vytvořil instalační balíček, ten se nahrál do zařízení, na kterém testování probíhalo, pokud tam aplikace již běžela, tak byla restartována. Nyní, s funkcí „Instant Run“, již není nutné, aby se aplikace vždy po překladu restartovala, ale restartuje se pouze právě spuštěná aktivita. Překlad a spuštění aplikace na testovaném zařízení také prošly viditelným zrychlením.

Pro testování byly použity jak fyzické přístroje, tak emulátory. Standardní emulátor v rámci android studia byl velmi pomalý. Situaci sice vylepšoval hardwarově podporovaný virtualizovaný engine pro procesory Intel (stažitelný zde: [21]), ale v rané fázi vývoje byla dána přednost emulátoru od firmy Genymotion (stažitelný zde: [20]), který vykazoval vyšší výkon a nižší nároky na HW. V posledních verzích Android studia je již standardní emulátor dostatečně výkonný, a tak není třeba využívat SW třetích stran.

Vývoj probíhal pouze v OS Windows, a to konkrétně ve verzích Windows 7 – Windows 10, postupně na noteboocích Acer Aspire 5741G, HP G62 a Lenovo Z50-70.

Na správu zdrojového kódu byl použit program SourceTree (stažitelný zde: [19]). Ukázka použití SourcTree v rámci vývoje aplikace eLogika pro android je v příloze na obrázku 6. Více o správě zdrojového kódu se můžete dočíst v kapitole věnující se TFS.

4.2 Způsob vedení vývoje

Metody pro vývoj aplikací

Metody můžeme dělit do 2 hlavních kategorií, a to rigorózní a agilní metodiky.

Rigorózní metodiky

Rigorózní metodiky se vyznačují názorem, že při vývoji softwaru ??? lze popsat všechny procesy, celý vývoj popsat, plánovat, řídit a měřit. Všechny požadavky jsou známy a dobře definovány před začátkem vývoje. Jedná se spíše o metodiky používané pro velké projekty. Buď mohou být založeny na vodopádovém vývoji, nebo na vývoji iterativním. Jako příklady můžeme uvést například OPEN, Rational Unified Process nebo Enterprise Unified Process. [39]

Agilní metodiky

Agilní metodiky se soustředí na podporu rychlého vývoje, připravenost na změny, velmi podporují (osobní) komunikaci se zákazníkem i mezi vývojáři. Dávají přednost dobře fungujícímu softwaru před dobrou dokumentací. Na návrhu se pracuje kontinuálně i během

vývoje. Jako příklady můžeme uvést Feature-Driven Development, Dynamic Systems Development Method a Extrémní programování nebo Scrum.

Metoda použitá při vývoji aplikace eLogika

V rámci vývoje aplikace eLogika byla použita metoda Scrum, a to v době, kdy souběžně probíhal vývoj pro platformu Windows Pohone 8.1, Windows 8.1, iOS. S těmito mobilními zařízeními také zároveň probíhaly úpravy serverové části, a to především z důvodu realizace požadavků na rozhraní pro komunikaci s jednotlivými mobilními zařízeními, byly opravovány chyby nalezené při vývoji aplikací pro mobilní zařízení a také byly na server implementovány nové požadavky na funkčnost celého systému eLogika.

Scrum meetingy – tedy pravidelné porady – probíhaly přibližně na týdenní bázi. Meetingy svolával vedoucí vývoje Ing. Martin Prokeš a účastnili se jich vývojáři mobilních aplikací i správce serveru, a to buď osobně v prostorách Vysoké školy báňské – Technické univerzity Ostrava, nebo online přes program Skype. Komunikace mezi vývojáři a správcem serveru probíhala buď skrze e-mailové zprávy, program Skype, nebo systém TFS popsaný v následující kapitole. Protože se meetingy nekonaly denně, jak je u Scrum metody zvykem, o to déle trvaly. Jednalo se přibližně o 1 – 2 hodinové meetingy. V rámci setkání probíhalo zadávání některých konkrétních činností do systému TFS. Největší prostor byl však věnován komunikaci mezi vývojáři mobilních zařízení a správcem serveru, který zodpovídal nejasnosti spojené s funkčností serveru nebo mu byly vysvětlovány požadavky na komunikaci, které se mohly poupravit tak, aby byl návrh vhodný pro všechny mobilní platformy.

Úpravy serverové části probíhaly nejdříve na testovacím serveru a až po jejich otestování byly změny přeneseny i na „ostrý“ server, na kterém již běžela webová aplikace používaná například pro předmět Matematická logika.

Sprinty – tedy ukázky spustitelného softwaru s ukázkou postupu ve vývoji – probíhaly přibližně jednou měsíčně, a to buď na fyzickém zařízení, nebo prostřednictvím emulátoru v rámci sdílení obrazovky programu Skype.

TFS

TFS neboli Team Foundation Server je sada nástrojů určená ke spolupráci na projektech. TFS umožňuje správu verzí kódu, čehož bylo využíváno za asistence programu SourceTree stažitelného zde: [19]. TFS byl také využíván k vytváření úkolů a označení stavu, v jakém se nachází. Chyby serverové části byly také nahlašovány skrze TFS, na který byl po přihlášení přístup skrze url adresu http://elogika.vsb.cz:8080/tfs/eLogika%20Collection/Android/.

O TFS se můžete dozvědět více zde: [38], kde je možné také sadu nástrojů získat. Ukázka použití tohoto nástroje v rámci vývoje aplikace eLogika pro android je zachycena na obrázku 5 v příloze.

4.3 Implementace jednotlivých případů užití

Nepřihlášený uživatel

Veřejné kurzy

Metoda J 13a) je volána pro získání všech veřejných kurzů, které jsou následně zobrazeny do RecyclerView – pro více informací o této komponentě vizte: 4.7. Pokud při registraci na veřejný kurz nebyl zadán login, tak je potřeba jej vygenerovat. K tomu slouží metoda J 1i). Následně je volána metoda J 1g), která vytvoří uživateli účet v systému eLogika.

Student

Přihlášení

Po spuštění aplikace, pokud se uživatel nechce zapsat do veřejného kurzu, je nutné, aby vyplnil své přihlašovací údaje - svůj login a heslo do systému eLogika.

Nejdříve se uvnitř aplikace zavolá metoda J 1c) pro získání seznamu solí přihlašovaného uživatele. Heslo je konkatenováno první ze získaných solí a pomocí hašovací funkce je tento řetězec převeden algoritmem MD5 na otisk o velikosti 128 bitů. Výsledný řetězec je spolu se zadaným loginem předán metodě J 1b), která buď vrátí údaje přihlašovaného uživatele, nebo vrátí informaci o neúspěšném přihlášení. Pokud je přihlášení neúspěšné, tak se v případě existence použije další sůl v pořadí a spolu s heslem se opět zahašuje a proces se opakuje do té doby, dokud se nevyzkouší všechny soli. Přihlášení, které je s každou získanou solí neúspěšné, je neúspěšné.

Po úspěšném přihlášení se uživatelské jméno konkatenuje se solí pro uživatelské jméno, která je uložena v aplikaci, a následně se výsledný řetězec za pomoci šifry sha1 převede na haš. Totéž se provede s konkatenací haše pro heslo uložené v aplikaci s heslem a jménem. Spolu s těmito dvěma haši jsou id uživatele, uživatelské jméno a příjmení uloženy do úložiště aplikace pomocí SharedPreferences - vizte: 4.7, pokud se zde již nenachází.

V dalším kroku jsou pomocí metody J 2a) načteny všechny školy, které jsou pro uživatele dostupné. Dále aplikace zavolá metodu J 3a) se schoolId z první načtené školy v předchozím kroku.

Aktuality

Pokud je uživatel v roli studenta, tak aktuality může pouze zobrazit. Aktuality jsou získány metodou J 10a) a zobrazeny pomocí komponenty RecyclerView a šablony recycler_view_with_button_swipe – vizte: 4.9. Přílohy pro jednotlivé aktuality jsou získány pomocí metody J 10b) a zobrazeny u příslušných aktualit.

Přihlásit se na termín

Metodou J 8a) jsou získána data potřebná pro zobrazení skupin aktivit, aktivit a testů. Na základě parametru „typ„ jsou data roztříděna do těchto 3 skupin, které v aplikaci tvoří 3 záložky komponenty ViewPager - vizte: 4.7 se šablonou list_view_swipe – vizte: 4.9.

Jednotlivé termíny jsou zobrazeny pod sebou v ListView - vizte: 4.7 a po jejich vybrání vyskočí okno s detaily konkrétního termínu. V těchto detailech se lze přihlásit na termín, nebo se z termínu odhlásit pomocí příslušného tlačítka. Toto tlačítko je však aktivní pouze v případě, že se aktuální datum a čas nachází pro případ možnosti přihlášení v rozmezí „přihlášení od“ - „přihlášení do“, anebo pro možnost odhlášení je tlačítko aktivní pouze před datem a časem „odhlášení do“. Stejná funkcionalita je také v seznamu termínů při použití postranního zaškrtávacího políčka. Zaškrtnuté políčko indikuje přihlášení na termín.

Vypracovat test

Po zavolání metody J 8h) se do ListView zobrazí seznam dostupných testů vybraného studenta. Pro každý test se v detailech zobrazí tlačítko pro vygenerování testu. Nejdříve je zavolána metoda J 11j), která na serveru zjistí, jestli je možné test vygenerovat. Vygenerování testů pak probíhá tím způsobem, že je zavolána metoda J 11d). Po vygenerování testu server odešle zpět aplikaci identifikátor vygenerovaného testu a je zavolána metoda J 11c) pro vygenerování otázek. Texty otázek jsou získány z metody J 11c) a texty odpovědí z metody J 11b). Při možnosti uložit test offline se informace o testu uloží do SQLite databáze android aplikace.

Pro zobrazení otázek se využívá komponenta ViewPager. Statistiky při vypracovávání testu se ukládají díky metodě J 18a) a J 18b). Kompletní statistiky testu se pak odesílají v těle metody J 8e). Odpovědi celého testu se ukládají za asistence metody J 11e).

Offline testy

Offline testy jsou implementovány analogickým způsobem, jaký je popsán v kapitole Vypracovat test. Rozdíl je v tom, že místo se serverem se pracuje s databází. Všechna data pro test byla vygenerována a zbývá už jen test spustit a na to není připojení k internetu potřeba. Po vypracování testu se všechna data (statistiky a výsledky) opět ukládají do databáze.

Vrácení pokusu probíhá zavoláním metody J 11f) a po jejím úspěšném provedení smazáním testu z databáze.

Odeslání offline testů probíhá také analogickým způsobem jako odeslání testů vypracovaných online. Zavolá se metoda J 11e), a to v případě, že uživatel zvolí možnost pro odeslání testů po přihlášení v záložce „Vypracované offline testy“ komponenty ViewPager. Další záložka obsahuje ListView s testy, které jsou staženy v zařízení (SQLite databázi) pro možnost vypracování.

K vypracování offline testů musí být uživatel přihlášen offline s přihlašovacími údaji, se kterými se již na zařízení přihlašoval a ukládal na něm test pro offline vypracování.

Vypracované testy

V ListView jsou vypsány jednotlivé vypracované testy získané pomocí metody J 12a).

V detailech testu (dialogovém okně) je zobrazeno tlačítko pro zobrazení testu, pokud test touto možností disponuje. Jestliže se jedná o vypracovaný papírový test, tak s využitím metody J

11h) je stažen papírový test a zobrazen uživateli. Nahlášení špatně rozpoznaného testu proběhne po zavolání metody J 11i). Pro zobrazení výsledků online testu je využito metody J 15a) a J 11g).

Hodnocení kurzu

Zavolaná metoda J 14a) nejprve vrátí data pro záložku komponenty ViewPager, kde jsou v ListView zobrazena celková hodnocení jednotlivých skupin aktivit.

Pro každou skupinu aktivit je vytvořena nová záložka v komponentě ViewPager s jednotlivými výsledky, které jsou získány metodou J 20b).

Záložky jednotlivých skupin aktivit obsahují vždy TableLayout - více vizte: 4.7 se známými údaji z celkového hodnocení aktivit, následován ListView s jednotlivými získanými výsledky ve vybrané skupině aktivit. Výsledky, ze kterých je tvořeno výsledné hodnocení skupiny aktivit, jsou zvýrazněny. Pro takové výsledky jsou rovněž zobrazeny informace ohledně pořadí a celkového počtu výsledků jak ve třídě, tak i v ročníku. Tyto informace jsou dopočítány v aplikaci, jakmile jsou dostupná všechna potřebná data. Dále je nutné volat metodu J 20e), pro získání všech výsledků - tedy i výsledků jiných uživatelů než pouze právě přihlášeného.

Aktivity

Po vybrání záložky v menu pro zobrazení aktivit je na obrazovku vykreslena komponenta ViewPager, která obsahuje jednu záložku pro „aktuálně vypsané„ a jednu záložku pro „odevzdané a vypracované„ aktivity.

Aktuálně vypsané aktivity jsou uvnitř komponenty ListView. Po vybrání některé aktivity se zobrazí okno s detailními informacemi. V případě, že je přítomen popis aktivity v souboru, je zobrazeno tlačítko pro možnost jeho stažení. Soubor je pak získán pomocí metody J 23e). Obdobně je tomu s případným příkladem v souboru získaného metodou J 23d). Jestli byl soubor již do aktivity nahrán, je zjišťováno metodou J 7b). Pokud soubor existuje, tak k jeho získání slouží metoda J 7c). Aktivita se odevzdává pomocí metody J 7d), což je možné pouze v případě, že se aktuální čas nachází v rozmezí „aktivní od“ - „aktivní do“.

Odevzdané a vypracované aktivity jsou rovněž, jako aktuálně vypsané aktivity, zobrazeny uvnitř komponenty ListView. Po vybrání některé aktivity se zobrazí okno s jejími detailními informacemi, kde je možné získat již odevzdaný soubor metodou J 7c), změnit soubor a popis odevzdané aktivity zavoláním metody J 7d).

Rozvrh

Nejdříve je nutné získání semestrů pomocí metody J 5a). Pro každý takový semestr jsou metodou J 6a) získána data, která se zobrazí do komponenty ViewPager. Komponenta obsahuje 6 záložek - pro každý den kromě neděle jednu. Pokud je předmět pro kombinované studium, tak obsahuje datum, ze kterého se pomocí třídy TimeLogika- vizte: 4.8 zjistí, o který den v týdnu se jedná, a předmět může být zanesen do rozvrhu stejným způsobem jako předmět prezenční formy studia. Předměty jsou zobrazeny v komponentě RecyclerView - vizte: 4.7. Po vybrání předmětu je zobrazeno okno s jeho detaily. Pokud se jedná o vlastní záznam, tak jej lze z rozvrhu smazat za

použití metody J 6g). Vlastní záznam v rozvrhu je také možné upravit, k čemuž slouží metoda J 5a) spolu s metodou J 6f).

Pro zapsání předmětu do rozvrhu se zobrazí RecyclerView s dostupnými hodinami získanými metodou J 6b). Vedle nabízené hodiny je vždy zobrazeno zaškrtávací políčko. Pokud je políčko zaškrtnuto, je hodina zapsána. Pokud políčko zaškrtnuto není, tak se hodina v rozvrhu nenachází. Pokud je již kapacita hodiny naplněna, nelze si hodinu do rozvrhu zapsat. Při kliknutí na nezaškrtnuté tlačítko se objeví dialog s dotazem, jestli chce uživatel hodinu do rozvrhu zapsat. Při vybraném zaškrtnutém tlačítku je zobrazen dialog s dotazem, jestli chce uživatel hodinu z rozvrhu odepsat. Na základě volby uživatele je vykonaná požadovaná akce metodou J 6c) pro zapsání hodiny, případně metodou J 6d) pro odstranění hodiny z rozvrhu. Výsledek akce je ihned promítnut do rozvrhu.

Přidání vlastního předmětu probíhá tak, že musí být vyplněny všechny potřebné informace (název předmětu, zkratka předmětu, tutor a učebna) a následně se zavolá metoda J 5a) spolu s metodou J 6e).

Teorie

Pod položkou menu s názvem Teorie se nachází SeparatedListAdapter - více vizte: 4.8, který obsahuje ListView se seznamem kapitol, ListView se seznamem materiálů a ListView se seznamem cvičných testů. Kapitoly jsou získány ze serveru metodou J 17a). Pro vybranou kapitolu je zobrazen seznam materiálů získaných metodou J 17b). Seznam cvičných testů je získán metodou J 8g). Povinné testy jsou označeny červenou barvou.

Pro vybranou kapitolu se zobrazí seznam kapitol, načte se seznam materiálů a cvičných testů. Pomocí šipky ← ve Actionbaru - více vizte: 4.8 lze zobrazit kapitolu, která je nadřazenou kapitolou právě zobrazené kapitoly. Vedle šipky je v Actionbaru vyobrazena hierarchie kapitol až po právě zobrazenou. Lze v nich libovolný nadpis kapitoly vybrat a kapitola se pak společně s materiály a cvičnými testy zobrazí.

Při výběru položky v seznamu materiálů se za pomoci třídy DownloadFile - více vizte: 4.8 soubor stáhne do zařízení a zobrazí.

Pokud je vybrán cvičný test, tak je zobrazeno okno s podrobným popisem a možností test spustit. Jestli může být test vygenerován, je zjišťováno pomocí metody J 11j). Pokud již není žádný pokus, pak nelze test vygenerovat. Po spuštění testu se vše děje stejně jako při vypracování online testu - vizte: Vypracovat test

Konzultační hodiny

Konzultační hodiny jsou načítány do RecyclerView. Nejprve je potřeba pomocí metody J 9a) zjistit, na které termíny je student přihlášen. Poté se zavolá metoda J 9b) vracející seznam všech dostupných termínů. Po vybrání některého termínu konzultačních hodin je zobrazeno okno s detaily. Pokud je uživatel na termín přihlášený, tak se zde zobrazí důvod přihlášení, byl-li zadán.

Vedle každého termínu konzultace je zaškrtávací políčko indikující přihlášení. Pokud je tlačítko nezaškrtnuté, tak se lze na termín přihlásit. V opačném případě je možné se z termínu

konzultace odhlásit. Při přihlašování je možné zadat důvod přihlášení a vše se provede pomocí metody J 9d). Podle návratové hodnoty metody se pozná, jestli proběhlo přihlášení v pořádku, anebo jestli již byla naplněna kapacita pro přihlášení na termín konzultační hodiny. V tomto případě se na tento termín nelze přihlásit. Pro odhlášení se používá metoda J 9e).

Profil

Po zavolání metody J 1a) se uživateli zobrazí údaje, které jsou k jeho osobě v systému vedeny. Údaje (s výjimkou jména, příjmení, loginu a identifikátoru) lze změnit za pomoci metody J 1h). Změna hesla probíhá tím způsobem, že pro zadané staré heslo se s pomocí soli, která byla uložena při přihlášení, vytvoří algoritmem MD5 haš o velikosti 128 bitů. Tento haš je porovnán s hašem uloženým při přihlášení uživatele a pokud se shodují, tak je umožněna změna hesla. Nové heslo musí být zadáno dvakrát, aby se ověřilo, že nové heslo bude bez překlepu. Pokud se haše shodují, nové heslo je 2x zadáno, pak je změna provedena metodou J 1j).

Veřejné kurzy

Metoda J 13a) je volána pro získání všech veřejných kurzů, které jsou následně zobrazeny do RecyclerView. Pro přihlášení na kurz (na který uživatel ještě přihlášen není) se využívá služeb serveru skrze zavolání metody J 1e). Z přihlášeného kurzu se lze odhlásit za asistence metody J 1f).

Nahlásit chybu

Actionbar – vizte: 4.8 obsahuje menu s položkou „Nahlásit chybu“, která je pro přihlášeného uživatele viditelná téměř všude v aplikaci. Po kliknutí na Actionbar se objeví uživateli okno s možností popsání chyby a jejím odesláním. Po úspěšném odeslání chyby navíc s podrobnostmi o verzi systému, používaném zařízení a identifikátorem uživatele je okno zavřeno a uživatel o odeslání zprávy informován. Pro nahlášení chyby se využívá metoda J 1q).

Garant

Přihlášení

Přihlášení je stejné jako přihlášení studenta – vizte kapitolu Přihlášení pro roli Student.

Zavedení podmínek kurzu

Komponenta ViewPager obsahuje jednu záložku pro studenty prezenční formy studia a jednu pro studenty formy kombinované. Data jsou společně získána po zavolání metody J 8f) a v záložkách uloženy do RecyclerView. Ke každé aktivitě lze získat seznam termínů uložený opět v RecyclerView. Data jsou získána metodou J 19a).

Pro každý termín lze zobrazit ViewPager se záložkou „informace“, „přihlášení studenti“ a „napsat zprávu“. V záložce informace lze přihlásit studenty pomocí metody J 19e). Pdf soubor s QR kódy studentů je získán za asistence metody J 19g) a pdf soubor se seznamem studentů využívá služeb metody J 19h). Dále se zde pomocí metody J 28a) a zadaného minima bodů zobrazí v záložce „přihlášení studenti“ studenti s dosaženým počtem bodů na vybraném termínu v RecyclerView. Studenty lze odhlásit metodou J 19f).

V záložce „napsat zprávu“ se využívá metoda J 1z) pro získání e-mailových adres všech studentů kurzu. Následně se z aplikace zavolá metoda J 33c), sloužící k odeslání e-mailové zprávy na všechny získané e-mailové adresy.

Přidání nového termínu probíhá s pomocí metody J 19d), aktualizace pomocí J 19c) a smazání zajišťuje metoda J 19b).

Souhrn výsledku vybrané skupiny aktivit je realizován komponentou RecyclerView a data jsou do ní získána metodou J 20a). Při výběru některého záznamu je za pomoci dvou ListView a metody J 20b) zobrazen souhrn výsledků vybraného studenta. V jednom sloupci jsou zobrazeny všechny výsledky a ve druhém sloupci výsledky, které se započítávají do celkového hodnocení studenta. Při výběru některého z výsledků v levém sloupci lze vybraný výsledek upřednostnit. Volá se k tomu metoda J 21a).

Skupinu aktivit lze vytvořit novou pomocí metody J 22c), upravit za asistence metody J 22a) nebo odstranit s použitím metody J 22d).

Načíst minimum/maximum kurzu lze pomocí J 13g) upravit s použitím metody J 13f). Pro prezenční a kombinovanou formu se data získávají i upravují separátně.

K zobrazení celkového souhrnu kurzu se používá metoda J 20c) a komponenta RecyclerView. Po výběru záznamu v RecyclerView je zobrazena komponenta ViewPager, která má tolik záložek, na kolik skupin aktivit, které se započítávají do výsledku, je uživatel přihlášen. V jednotlivých záložkách se zobrazuje souhrn výsledků daného studenta stejným způsobem jako v případě souhrnu výsledku některé vybrané skupiny aktivit, jak je popsáno výše.

Správa tutorů

Pro zobrazení všech tutorů v kurzu se využije dat vrácených metodou J 1k) a komponenty RecyclerView, ve které jsou zobrazeny. Jednotlivé tutory lze editovat za pomoci metody J 1a), která zajistí zobrazení všech potřebných upravitelných dat, a metody J 1h), která upravená data odešle na server. Tutora lze odstranit zavoláním metody J 1m).

V menu uloženém v Actionbaru lze vybrat možnost vytvoření nového tutora, což se děje s pomocí metody J 1l), která vytvoří nového uživatele a následně je zavolaná metoda J 1n), která zajistí propojení tutora s aktuálním kurzem. Další možností v Actionbaru je přidání tutora z již existujících (uživatelů). Po zadání alespoň části příjmení a stisknutí tlačítka hledat je zavolána metoda J 1p), která do komponenty RecyclerView vypíše tutory, které zadané části příjmení vyhovují. Ze zobrazených uživatelů lze vybrat jejich libovolný počet a po zavolání metody J 1o) se z nich stávají tutoři.

Seznam tříd a jejich studentů

Pro získání dat potřebných k zobrazení tříd slouží metoda J 31a). Třídy jsou pak vypsány pomocí RecyclerView do obsahu první záložky „správa tříd“ komponenty ViewPager. Pro vybrané třídy kombinovaného studia jsou viditelné nebo skryté jednotlivé tutoriály v témže RecyclerView s jinými informacemi a odlišným vzhledem.

Třídy lze vytvářet pomocí metody J 31b) a pomocí metody J 31c) upravovat. Pro kombinovanou formu a pro prezenční formu studia se sice používají stejné metody, ale při vytváření i úpravách tříd jsou zde patrné odlišnosti. Pro obě formy studia se volí tutor třídy. Tutor třídy se volí z komponenty RecyclerView, kam jsou vložena data získaná metodou J 1k). V prezenční formě se navíc, oproti údajům společným pro obě formy studia, volí den v týdnu. Pro formu kombinovanou se pak navíc volí jednotlivé tutoriály. Při nastavení konkrétního data se objeví tlačítko „Přidat do seznamu kombinovaných tříd“, které po vybrání dosud zadané údaje spolu s vybraným datem vloží do komponenty RecyclerView. Při nastavení jiného data a v případě potřeby jiných dalších hodnot lze přidávat do RecyclerView libovolný počet tutoriálů a také je odtud odstraňovat. Tutoriály s kolidujícím datem nejsou vloženy. Veškeré změny jsou uloženy až při skončení vkládání nové nebo úpravě stávající třídy.

Pro každou třídu v seznamu tříd je možné zobrazit seznam studentů. Studenti, získaní za pomoci metody J 1s), jsou vkládáni do RecyclerView, které je uvnitř komponenty ViewPager – konkrétně se jedná o obsah její první záložky. Každého studenta lze z kurzu odebrat s pomocí metody J 1t). Přidání studenta probíhá tak, že se podle zadaného loginu vyhledá (s využitím volání metody J 1y)) student, kterého lze do třídy přidat. Pokud student s takovým loginem neexistuje, tak je nejdříve vytvořen student nový s pomocí metody J 1l). Student je nakonec do třídy přidán tak, že je zavolána metoda J 1u). V popisovaném ViewPageru je další záložka s titulkem „napsat zprávu“. Ta využívá k odeslání zpráv metody popsané v podkapitole E-mailové zprávy.

Časové omezení pro změnu třídy probíhá tím způsobem, že jsou získána data po zavolání metody J 13e). Po změně těchto dat se v případě potřeby změny začátku nebo konce přihlašování do tříd využívá metody J 13h).

Vedle záložky „správa tříd“ komponenty ViewPager se nachází záložka „tabulková verze tříd“. Všechna potřebná data jsou získána stejně jako v případě „správy tříd“ z metody J 31a) a dále je dopočítáno, kde se v tabulkové verzi bude která hodina nacházet. Každý den se vykresluje do tabulky zvlášť. V případě kolizí se vytváří pro daný den, ve kterém kolize v rozvrhu vznikla, nový řádek, aby byly viditelné všechny předměty. Pokud se jedná o kombinovanou formu studia, tak se nepřidává pro každé datum nový záznam, ale pokud jsou tutoriály ve stejný den, čas a v jiné datum, tak je jen připsáno datum do záznamu v tabulkové verzi tříd. Mezery mezi jednotlivými hodinami jsou vykresleny „neviditelnými“ hodinami, aby bylo vždy korektní zarovnání. Přednáška i cvičení jsou odlišeny různou barvou. Celá tabulka je vytvořena komponentou TableLayout – vizte: 4.7. Dny v týdnu a časové rozmezí se nachází v jednotlivých TableRow – řádcích komponenty TableLayout. Další dělení už probíhá vždy pomocí komponenty LinearLayout – vizte: 4.7, to jak rozdělení jednotlivých záznamů v každém řádku, tak i uspořádání informací v jednotlivých záznamech.

V komponentě ViewPager se vedle záložky „tabulková verze tříd“ nachází záložka s odesláním zprávy třídě s titulkem „odeslat zprávu“. Obsah této záložky slouží k odesílání zpráv a využívá k tomu metody popsané v podkapitole E-mailové zprávy.

Správa testů a aktivit

Pro skupinu aktivit vybranou ze seznamu aktivit získaného pomocí metody J 22b) je uživateli zobrazena komponenta ViewPager se dvěma záložkami – jednou s titulkem „aktivity“ a jednou s titulkem „testy“.

Data do záložky „aktivity“ jsou získána z volání metody J 23a) a zobrazena v RecyclerView. Novou aktivitu lze vytvářet s pomocí metody J 23c), editovat tak, že upravená data jsou odeslána metodou J 23f) a nebo mazat vybranou aktivitu zavoláním metody J 23b). Zobrazování termínů a další operace s nimi spojené probíhají analogickým způsobem jako v případě Zavedení podmínek kurzu. Řešitelé konkrétní aktivity jsou získáni s využitím metody J 19a) pro získání termínů a pro vybraný termín se pak v RecyclerView zobrazí řešitelé, jejichž seznam je obdržen po zavolání metody J 19i) a následně J 7a). Pro stažení konkrétní přílohy se navíc využívá metody J 7c) až v případě potřeby pro úsporu přenášených dat. Hodnocení se pro jednotlivé studenty uděluje za asistence metody J 7e).

Data do záložky „testy“ jsou získána z volání metody J 24a). Data pro získání šablon pak z metody J 27a). Každému testu je zadán název šablony, která je k němu přiřazena a data jsou zobrazena opět v RecyclerView. Nový test se vytváří za asistence dat získaných z J 27a) a metody J 24c). Editace pak opět využívá data z J 27a) a pak pro samotnou úpravu metodu J 24d). Mazání testů zajišťuje volání metody J 24b). Co se týče termínů, tak práce s nimi opět probíhá analogicky jako v případě Zavedení podmínek kurzu. Pro každý test lze zobrazit v komponentě RecyclerView pro termín vybraný ze seznamu termínů (získaných z volání metody J 19a)) generované testy. Data pro seznam generovaných testů jsou získávána pomocí volání metody J 11k). Nový test se vygeneruje po zavolání metody J 11l).

Správa otázek

Pro zobrazení otázek je třeba v aplikaci nejdříve zavolat metodu J 17a) pro získání kapitol, následně metodu J 29a) pro získání kategorií. Otázky jsou získány z dat obdržených po zavolání metody J 26b) a zobrazeny v ListView. Zobrazené otázky je možné odstranit za asistence metody J 26d).

K vytvoření nové otázky se využívá dat získaných z metod J 17a) – pro možnost zobrazení a vybrání kapitol, J 29a) – pro možnost zobrazení a vybrání kategorií vybrané kapitoly, J 25a) – pro získání kroků vybrané kategorie. Dále je k úspěšnému vytvoření otázky nutné zavolat metodu J 26c) pro vložení otázky a následně metodu J 30a) pro vložení odpovědí k vkládané otázce. Pro zobrazování speciálních znaků využívaných v předmětu Matematická logika je využíváno třídy SpecialCharDialogButton – vizte: 4.8. Pro komunikaci ze serverovou částí se pak využívá transformace speciálních znaků za asistence třídy LogicParser – vizte: 4.8.

Správa kategorií

S pomocí metody J 17a) jsou získány kapitoly a po vybrání některé z nich se do ListView vloží její kategorie. Aby tomu tak mohlo být, tak je samozřejmě nejprve potřeba pro vybranou kapitolu zavolat metodu pro získání kategorií vybrané kapitoly. Metodu lze nalézt v dokumentaci zde: J 29a). Každou zobrazenou kategorii lze smazat s využitím metody J 29d). Pro vytvoření i

aktualizaci kategorií se využívá dat z metody J 17a). Při vytváření kategorie je potřeba volat metodu J 29b) k vytvoření samotné kategorie spolu s metodou J 25b) pro vložení nově vytvořených kroků svázaných s danou kategorií na server. Pro úpravu údajů se volají metody J 29c) pro aktualizaci údajů již existující kategorie společně s metodou J 25c) potřebnou pro aktualizaci upravených kroků, stejně tak jako pro vkládání kroků nových. Pro zobrazení kroků se využívá android komponenta RecyclerView.

Správa šablon

Šablony jsou zobrazeny v RecyclerView s možností editace, smazání a vytváření šablon nových. Pro získání šablon k zobrazení se využívá metoda J 27b), ke smazání pak J 27d). V prvním kroku vytváření a úpravy šablon se volí počet bloků, které se dále upravují. Pro zobrazení všech potřebných dat bloků se používá HierarchicalAdapter – vizte: 4.8.

Pro vytváření nové šablony se nejdříve zavolá metoda J 17a) pro získání kapitol a podkapitol, dále metoda J 29a) pro získání kategorií pro každou kapitolu, pro každou kategorii se získají její kroky pomocí metody J 25a). Informace o otázkách každé kategorie je možné získat po zavolání metody J 26a). Po získání všech těchto dat již lze vytvořit šablonu. K tomu se využívá metoda J 16c).

Pro úpravu existující šablony se nejdříve volá metoda J 16a) pro získání bloků vybrané šablony, poté jsou volány metody pro získání všech kapitol a podkapitol, pro každou kapitolu pro získání její kategorie, pro každou kategorii je volána metoda pro získání kroků a otázek. Toto vše se získává stejným způsobem jako pro vkládání nové šablony. Po skončení úprav se volá metoda J 27c) k úpravě dat o šabloně a následně J 16d) k úpravě všech bloků upravované šablony.

Správa kapitol

Pomocí SeparatedListAdapter jsou zobrazeny sekce v ListView. Po zavolání metody J 17a) jsou získaná data potřebná k zobrazení kapitol a podkapitol. Data potřebná pro zobrazení materiálů jsou získána metodou J 17b) a data pro zobrazení cvičných testů zajišťuje metoda J 8g).

Kapitolu lze smazat za asistence metody J 17f) a upravit pomocí metody J 17e). Vložení nové kapitoly je zajištěno s pomocí metody J 17c) a vkládání nového materiálu pomocí J 17g). Materiál může být také smazán, což je snadno proveditelné díky metodě J 17d).

Aktuality

Aktuality jsou získány metodou J 10a) a zobrazeny pomocí komponenty RecyclerView a šablony list_view_swipe. Přílohy pro jednotlivé aktuality jsou získány po zavolání metody J 10b). Vkládání nové aktuality je umožněno za asistence metody J 10e) a úprava existujících aktualit vyžaduje použití metody J 10d). Smazání aktuality je umožněno zavoláním metody J 10g). Odstranění příloh je možné při použití metody J 10c).

E-mailové zprávy

Odeslané e-mailové zprávy jsou po získání ze serveru metodou J 33a) zobrazeny v RecyclerView, odkud mohou být odstraněny zavoláním metody J 33b).

Pro vytvoření nové zprávy je zobrazena android komponenta ViewPager obsahující 2 záložky. Na první záložce je zobrazen seznam tříd v RecyclerView, který je získán z dat obdržených ze serveru po zavolání metody J 31a). Vybraným třídám lze odeslat zprávu z druhé záložky komponenty ViewPager, a to zavoláním metody J 33d). Pro libovolnou třídu ze seznamu tříd v RecyclerView lze zobrazit všechny její studenty. Tito studenti jsou získáváni metodou J 1s) a zobrazeni v první záložce komponenty ViewPager uvnitř RecyclerView s možností filtrování podle loginu. Vedle záložky „seznam studentů“ se nachází záložka s možností sestrojení a odeslání zprávy. V tomto případě se zpráva odešle s pomocí metody J 33c).

Konzultační hodiny

Seznam konzultačních hodin je zobrazen v první záložce komponenty ViewPager. Data jsou získávána ze serveru po zavolání metody J 9f) a zobrazena v RecyclerView. Konzultační hodiny zde lze mazat s pomocí metody J 9i), vytvářet nové a existující upravovat s využitím metody J 9h). Druhá záložka komponenty ViewPager zobrazuje nejbližší obsazené konzultační hodiny. Data jsou získána ze serveru po zavolání metody J 9g) a zobrazena v RecyclerView s využitím třídy HierarchicalAdapter. Třetí záložka „historie konzultačních hodin“ zobrazuje rovněž RecyclerView, využívá třídu HierarchicalAdapter a data získává z volání metody J 9f).

Z druhé a třetí záložky je možné pro vybranou konzultační hodinu zobrazit dialogové okno se studenty, kteří jsou na tuto hodinu přihlášeni. Tito studenti jsou získáváni pomocí metody J 9c) a e-mailová zpráva je pak odesílána za asistence metody J 33c).

Logování

Pro získání dat k zobrazení logu přístupů je použita metoda J 1v), data k zobrazení online testů jsou získána z metody J 1x) a data pro zobrazení všech akcí zajišťuje metoda J 1w). Data jsou získána po zadání loginu a zvolení typu logu, který je následně vypsán do RecyclerView.

Nahlásit chybu

Implementace je shodná s implementací popsanou v části Nahlásit chybu uživatele v roli studenta.

Profil

Implementace je shodná s implementací popsanou v části Profil uživatele v roli studenta.

Tutor

Přihlášení

Přihlášení je stejné jako přihlášení studenta – vizte kapitolu Přihlášení pro roli Student.

Zavedení podmínek kurzu

Komponenta ViewPager obsahuje jednu záložku pro studenty prezenční formy studia a jednu pro studenty formy kombinované. Data pro obě formy studia jsou získána po zavolání metody J 8f) a v jednotlivých záložkách zobrazeny v RecyclerView.

Souhrn výsledku vybrané skupiny aktivit je realizován komponentou RecyclerView a data jsou do ní získána metodou J 20a). Při výběru některého záznamu je za pomoci dvou ListView a metody J 20b) zobrazen souhrn výsledků vybraného studenta. V jednom sloupci jsou zobrazeny všechny výsledky a ve druhém sloupci výsledky, které se započítávají do celkového hodnocení studenta. Při výběru některého z výsledků v levém sloupci lze vybraný výsledek upřednostnit. Volá se k tomu metoda J 21a).

Skupinu aktivit lze novou vytvořit pomocí metody J 22c).

Načíst minimum/maximum kurzu lze pomocí J 13g) upravit s použitím metody J 13f). Pro prezenční a kombinovanou formu se data získávají i upravují separátně.

K zobrazení celkového souhrnu kurzu se používá metoda J 20c) a komponenta RecyclerView. Po výběru záznamu v RecyclerView je zobrazena komponenta ViewPager, která má tolik záložek, na kolik skupin aktivit, které se započítávají do výsledku, je uživatel přihlášen. V jednotlivých záložkách se zobrazuje souhrn výsledků daného studenta stejným způsobem jako v případě souhrnu výsledku některé vybrané skupiny aktivit, jak je popsáno výše.

Správa testů a aktivit

Implementace je shodná s implementací popsanou v části Správa testů a aktivit uživatele v roli garanta.

Správa otázek

Implementace je shodná s implementací popsanou v části Správa otázek uživatele v roli garanta.

Seznam tříd a asociace se studenty

Implementace je analogická s implementací popsanou v části Správa otázek uživatele v roli garanta s tím rozdílem, že editace a odstranění existující třídy jsou dostupné pouze u tříd, kde je tutor uveden jako učitel. Zobrazení komponenty ViewPager se seznamem studentů vybrané třídy v RecyclerView uvnitř první záložky a možností odeslání zprávy ze záložky druhé je také umožněno tutorovi pouze u tříd, kde je veden jako učitel.

Správa kategorií

Po zavolání metody J 17a) jsou získány kapitoly. Pro vybranou kapitolu se do ListView vloží její kategorie získané z metody J 29a).

Správa šablon

Implementace je shodná s implementací popsanou v části Správa šablon uživatele v roli garanta.

Správa kapitol

Pomocí SeparatedListAdapter jsou zobrazeny sekce v ListView. Po zavolání metody J 17a) jsou získána data potřebná k zobrazení kapitol a podkapitol. Data potřebná pro zobrazení materiálů jsou získána metodou J 17b) a data pro zobrazení cvičných testů zajišťuje metoda J 8g). Materiál může být smazán, což je snadno proveditelné díky metodě J 17d).

Aktuality

Implementace je shodná s implementací popsanou v části Aktuality uživatele v roli garanta.

E-mailové zprávy

Implementace je shodná s implementací popsanou v části E-mailové zprávy uživatele v roli garanta.

Konzultační hodiny

Implementace je shodná s implementací popsanou v části Konzultační hodiny uživatele v roli garanta.

Logování

Implementace je shodná s implementací popsanou v části Logování uživatele v roli garanta.

Nahlásit chybu

Implementace je shodná s implementací popsanou v části Nahlásit chybu uživatele v roli studenta.

Profil

Implementace je shodná s implementací popsanou v části Profil uživatele v roli studenta.

4.4 Bezpečnost offline testů

Testy jsou uloženy do databáze SQLite do 4 tabulek. Jedna tabulka slouží k uložení informací o ukládaném testu, následně se zde nachází tabulka k uchování otázek, tabulka pro odpovědi a čtvrtá tabulka obsahuje další informace o testech, které nejsou odeslány na server.

Data z SQLite databáze jsou čitelná kdekoli v aplikaci, ale ne mimo ni. [23]

Informace o vygenerovaném testu, otázkách a odpovědích jsou šifrovány algoritmem AES v jeho nejjednodušší formě - módu ECB. Jedná se o symetrickou šifru se 128bitovým klíčem. K šifrování byla použita třída jazyka Java SecretKeySpec ke správné manipulaci s klíčem a třída jazyka Java Cipher pro samotné zašifrování či odšifrování textu. Více informací o těchto třídách lze nalézt v dokumentaci pro vývoj android aplikací nebo v dokumentaci jazyka Java – vizte: [32] a [33].

4.5 Jazykové mutace

Při vývoji aplikace bylo dbáno na to, aby všechny texty, které jsou v aplikaci viditelné, byly uloženy v jednom souboru. Tento soubor se nachází uvnitř složky s android aplikací. Relativní cesta k tomuto souboru v rámci složky se zdrojovými soubory aplikace se nachází zde: „\app\src\main\res\values\strings.xml“.

Defaultním jazykem je nyní čeština. Pokud by byl tento soubor s texty, které se v aplikaci uživatelům zobrazují, přeložen do jiného jazyka – například do angličtiny, tak by se na zařízeních,

které mají nastaven jazyk prostředí na angličtinu, zobrazovala aplikace s anglickými texty. Analogickým způsobem lze doplnit veškeré jazyky, které jsou uvedeny v normě ISO 639.2 – vizte: [3]. Z této normy lze vyčíst dvojpísmennou zkratku pro každý uvedený jazyk. V případě podpory některého jazyka je do tohoto jazyka přeložený XML soubor „strings.xml“ uložen do složky, která obsahuje ve svém názvu onu dvojpísmennou zkratku – například pro španělský jazyk se bude složka se souborem „strings.xml“ jmenovat values-es. Složka ve výše uvedené cestě nemá dvojpísmennou zkratku, protože se jedná o výchozí jazyk, který je použit pro všechny jazyky, které nejsou explicitně uvedeny – tedy neexistuje složka values s dvojpísmennou zkratkou daného jazyka.

4.6 Použité návrhové vzory

Návrhové vzory jsou návrhem osvědčeného řešení problému, který se v historii již mnohokrát opakoval, a byl tímto způsobem úspěšně vyřešen. Návrhové vzory můžeme klasifikovat do 3 skupin – Návrhové vzory tvořící, strukturální a vzory chování.

Jedináček (Singleton)

Tvořící návrhový vzor, který zajišťuje, že třída bude mít nejvýše jednu instanci a k této instanci zajišťuje přístup (případně vytvoření instance, pokud se jedná o první přístup ke třídě nebo pokud byla instance odebrána garbage collectorem).

Třídní diagram tohoto návrhového vzoru použitého při vytváření aplikace eLogika lze nalézt v příloze B obrázek 13.

Továrna (Factory Method)

Továrna je dalším tvořícím návrhovým vzorem, který definuje rozhraní pro vytváření objektů, ale nechává podtřídy rozhodnout, která třída bude instanciována. [40]

Třídní diagram tohoto návrhového vzoru použitého při vytváření aplikace eLogika lze nalézt v příloze B obrázek 12.

Kompozit (Composite)

Návrhový vzor Kompozit řadíme do skupiny strukturálních návrhových vzorů. Skládá objekty hierarchicky do stromové struktury a nechává klienty k jednotlivým objektům přistupovat. Listový uzel a vnitřní uzel stromu mají stejné rozhraní. [40]

Ukázku použití tohoto návrhového vzoru při vytváření aplikace eLogika lze nalézt v příloze B obrázek 14.

Šablona (Template method)

Návrhový vzor Šablona patří do skupiny vzorů chování. Je definována kostra, ale některé její části jsou ponechány neimplementované a implementují je potomci této třídy. [40]

Třídní diagram tohoto návrhového vzoru použitého při vytváření aplikace eLogika lze nalézt v příloze B obrázek 13.

Lazy Load

Všechny výše zmíněné návrhové vzory jsou popsány v knize notoricky známé jako „GOF book“, neboť její název je příliš dlouhý (Design Patterns: Elements of Reusable Object-Oriented Software), a napsali ji čtyři autoři – Erich Gamma, Richard Helm, Ralph Johnson a John Vlissides. Lazy Load je návrhový vzor, který není zaznamenán v této knize, ale popsal jej Martin Fowler ve své publikaci Patterns of enterprise application architecture.

Lazy Load je objekt, který neobsahuje všechna potřebná data, ale v případě potřeby ví, jak je získat. Tento vzor je v aplikaci naimplementován například v aktivitách, kdy se v případě potřeby stahují soubory se vzorovým řešením nebo soubory s popisem zadání.

4.7 Android API

View

Třída View (android.view.View) je základním stavebním blokem pro komponenty grafického uživatelského rozhraní. Jde o oblast vyplňující displej zařízení s OS Android a zodpovídá za vykreslení dat na onen displej a odchytávání akcí uživatele. View má podtřídu ViewGroup, která je základní třídou pro „layouty“. Layout je neviditelný kontejner držící další View (ViewGroup) a definuje jejich vlastnosti. [35]

Některé příklady Layoutů jsou popsány dále v této kapitole a jsou to základní stavební kameny grafického uživatelského rozhraní používaného v rámci aplikace eLogika.

Adapter

Rozhraní Adapter (android.widget.Adapter) slouží k získávání dat (ze struktury, databáze, pole) a jejich transformaci do View. [37]

Třídy implementující toto rozhraní a použité při implementaci aplikace eLogika jsou ArrayAdapter, SimpleAdapter, RecyclerView.Adapter, PagerAdapter a další.

ListView

ListView (android.widget.ListView) je android komponenta dědící ze třídy View a ViewGroup. Slouží k zobrazení seznamu posouvatelných prvků v jednom sloupci. Prvky jsou do ListView automaticky vkládány skrze třídu implementující rozhraní Adapter. [26]

Pro ukázku použití ListView vizte obrázek 42.

LinearLayout

Třída LinearLayout (android.widget.LinearLayout) je potomkem třídy View (i ViewGroup). Zarovnává všechny další komponenty uživatelského rozhraní, které jsou definovány uvnitř jednoho LinearLayoutu, jedním směrem. Všechny prvky jsou tedy vkládány za sebe horizontálně – do jednoho řádku, anebo pod sebe vertikálně – do jednoho sloupce. Více o vlastnostech LinearLayoutu se můžete dočíst zde: [28]

LinearLayout je nejčastější Layout použitý pro konstrukci grafického uživatelského rozhraní aplikace eLogika.

TableLayout

TableLayout (android.widget.TableLayout) je přímým potomkem třídy LinearLayout. TableLayout obsahuje komponenty TableRow, které reprezentují řádky dvourozměrné matice uživatelského rozhraní. Sloupců pak je v rámci třídy TableLayout stejný počet, kolik má řádek s nejvíce buňkami. [29]

RelativeLayout

Posledním potomkem třídy View (i třídy ViewGroup) zmíněným v tomto textu je RelativeLayout (android.widget.RelativeLayout). Elementy uvnitř elementu RelativeLayout nejsou jako dříve zmíněné Layouty umísťovány do řádků či sloupců, ale jejich pozice je definována jako relativní (odtud název RelativeLayout) vzhledem k dalším elementům nebo okrajům displeje android zařízení, na kterém je aplikace spuštěna.

SharedPreferences

Třída SharedPreferences (android.content.SharedPreferences), která poskytuje obecné rozhraní umožňující ukládat a načítat perzistentní data uložená ve tvaru klíč-hodnota. Lze ukládat libovolné primitivní datové typy - booleovské hodnoty, hodnoty typu float, int, long a řetězce. Data zůstávají v aplikaci i po jejím ukončení. [24]

K datům může ve výchozím stavu přistupovat pouze aplikace, která data uložila, nebo aplikace se stejným identifikátorem. [24]

Support knihovna

Balíček knihovních funkcí nesoucí název „Android Support Library“ (android.support.*) slouží ke zpětné podpoře nových funkcí pro starší verze OS Android (rozhraní android API s nižším číslem), než pro které byly napsány. Při implementaci lze tedy využívat vlastností z novějšího API, než pro které bude aplikace stále kompatibilní. Více informací a změny, které byly provedeny ve které verzi balíčku knihoven, lze vysledovat zde: [25]

Každá z knihoven, které lze volitelně do aplikace při jejím vývoji přidat, je určena pro určité verze android API. Například knihovny, které mají na začátku svého názvu „v7“, zavádí podporu vybraných vlastností pro android API 7 a novější. Všechny knihovny s jejich vlastnostmi lze nalézt zde: [34]

SwipeRefreshLayout

Celý návrhový vzor uživatelského rozhraní „Swipe-to-Refresh“ je implementován pomocí třídy SwipeRefreshLayout (android.support.v4.widget.SwipeRefreshLayout), která detekuje vertikální swipe (gesto prováděné táhlým pohybem po displeji zařízení), po detekci zobrazí progress bar (komponenta, díky níž je uživatel informován o tom, že probíhá nějaká akce) a zavolá metodu, která znovu provede definovanou činnost. [30]

Typickým příkladem použití je přidání třídy SwipeRefreshLayout do třídy ListView (případně RecyclerView) a implementace metody (zavolané pomocí gesta swipe), která znovu načte data do ListView (případně RecyclerView).

Pro ukázku použití SwipeRefreshLayout uvnitř ListView vizte obrázek 42.

ViewPager

Třída ViewPager (android.support.v4.view.ViewPager) se využívá pro změnu obsahu komponenty skrze horizontální swipe. [31]

V aplikaci eLogika se používá nejčastěji v kombinaci se třídou FragmentStatePagerAdapter (android.support.v4.app.FragmentStatePagerAdapter), kdy se definované fragmenty pomocí horizontálního swipe gesta střídají a uživatel si tak volí, který fragment chce mít viditelný.

Pro ukázku použití ViewPager vizte obrázek 42.

RecyclerView

RecyclerView (android.support.v7.widget.RecyclerView) je flexibilnější verze ListView. Jeho použití je doporučeno zejména pro zobrazování velkého množství dat nebo dat, které se mění za běhu aplikace (a to buď na popud uživatele, nebo událostmi vznikajícími skrze internetové připojení). Třída RecyclerView obsahuje třídu LayoutManager, která slouží k pozicování jednotlivých elementů a rozhoduje, kdy již není položka pro uživatele viditelná a lze ji tedy „recyklovat“ tak, že se její view použije s novými daty. Tento přístup zlepšuje výkon tím, že se nevytváří View pro všechny položky uvnitř třídy RecyclerView. [27]

Třída RecyclerView obsahuje vlastní animace pro přidávání a odebírání jednotlivých položek. Jak přidávání nových, tak odebírání stávajících položek z RecyclerView je snadno proveditelné.

Pro ukázku použití RecyclerView vizte obrázek 57.

4.8 Knihovna vlastních funkcí

Knihovna vlastních funkcí v rámci aplikace eLogika vznikla pro soustředění tříd, které obsahovaly buď služby, které jsou využívány na více místech v aplikaci, nebo služby, které by mohly být v budoucnu využity i na jiných místech, než kde jsou použity. Jde o samostatné třídy, které jsou často statické nebo obsahují statické metody. Některé z nich budou zmíněny v této kapitole, jelikož je o nich zmínka v kapitole 4.3.

DownloadFile

Třída slouží k vytvoření souboru a jeho uložení do složky android zařízení, která je určena pro stahované soubory. Využívány jsou k tomu 2 metody, které následně vrací Intent (abstraktní popis nějaké operace, která se má provést), jenž nese informaci o tom, že má být soubor v zařízení otevřen. Intent nese doplňující informaci o typu souboru. Rozdíl v těchto dvou metodách spočívá v tom, že jedna kromě jména, které určuje výsledný název souboru, obsahuje parametr s polem

bytů (v kódování Base64), ze kterého je soubor vytvořen. Druhá metoda místo pole bytů bere jako druhý parametr řetězec.

SeparatedListAdapter

Třída dědící ze třídy BaseAdapter. Více o abstraktní třídě Adapter se můžete dočíst v kapitole 4.7. Tato třída rozděluje ListView na sekce a ke každé sekci přidává hlavičku.

Pro ukázku použití SeparatedListAdapteru vizte obrázek 48.

Actionbar

Statická třída Actionbar slouží k zobrazování hierarchie fragmentů uvnitř panelu v horní části aplikace a umožňuje ovládání hierarchie fragmentů. Realizuje přechody do fragmentů hierarchicky vyšších buď po vybrání uživatelem, nebo programově.

Z vyobrazeného panelu obsluhovaného třídou Actionbar také probíhá nahlášení chyby, což je položka menu ukrytého na pravém kraji panelu pod tlačítkem se třemi tečkami, jak je v android zařízeních zvykem. Do tohoto menu lze přidat více položek opět za asistence třídy Actionbar. Více se o možnostech použití nebo o implementaci můžete dočíst v programátorské dokumentaci či ve zdrojovém kódu.

Pro ukázku použití třídy Actionbar vizte obrázek 58.

TimeLogika

Třída TimeLogika soustřeďuje a zjednodušuje práci s všudypřítomnými časovými údaji. Převádí formát data a času mezi standardním formátem serverové části aplikace a zobrazováním informací uživateli, dále časy zaokrouhluje, porovnává, vybírá jen datum, nebo jen časový údaj, poskytuje informace o tom, jestli aktuální čas je uvnitř intervalu mezi dvěma časy, provádí převody mezi časovými jednotkami, vrací aktuální čas v požadovaném formátu, rozděluje čas do polí podle časových jednotek, zjišťuje den v týdnu z data, nebo pro vybraný den v týdnu vrací jeho název v jazyce koncového zařízení.

Více o možnostech použití této třídy a o detailech její implementace můžete nalézt v přiložené programátorské dokumentaci nebo v komentářích jednotlivých metod ve zdrojovém kódu.

SpecialCharDialogButton

Třída dědící ze třídy Button reprezentuje tlačítko. Po kliknutí na toto tlačítko se zobrazí uživateli speciální klávesnice, která obsahuje znaky používané v předmětu Matematická logika a slouží například pro popis rovnic v otázkách testů.

Pro ukázku použití třídy SpecialCharDialogButton vizte obrázek 54.

LogicParser

LogicParser je třída, která obousměrně převádí speciální znaky uvedené v popisu třídy SpecialCharDialogButton na znaky, které jsou používány pro přenos těchto informací mezi serverovou částí a aplikací na zařízení s OS Android.

TimePickerDialogButton

Třída TimePickerDialogButton rozšiřuje funkčnost třídy Button a využívá android třídu TimePickerDialog, která umožňuje výběr času na koncovém android zařízení, a to takového, že je na něj uživatel daného zařízení zvyklý, protože vychází ze vzhledu, která je pro API koncového zařízení typická. Dále obsahuje metody pro nastavení data, zobrazení data na tlačítku, získání údajů o datu z dialogového okna vytvořeného za asistence třídy TimePickerDialog. O této třídě se můžete více dočíst v programátorské dokumentaci nebo komentářích zdrojového kódu.

TimePickerDialogOnButton

TimePickerDialogOnButton funguje obdobně jako třída TimePickerDialogButton s tím rozdílem, že nerozšiřuje funkčnost třídy Button, ale bere její instanci jako parametr. Využívá služeb android třídy TimePickerDialog, nastavuje čas a vrací jej v různých formátech. O třídě se můžete více dočíst v programátorské dokumentaci nebo komentářích zdrojového kódu.

Pro ukázku použití třídy TimePickerDialogOnButton vizte obrázek 44.

TimePickerDialogButton_HMS

V otázkách se využívá pro nastavení času třída TimePickerDialogButton_HMS, která dědí ze třídy Button a od TimePickerDialogButton se liší tím, že má jiný vzhled a má funkce pouze pro nastavení času, jeho zobrazování v hodinách, minutách a vteřinách. Po vybrání požadovaného času je čas rovnou vrácen ve vteřinách, protože to je jednotka, která je vyžadována serverovou částí v případě vytváření a aktualizace otázek.

Pro ukázku použití třídy TimePickerDialogButton_HMS vizte obrázek 52.

DatePickerDialogButton

Třída DatePickerDialogButton rozšiřuje funkčnost třídy Button a využívá android třídu DatePickerDialog, která umožňuje výběr času na koncovém android zařízení, a to takovém, že je na ně uživatel daného zařízení zvyklý, protože vychází ze vzhledu, který je pro API koncového zařízení typický. Dále obsahuje metody pro nastavení data, zobrazení data na tlačítku, získání údajů o datu z dialogového okna vytvořeného za asistence třídy DatePickerDialog. O této třídě se můžete více dočíst v programátorské dokumentaci nebo komentářích zdrojového kódu.

DatePickerDialogOnButton

DatePickerDialogOnButton funguje obdobně jako třída DatePickerDialogButton s tím rozdílem, že nerozšiřuje funkčnost třídy Button, ale bere její instanci jako parametr. Využívá služeb android třídy DatePickerDialog, nastavuje čas a vrací jej v různých formátech. Třída DatePickerDialogOnButton je také využívána pro nastavení jak data, tak zároveň i času, a to díky

metodě setDateTime, kam je vkládán parametr typu TimePickerDialogOnButton. Obdobně lze také získat pomocí jedné metody jak informace o nastaveném datu, tak i o nastaveném čase. Tato třída dále umožňuje nastavit akce, které se vykonají po aktivaci tlačítka vkládaného do konstruktoru třídy DatePickerDialogOnButton, o níž se můžete více dočíst v programátorské dokumentaci nebo komentářích zdrojového kódu.

Pro ukázku použití třídy DatePickerDialogOnButton vizte obrázek 50.

HierarchicalAdapter

Abstraktní třída HierarchicalAdapter slouží jako hierarchický Adapter pro RecyclerView. Více o abstraktní třídě Adapter a třídě RecyclerView se můžete dočíst v kapitole 4.7.

Tato třída umožňuje zobrazení více než dvouúrovňové stromové struktury, což není nativně při vývoji android aplikací podporováno standardními komponentami. V rámci této abstraktní třídy se vytváří tzv. holdery (definují strukturu dat), vkládá se do layoutu tlačítko pro zobrazení a schování určité úrovně.

HierarchicalAdapter obsahuje vnitřní třídy, které pak umožňují vytváření dalších úrovní. V každé úrovni se mohou vytvářet kategorie, které mohou obsahovat další potomky stromové struktury, nebo se jedná o potomky, kteří už další potomky nemají. Dopodrobna se o této třídě můžete v případě zájmu dočíst v programátorské dokumentaci nebo v komentářích zdrojového kódu aplikace eLogika.

Pro ukázku použití třídy HierarchicalAdapter vizte obrázek 51.

4.9 Šablony pro View

V této kapitole jsou ukázky některých z používaných šablon vytvořených v rámci implementace grafického uživatelského rozhraní. Tyto šablony jsou uvedeny jako příklad použití komponent Android API popsaných v kapitole 4.7.

list_view_swipe

XML soubor list_view_swipe slouží jako šablona pro uživatelské rozhraní. Uvnitř FrameLayoutu, který je nepřímým potomkem třídy View (a také třídy ViewGroup) a zaujímá celý prostor, ve kterém je tato šablona vložena, se nachází komponenta SwipeRefreshLayout. Uvnitř této komponenty má své místo LinearLayout, který řadí za sebe další LinearLayout (s obrázkem a popisem, že se nenachází žádná data k zobrazení, viditelným pouze v případě, že nejsou přítomna žádná data v ListView) a ListView.

Pro ukázku použití list_view_swipe vizte obrázek 42.

recycler_view_with_button_swipe

Tento XML soubor sloužící jako další z několika šablon uživatelského rozhraní obsahuje RelativeLayout, ve kterém je vložen FrameLayout, obdobný jako v případě šablony list_view_swipe. Pod tímto layoutem je definováno kulaté tlačítko, které částečně překrývá pravý dolní roh FrameLayoutu. Toto tlačítko je vždy viditelné i v případě, že je FrameLayout

posunován. Uvnitř FrameLayoutu se nachází SwipeRefreshLayout umožňující „Swipe-to-Update“ a obalující LinearLayout, který pod sebe řadí LinearLayout (s obrázkem a popisem, že se nenachází žádná data k zobrazení, viditelným pouze v případě, že nejsou přítomna žádná data v RecyclerView) a RecyclerView.

4.10 Knihovny třetích stran

Při vývoji aplikace byly použity některé knihovny třetích stran. Konkrétně jde o knihovnu PhotoView ve verzi 1.2 [43], která slouží k přibližování obrázků u testů. Knihovna Snackbar [44] byla použita ve verzi 2.9.0 a to z důvodu přiblížení se vzhledu Material design v době, kdy ještě nebylo snadné použití této komponenty pouze s využitím Support knihovna. V současné době Support knihovna, více o této knihovně v kapitole 4.7, již využití komponenty Snackbar podporuje, ale nenabízí tolik možností, jako knihovna, která je v projektu použita. Poslední použitá knihovna, která není standardní pro vývoj android aplikací, je knihovna GSON [45]. Ta slouží k serializaci a deserializaci JAVA objektů do formátu JSON a zpět.

Pro ukázku použití knihovny Snackbar vizte obrázek 43.

4.11 Spuštění aplikace

Pro spuštění aplikace, je potřeba mít k dispozici zařízení (mobilní telefon nebo tablet s OS Android 4.1 nebo vyšším), doplněk ARC (v prohlížeči Chrome [42]) nebo emulátor (standardní emulátor v rámci Android studia nebo jiný emulátor s OS Android 4.1 nebo vyšším).

Dále je ke spuštění aplikace z přiložených zdrojových kódů nutné mít nainstalováno Android Studio [2], SDK nástroje (SDK Tools) verze 23 (což odpovídá OS Android 6.0) a JDK verze minimálně 7.

V rámci Android Studia je možné vygenerovat APK balíček, který je možné nahrát do zařízení se systémem Android skrze připojené zařízení, což vyžaduje ADB ovladač (více o ADB lze nalézt zde: [46]) a povolení funkce ladění v daném zařízení. Pro spuštění v emulátoru je pouze doporučen ovladač HAXM [21].

Android Studio umožňuje vygenerování APK balíčku a jeho podepsání certifikátem. Balíček podepsaný certifikátem lze spustit na všech výše zmíněných zařízeních (s povolením instalací aplikací mimo Google Play), emulátorech a doplňku ARC.

5 Závěr

Tato diplomová práce seznamuje jejího čtenáře s analýzou a návrhem aplikace pro OS Android, která bude použita pro e-learningovou výuku předmětů na Vysoké škole báňské – Technické univerzitě Ostrava. V textu práce je popsána požadovaná funkčnost aplikace a hlavní výhody mobilní aplikace oproti stávajícímu řešení.

Práce vyobrazuje některé diagramy, které sloužily k analýze a návrhu řešení android aplikace, některé návrhové vzory, které byly použity k řešení známých problémů při vývoji a aplikace, zařízení a technologie, kterých bylo při vývoji aplikace využíváno. Díky této diplomové práci jsem se přesvědčil o významu používání diagramů, smyslu výuky návrhových vzorů, často se mi osvědčilo používání komentářů určených pro programátorskou dokumentaci, seznámil jsem se s novými technologiemi zmíněnými v textu a měl jsem poprvé možnost pracovat na projektu v rámci menšího týmu. Jedná se zatím o největší aplikaci, kterou jsem vytvořil, a tak je můj osobní přínos, co se řešení různých problémů při vývoji týče, značný.

Jedním z cílů práce byl návrh rozhraní pro komunikaci se serverem systému eLogika. Při návrhu metod sloužících ke komunikaci byl kladen důraz na to, aby nebylo příliš zasahováno do serverové části, která se již pro výuku některých předmětů na VŠB-TUO využívá. Nově vytvářené metody byly nejprve otestovány na testovacím serveru a až poté nasazeny do ostrého provozu.

V rámci diplomové práce vznikla namísto dvou aplikací aplikace jedna, která splňuje požadavky prokonzultované s vedoucím diplomové práce. Největší výhodou jedné aplikace byl snadný vývoj společných částí pro všechny role a snadnější údržba a modifikovatelnost než v případě aplikací dvou. Toto řešení pak vyžadovalo zvláštní pozornost při tvorbě částí, které jsou společné jak pro mobilní telefony, tak pro tablety. V textu práce jsou stručně popsány části týkající se vývoje pro OS Android s četnými odkazy do online dokumentace, dále jsou popsány některé knihovní funkce, které v rámci vývoje aplikace vznikly, a na obrázku 7 je zobrazena část programátorské dokumentace vzniklé aplikace, která bude sloužit pro další vývoj.

Nástroje pro tvorbu aplikací OS Android se neustále vyvíjejí a při využívání novějších verzí s opravami chyb a novými funkcemi je potřeba provádět především regresní testování. Aplikace byla testována nejdříve na testovacím serveru a poté také na ostrém serveru s testovacími daty. Vývoj aplikace by se do budoucna měl ustálit na některé stabilní verzi vývojového prostředí a SDK nástrojů. V další řadě by mělo být provedeno důkladné testování na reálných datech a mnoha zařízeních s různými verzemi operačního systému, aby bylo docíleno skutečně toho, že všem studentům bude aplikace fungovat tak, jak má. Nová verze Android 6.0, nové SDK nástroje připravující se na příchod další verze OS (v současné době označované jako Android N) a nové Android Studio 2.0 vneslo do aplikace několik chyb způsobených nekompatibilitou se staršími produkty společnosti Google, nebo jejich použitím, které již není validní.

Pro snadné rozpoznávání případných nedostatků aplikace bude rozšířena metoda sloužící k nahlašování chyb a její funkčnost v rámci aplikace o logovací informace, které byly použity pro ladění při vývoji. Tento přístup lze pozorovat u některých moderních android aplikací.

V rámci celého systému eLogika je třeba přeložit texty využívané všemi aplikacemi, aby mohly být použitelné i pro zahraniční studenty. Aplikace je na různé jazykové mutace připravena, ale neobsahuje překlady do jiných jazyků než češtiny. Výchozím jazykem by se pak měla stát angličtina.

Další vývoj by se měl věnovat také vzhledu, neboť stávající řešení nesplňuje podmínky Material Designu, kterému se práce pouze okrajově věnuje. Aplikace by pak měla být dodána do obchodu Play společnosti Google, odkud by ji získávali studenti a používali k výuce předmětů využívajících systém eLogika.

6 Použitá literatura

[1] KLEMENT, Milan, CHRÁSKA Miroslav, DOSTÁL Jiří, MAREŠOVÁ Hana. E-learning - elektronické studijní opory a jejich hodnocení. Olomouc: Gevak, 2012. 341. ISBN 978-80-86768-38-0

[2] Download Android Studio and SDK Tools | Android Developers. http://developer.android.com. [online]. [cit. 2016-04-14]. Dostupné z: http://developer.android.com/sdk/index.html

[3] ISO 639-2 Language Code List – Codes for the representation of names of languages (Library of Congress). http://www.loc.gov/. [online]. [cit. 2016-04-21]. Dostupné z: http://www.loc.gov/standards/iso639-2/php/code_list.php

[4] AUTOR NEUVEDEN. http://www.uml-diagrams.org/. [online]. [cit. 2016-04-16]. Dostupný na WWW: http://www.uml-diagrams.org/uml-25-diagrams.png

[5] AUTOR NEUVEDEN. http://developer.android.com/ [online]. [cit. 13.4.2016]. Dostupný na WWW: http://developer.android.com/images/activity_lifecycle.png

[6] AUTOR NEUVEDEN. http://developer.android.com/ [online]. [cit. 13.4.2016]. Dostupný na WWW: http://developer.android.com/images/activity_fragment_lifecycle.png

[7] Activity | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-13]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html

[8] Fragments | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-13]. Dostupné z: http://developer.android.com/guide/components/fragments.html

[9] LACKO, Ľuboslav. Vývoj aplikací pro Android. 1. vyd. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[10] Welcome to the New Google Design – Articles – Google Design. https://design.google.com. [online]. [cit. 2016-04-13]. Dostupné z: https://design.google.com/articles/welcome-to-the-new-google-design/

[11] Support Library Features | Android Developers. https://developer.android.com. [online]. [cit. 2016-04-13]. Dostupné z: https://developer.android.com/tools/support-library/features.htm

[12] Introduction – Material design – Google design guidelines. https://www.google.com. [online]. [cit. 2016-04-13]. Dostupné z: https://www.google.com/design/spec/material-design/introduction.html

[13] Uvod_do_softwaroveho_inzenyrstvi.pdf. http://vondrak.cs.vsb.cz/. [online]. [cit. 2016-04-14]. Dostupné z: http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

[14] UML 2.5 Diagrams Overview http://www.uml-diagrams.org/. [online]. [cit. 2016-04-16]. Dostupné z: http://www.uml-diagrams.org/uml-25-diagrams.html

[15] MANĚNA, Václav. Moderně s Moodlem: jak využít e-learning ve svůj prospěch. Praha: CZ.NIC, z.s.p.o., 2015. CZ.NIC. ISBN 978-80-905802-7-5.

[16] KLEMENT, Milan. E-learning: elektronické studijní opory a jejich hodnocení. 1. vyd. Olomouc: Agentura Gevak, 2012. ISBN 978-80-86768-38-0.

[17] RESTful_Best_Practices-v1_1.pdf. http://www.restapitutorial.com. [online]. [cit. 2016-04-18]. Dostupné z: http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf

[18] JSON. http://www.json.org/. [online]. [cit. 2016-04-18]. Dostupné z: http://www.json.org/

[19] Free Mercurial and Git Client for Windows and Mac | Atlassian SourceTree. https://www.sourcetreeapp.com/. [online]. [cit. 2016-04-21]. Dostupné z: https://www.sourcetreeapp.com/

[20] Thank You – Genymotion Android Emulator | Free Version. https://www.genymotion.com/. [online]. [cit. 2016-04-21]. Dostupné z: https://www.genymotion.com/thank-you-freemium/

[21] Android* – Intel® Hardware Accelerated Execution Manager | Intel® Developer Zone. https://software.intel.com/. [online]. [cit. 2016-04-21]. Dostupné z: https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager

[22] Transmitting Network Data Using Volley | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/training/volley/index.html

[23] Saving Data in SQL Databases | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/training/basics/data-storage/databases.html

[24] Saving Key-Value Sets | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/training/basics/data-storage/shared-preferences.html

[25] Support Library | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/tools/support-library/index.html

[26] List View | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/listview.html

[27] Creating Lists and Cards | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/training/material/lists-cards.html

[28] Linear Layout | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/linear.html

[29] Table | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/guide/topics/ui/layout/grid.html

[30] Adding Swipe-to-Refresh To Your App | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/training/swipe/add-swipe-interface.html

[31] Using ViewPager for Screen Slides | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/training/animation/screen-slide.html

[32] SecretKeySpec (Java Platform SE 7). https://docs.oracle.com/. [online]. [cit. 2016-04-21]. Dostupné z: https://docs.oracle.com/javase/7/docs/api/javax/crypto/spec/SecretKeySpec.html

[33] Cipher (Java Platform SE 7). https://docs.oracle.com/. [online]. [cit. 2016-04-21]. Dostupné z: https://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html

[34] Support Library Features | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/tools/support-library/features.html

[35] Layouts | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews

[36] View | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/reference/android/view/View.html

[37] Adapter | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-21]. Dostupné z: http://developer.android.com/reference/android/widget/Adapter.html

[38] Team Foundation Server | Visual Studio. https://www.visualstudio.com/. [online]. [cit. 2016-04-22]. Dostupné z: https://www.visualstudio.com/cs-cz/products/tfs-overview-vs.aspx

[39] BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. 1. vyd. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1075-7.

[40] GAMMA, Erich. Design patterns: elements of reusable object-oriented software. Reading, Mass.: Addison-Wesley, c1995. ISBN 0201633612.

[41] FOWLER, Martin. Patterns of enterprise application architecture. Boston: Addison-Wesley, c2003. Addison-Wesley signature series. ISBN 0-321-12742-0.

[42] Getting Started with ARC – Google Chrome. https://developer.chrome.com/. [online]. [cit. 2016-04-23]. Dostupné z: https://developer.chrome.com/apps/getstarted_arc

[43] GitHub – chrisbanes/PhotoView: Implementation of ImageView for Android that supports zooming, by various touch gestures.. https://github.com/. [online]. [cit. 2016-04-25]. Dostupné z: https://github.com/chrisbanes/PhotoView

[44] GitHub – nispok/snackbar: [DEPRECATED] Android Library that implements Snackbars from Google\'s Material Design documentation.. https://github.com/. [online]. [cit. 2016-04-25]. Dostupné z: https://github.com/nispok/snackbar

[45] GitHub – google/gson: A Java serialization/deserialization library that can convert Java Objects into JSON and back.. https://github.com/. [online]. [cit. 2016-04-25]. Dostupné z: https://github.com/google/gson

[46] Android Debug Bridge | Android Developers. http://developer.android.com/. [online]. [cit. 2016-04-25]. Dostupné z: http://developer.android.com/tools/help/adb.html

7 Seznam příloh

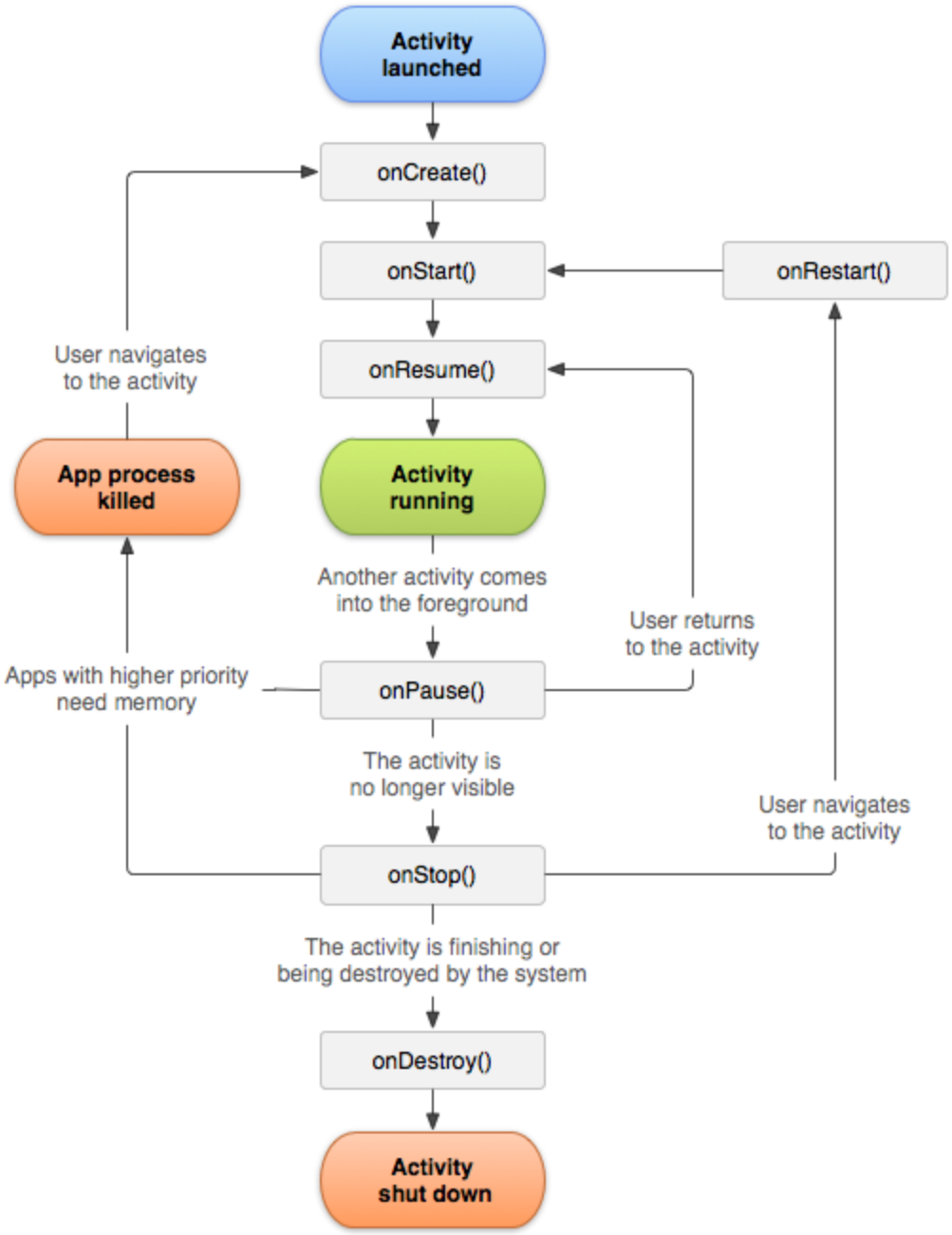
A Větší obrázky odkazované z textu I
B Třídní diagramy VII
C Diagramy případů užití XVI
D Scénář případů užití XVIII
E Diagramy aktivit XIX
F Sekvenční diagramy XXI
G Diagram komunikace XXIX
H Diagram nasazení XXXII
I Ukázky uživatelského rozhraní XXXIII
J Použité REST API XLV

Součástí DP je CD/DVD.

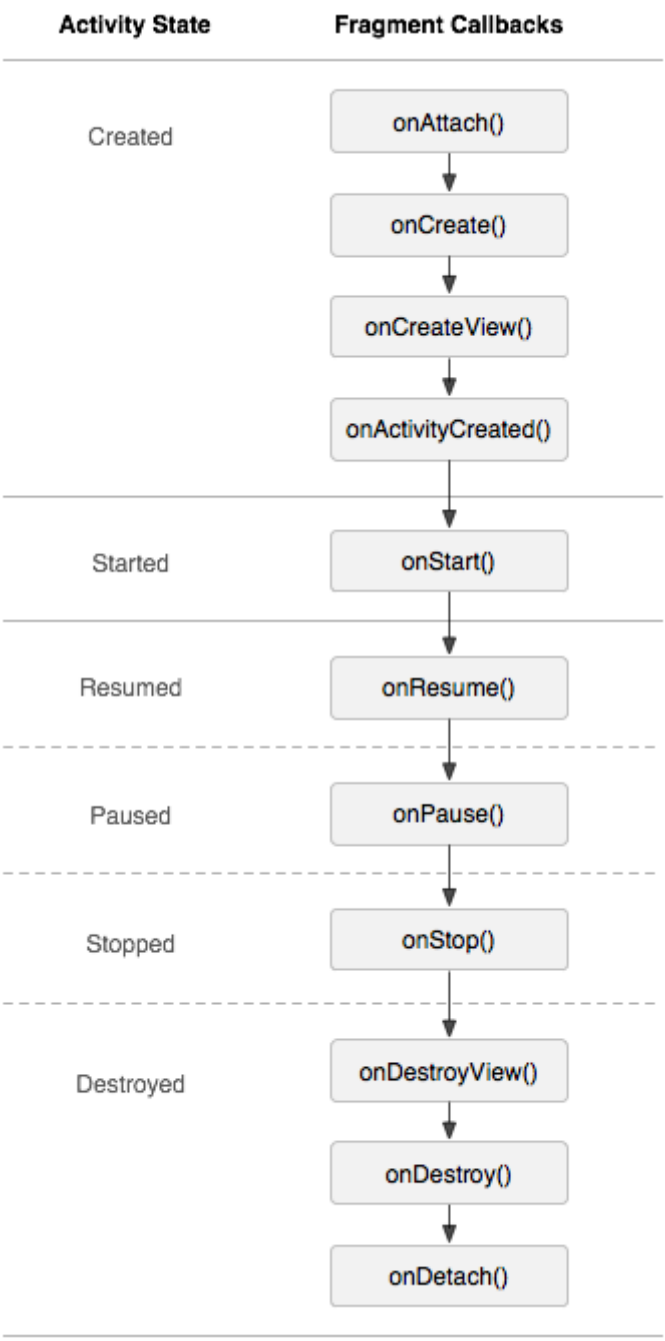
Adresářová struktura přiloženého CD/DVD:

- Diplomová práce
  - JavaDoc – programátorská dokumentace ve formátu HTML
  - ProjektAndroidStudia – zdrojové kódy aplikace eLogika nad platformou Android
  - TextDP – složka obsahující text diplomové práce ve formátu pdf
    - DiplomovaPrace_zno0011.pdf

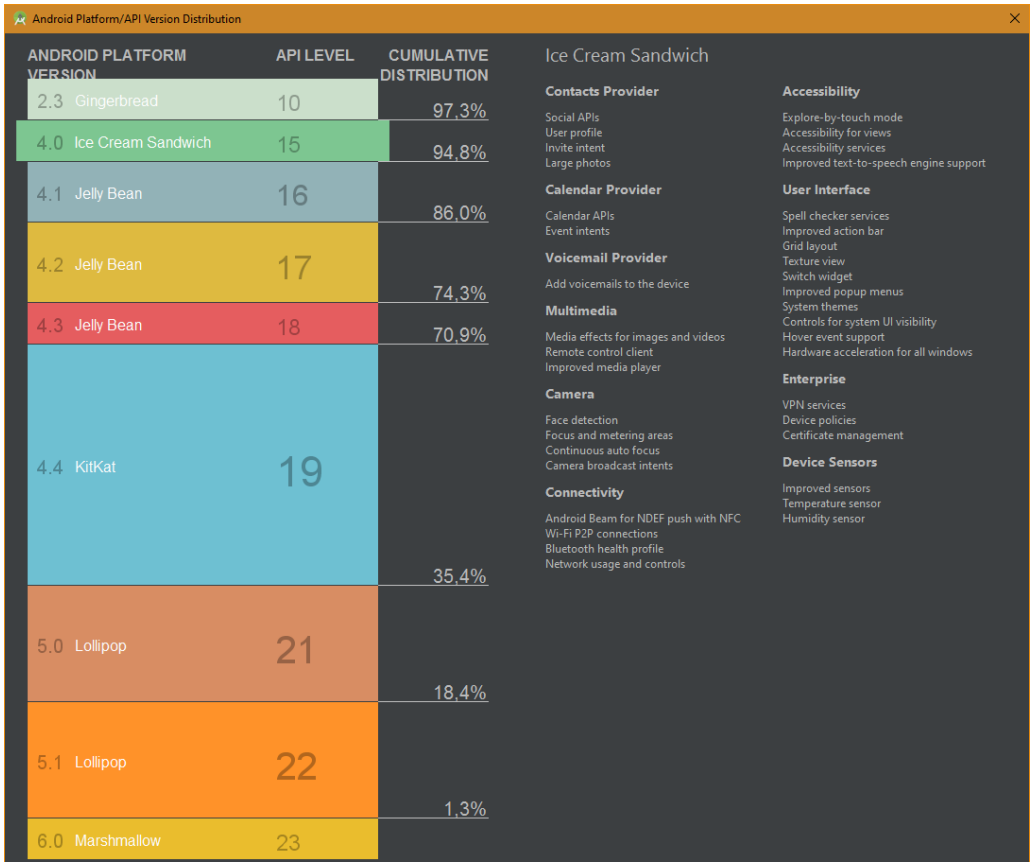
A *Větší obrázky odkazované z textu*



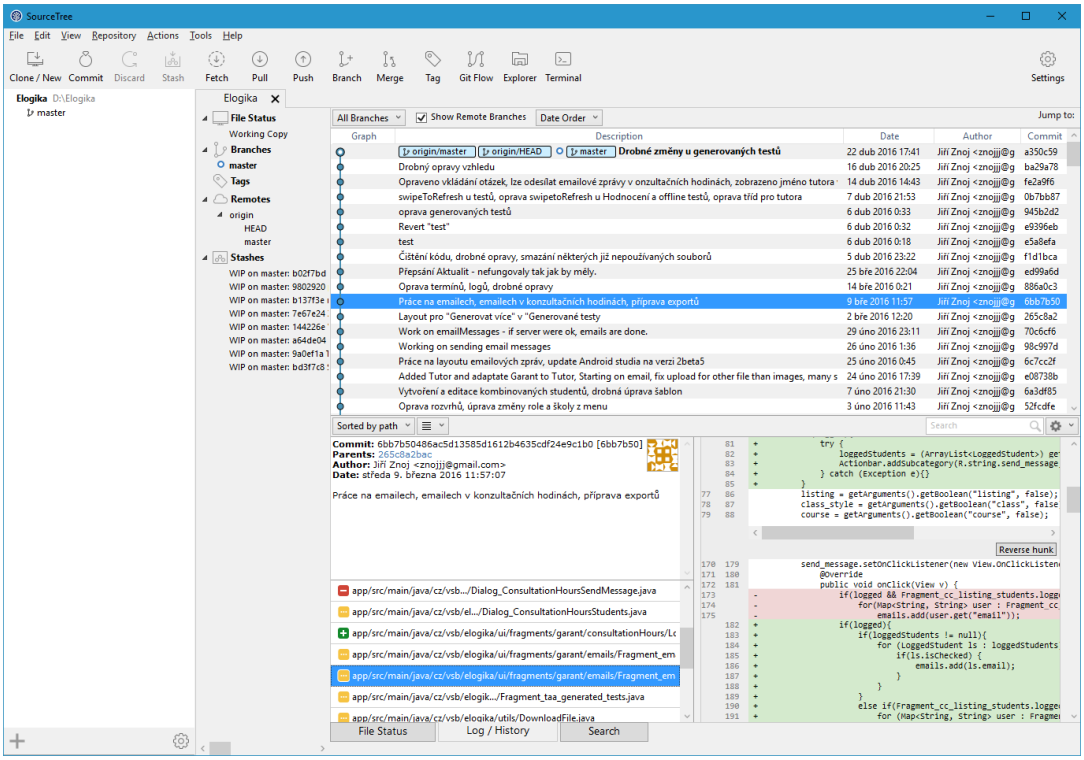
Obrázek 2: *Diagram životního cyklu android aktivity [5]*



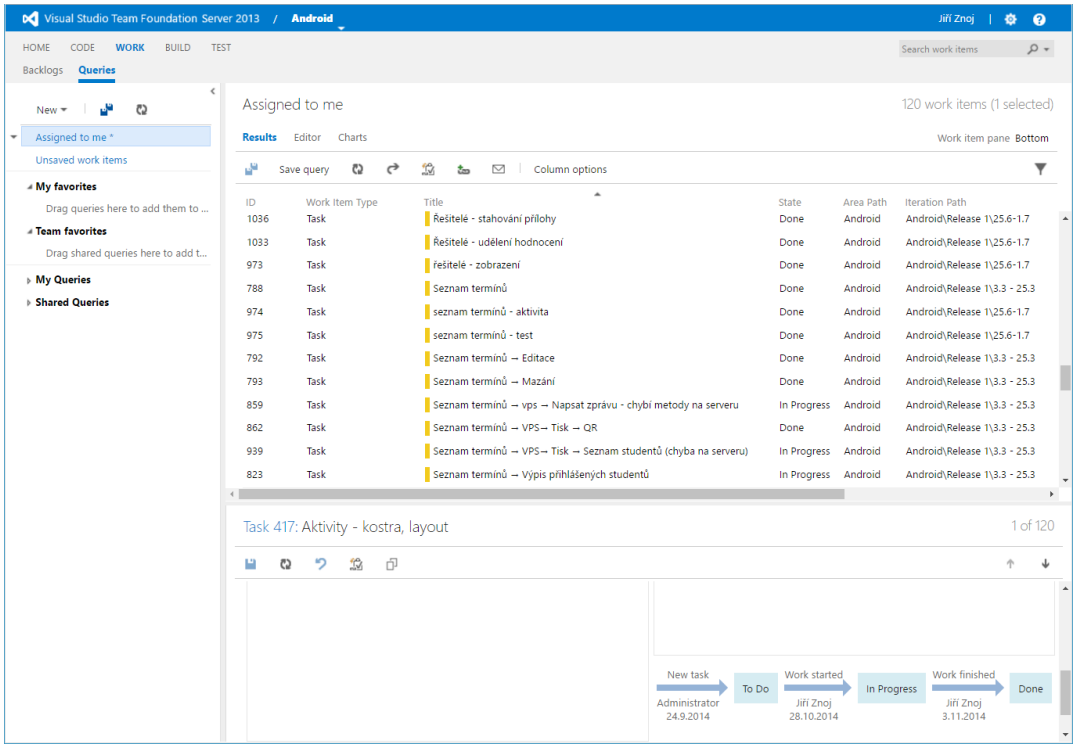
Obrázek 3: *Diagram životního cyklu fragmentu [6]*



Obrázek 4: *Diagram verzí OS Android, verzí API a počet zařízení k 14. 4. 2016 – výřez snímku obrazovky programu Android Studio 2.0, který lze stáhnout zde: [2]*

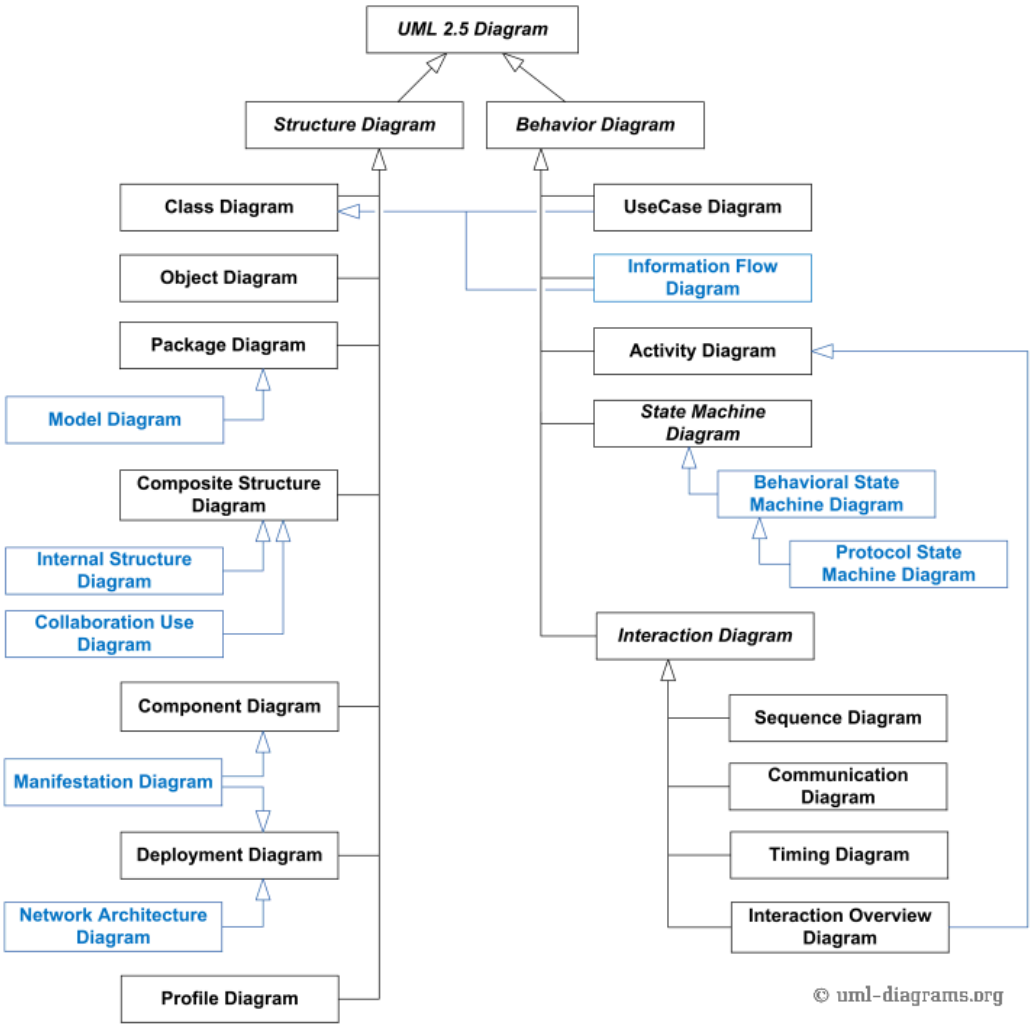


Obrázek 5: *Ukázka programu SourceTree stažitelného zde: [19]*



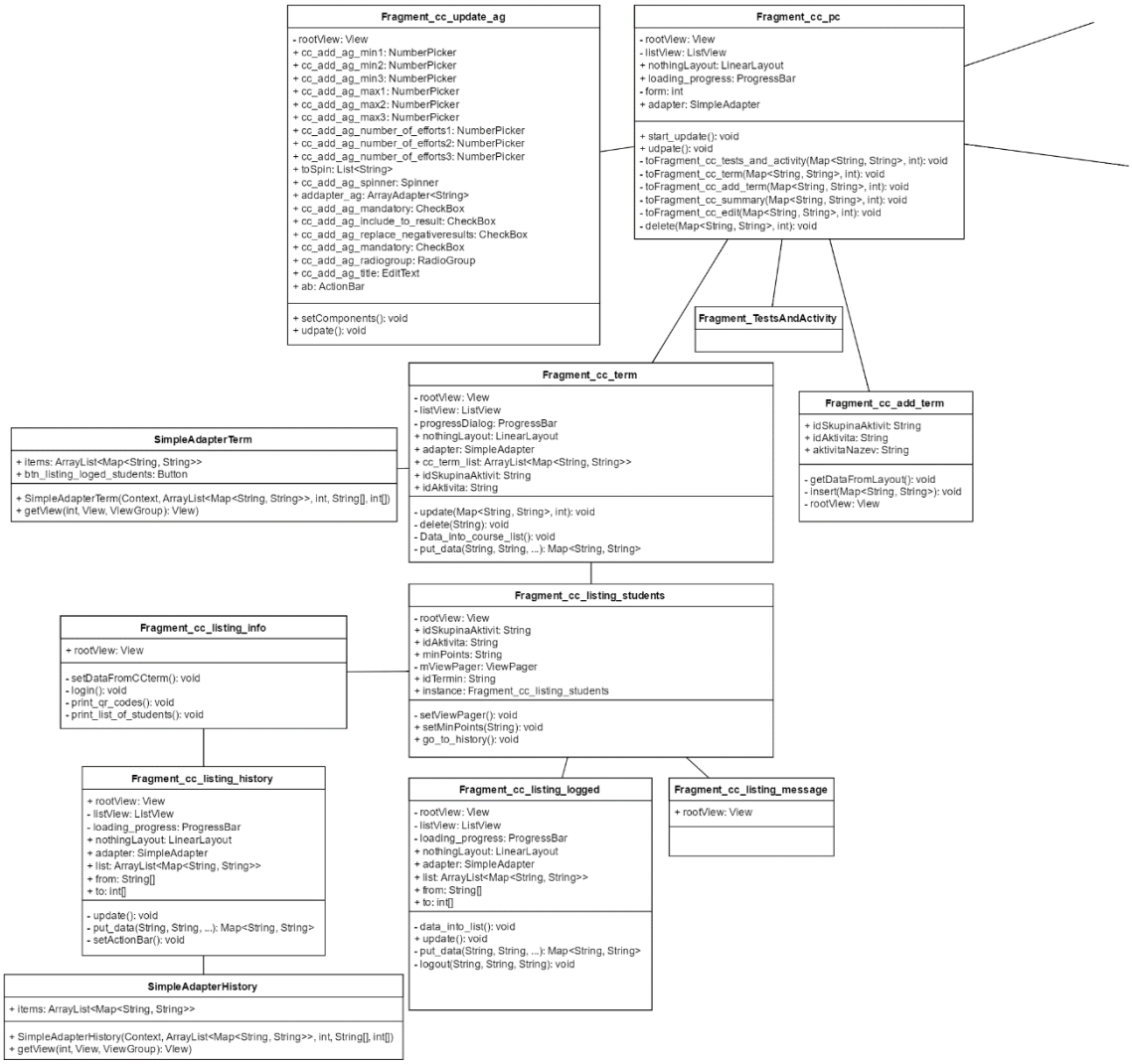
Obrázek 6: *Ukázka TFS dostupného na adrese http://elogika.vsb.cz:8080/tfs/eLogika%20Collection/Android*

Obrázek 7: *Ukázka programátorské dokumentace JavaDoc*

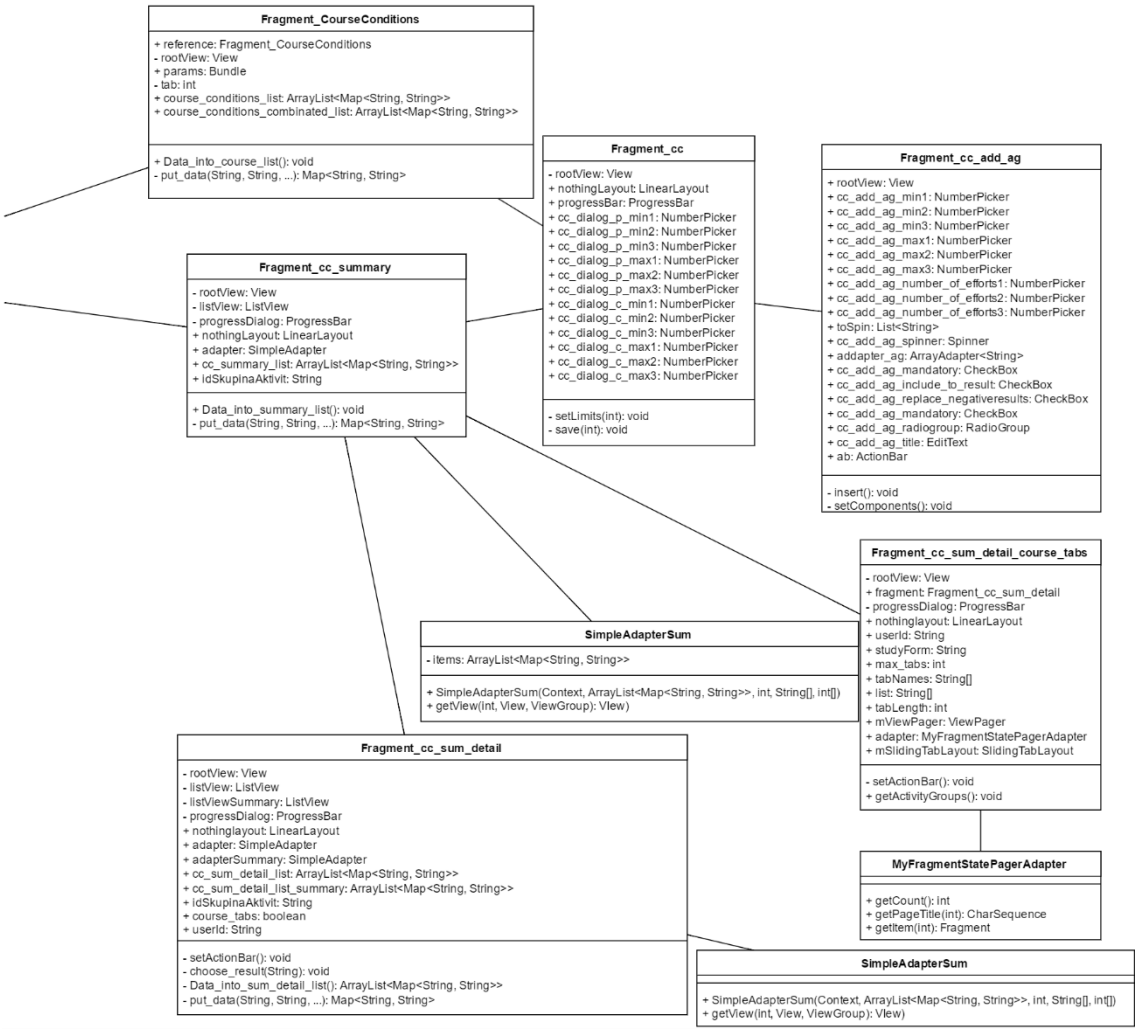


Obrázek 8: *Diagram ukázky klasifikace UML diagramů UML 2.5 [3]*

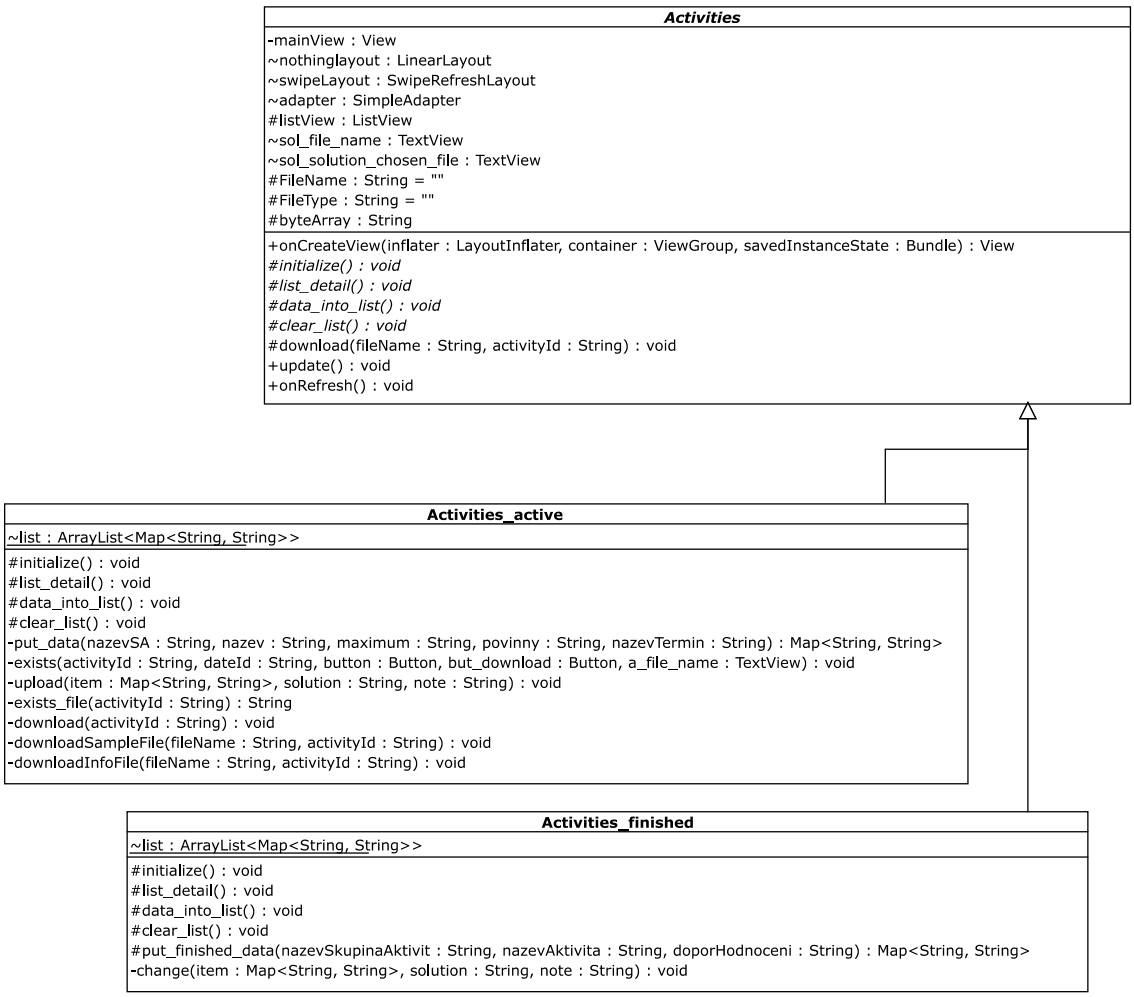
B *Třídní diagramy*



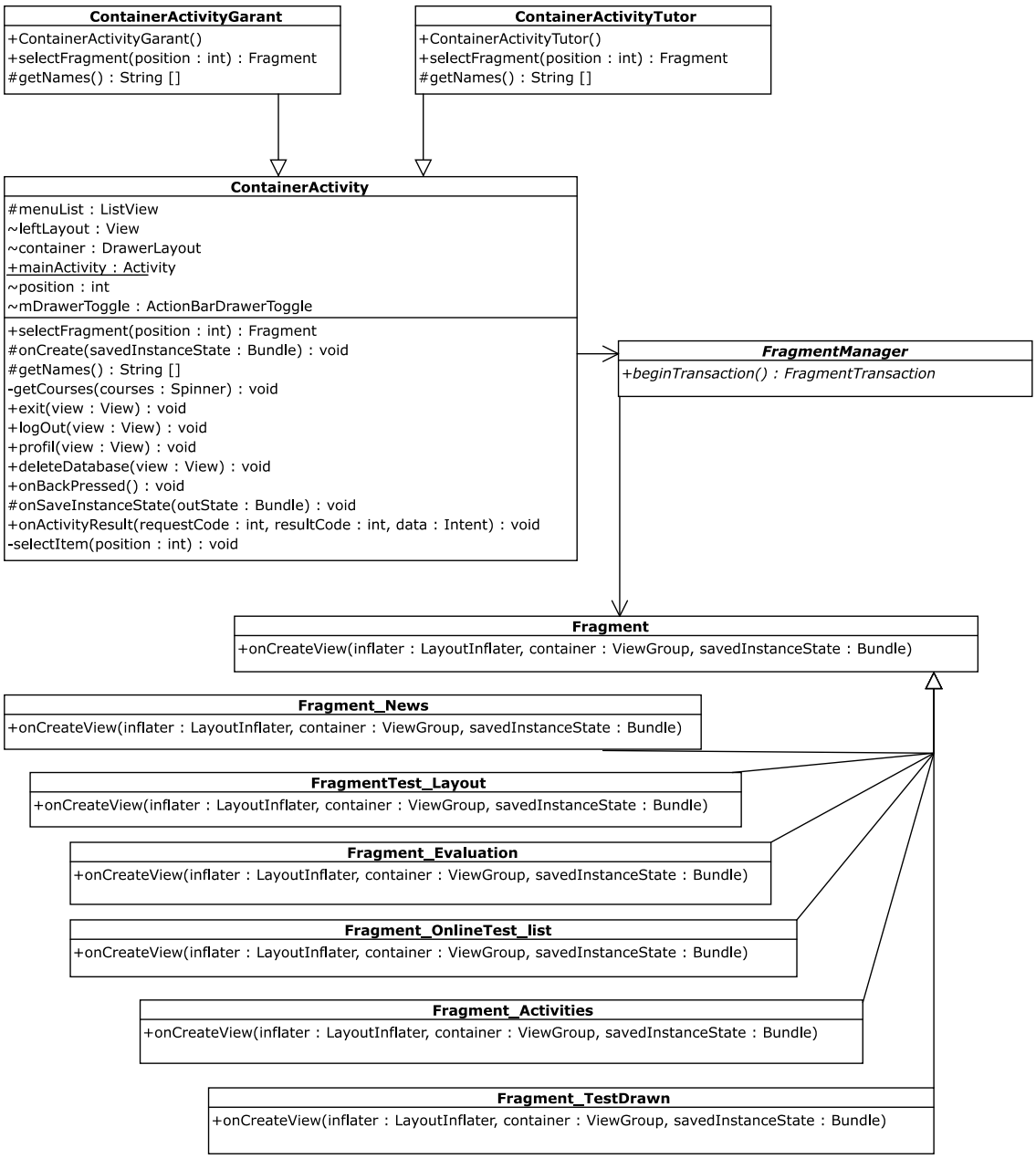
Obrázek 9: *Levá polovina třídního diagramu pro případ užití „Zavedení podmínek kurzu“ pro uživatele v roli garanta*



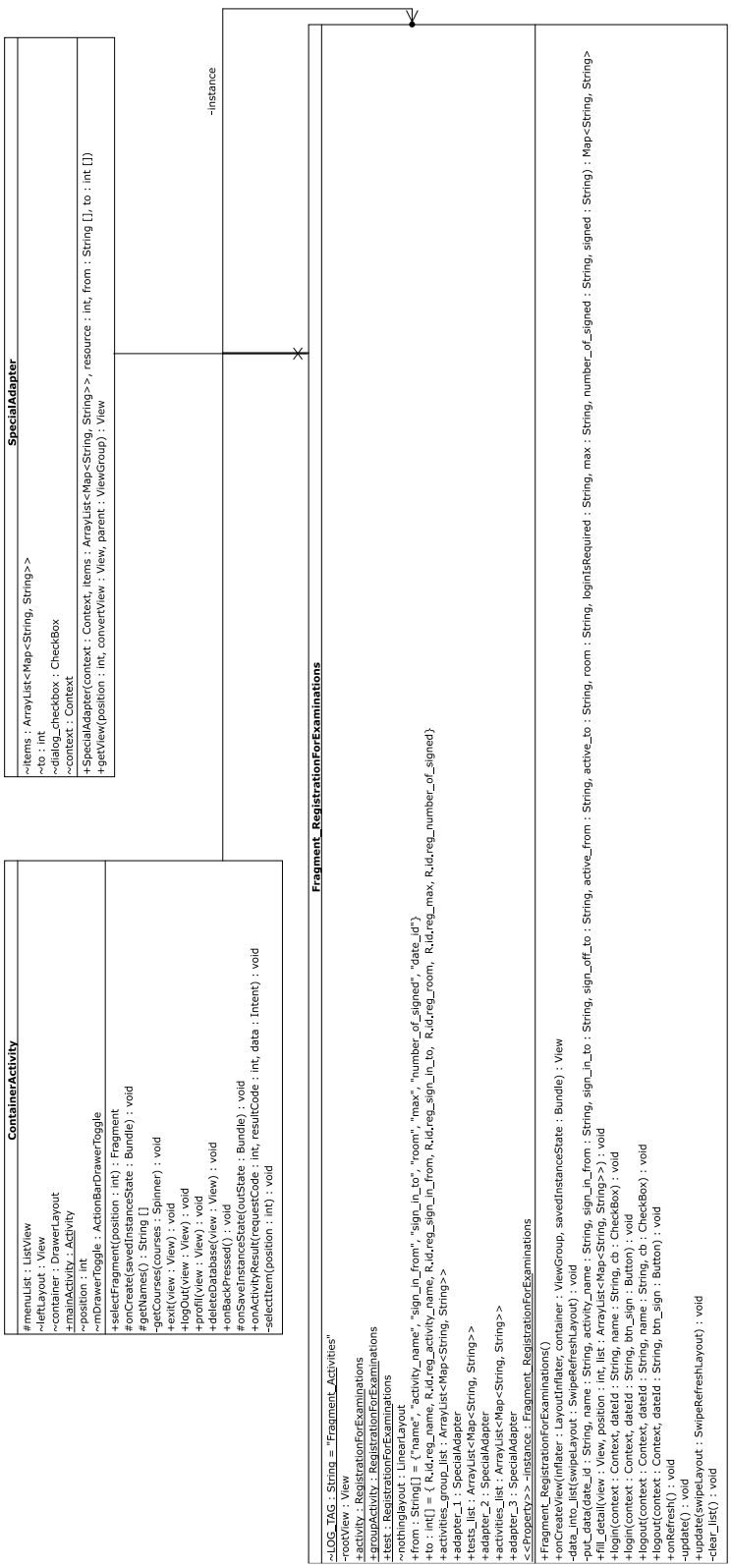
Obrázek 10: *Pravá polovina třídního diagramu pro případ užití „Zavedení podmínek kurzu“ pro uživatele v roli garanta*



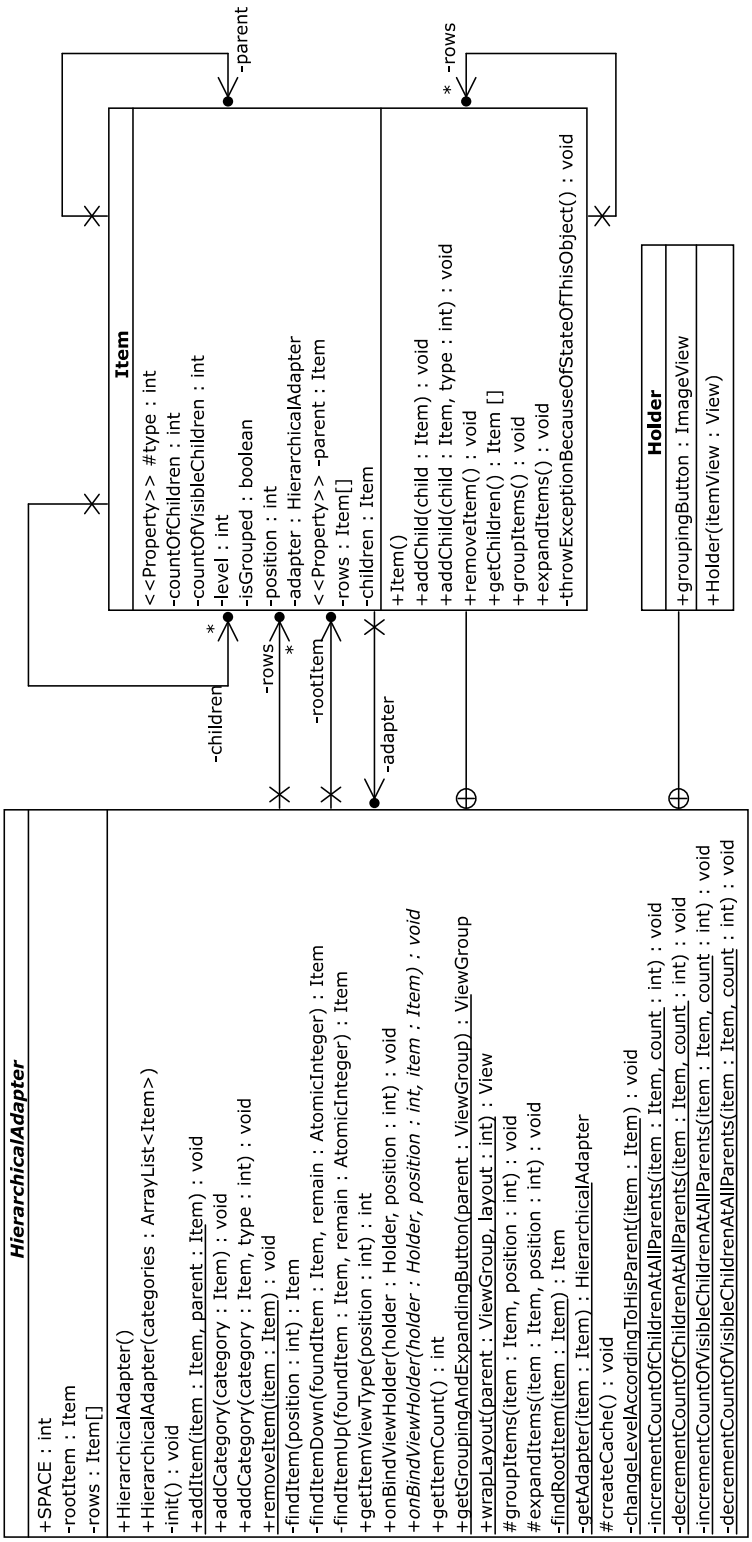
Obrázek 11: *Třídní diagram návrhového vzoru Šablona (Template method)*



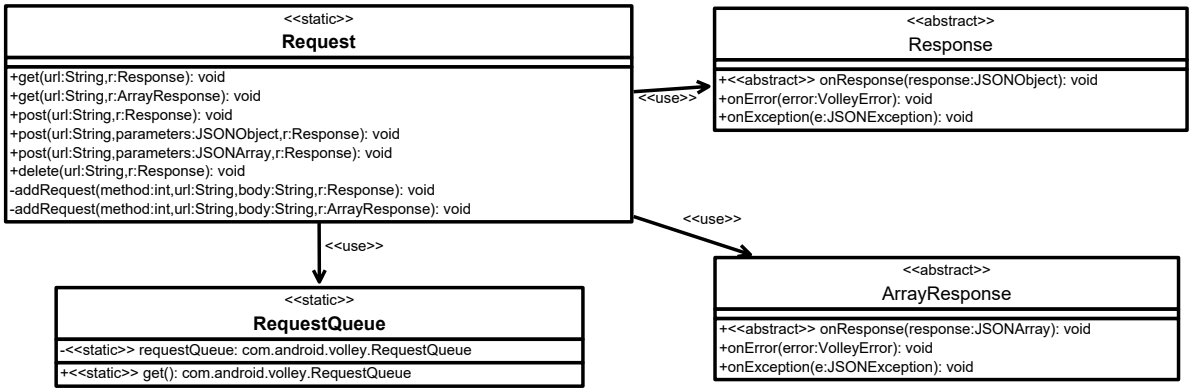
Obrázek 12: *Třídní diagram návrhového vzoru Továrna (Factory method)*



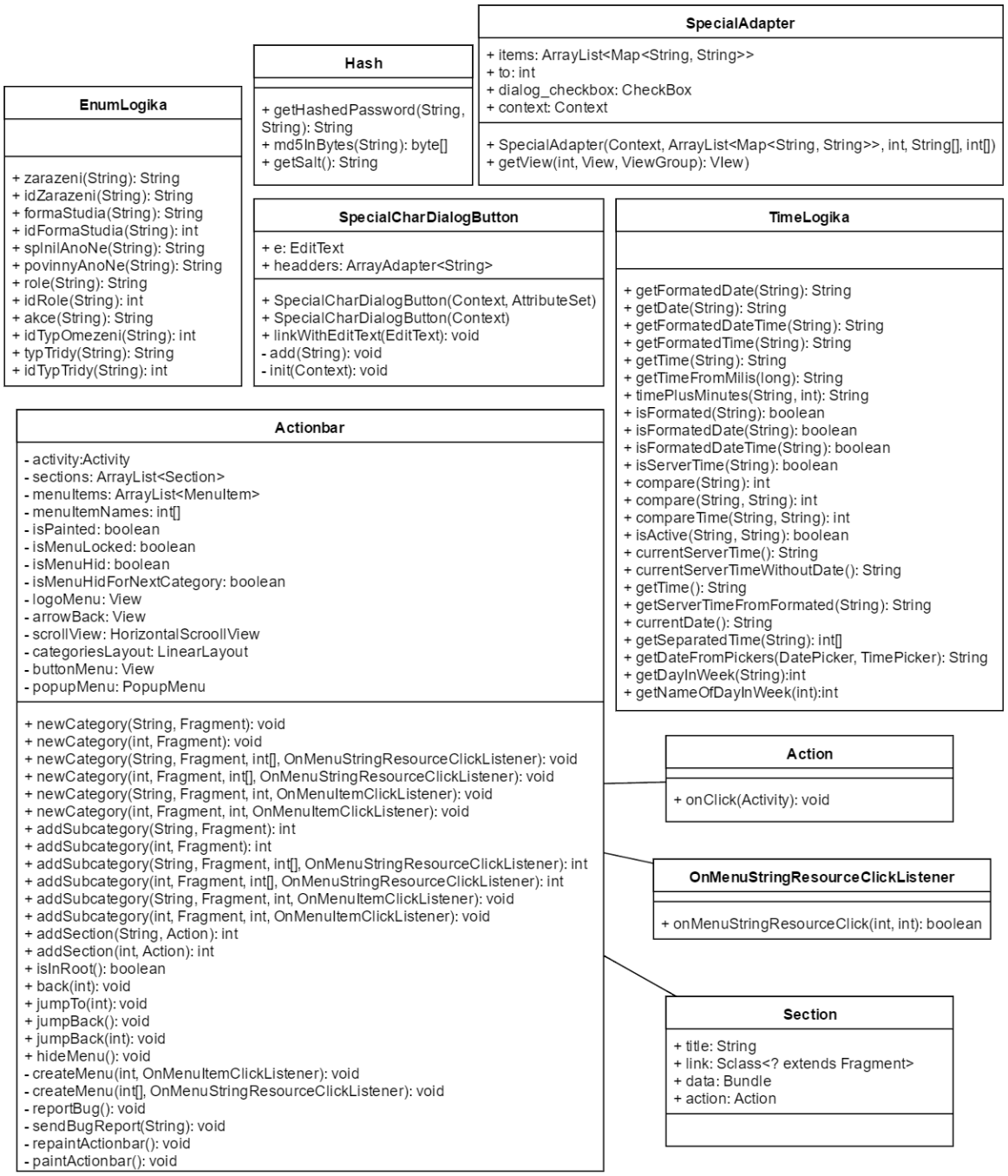
Obrázek 13: *Třídní diagram návrhového vzoru Jedináček (Singleton)*



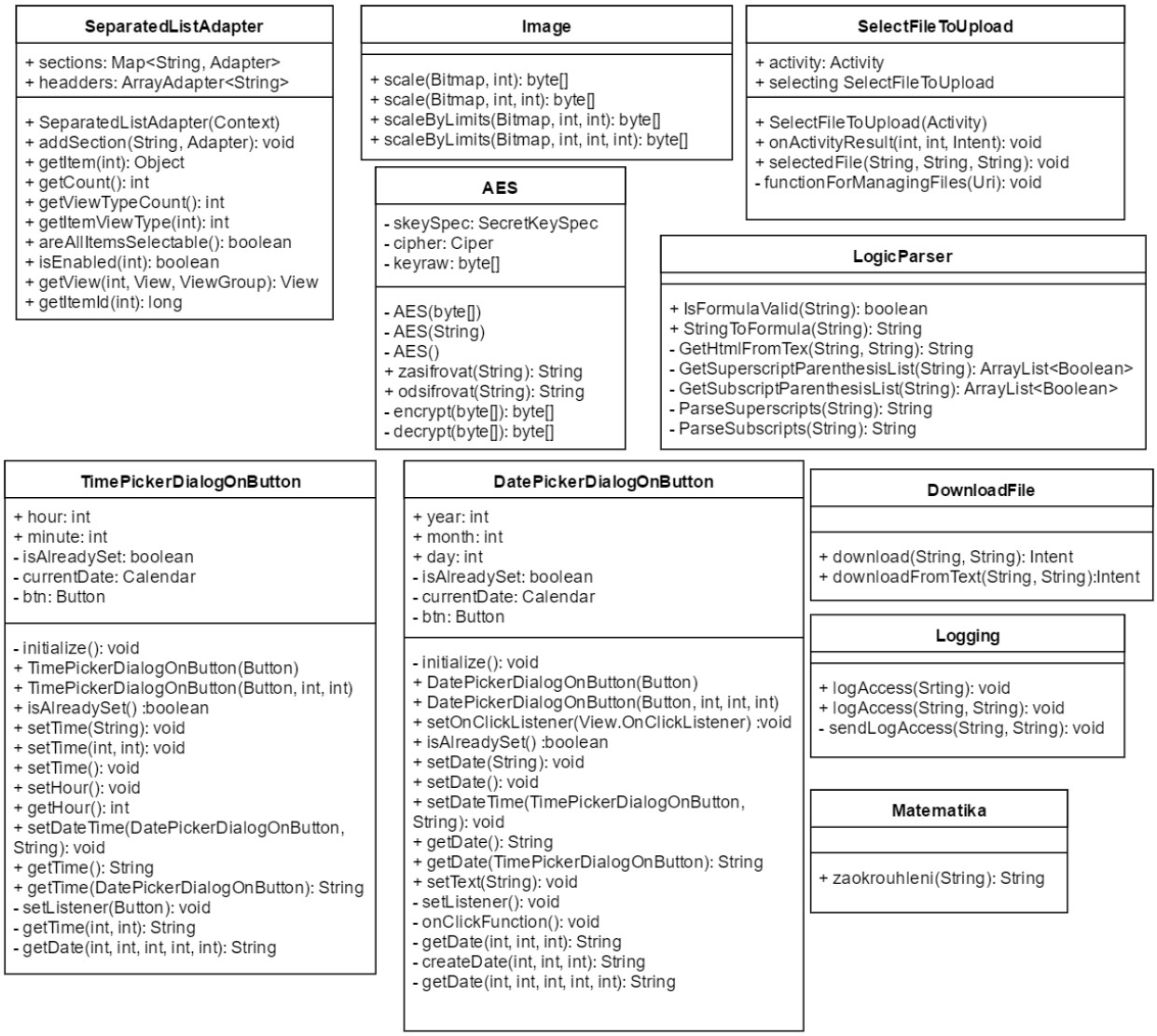
Obrázek 14: *Třídní diagram Hierarchického adaptéru se třídou Item demonstrující návrhový vzor Kompozit (Composite)*



Obrázek 15: *Třídní diagram komunikace s knihovnou Volley*

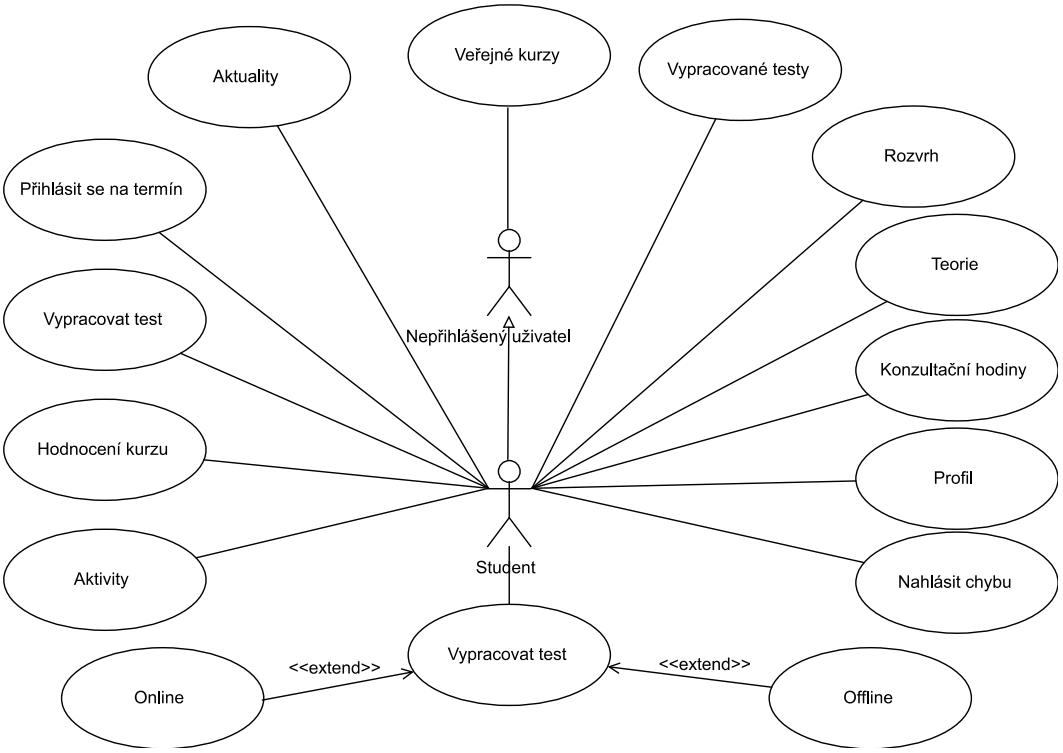


Obrázek 16: *Třídní diagram některých pomocných tříd aplikace eLogika*

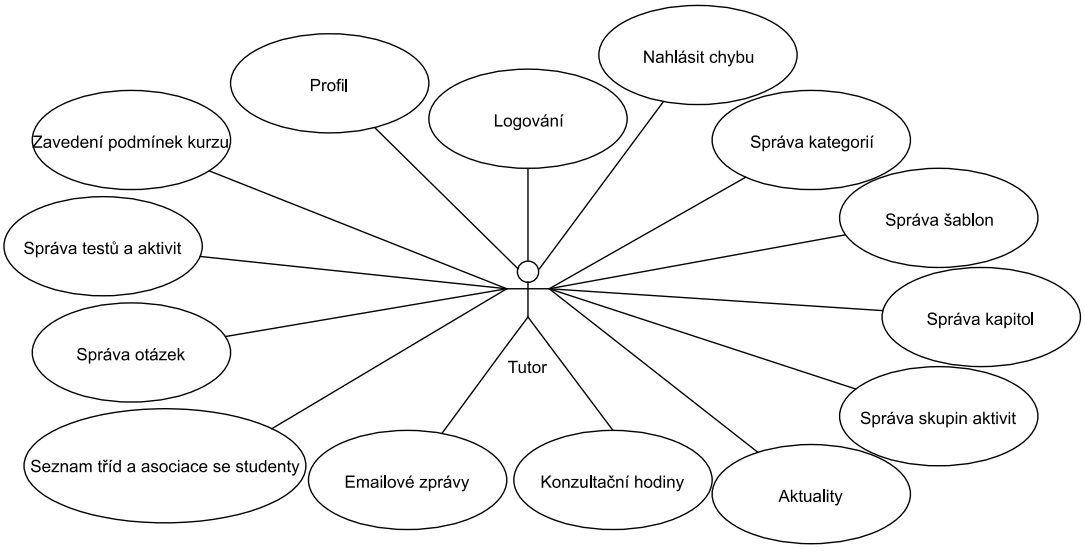


Obrázek 17: *Třídní diagram některých pomocných tříd aplikace eLogika*

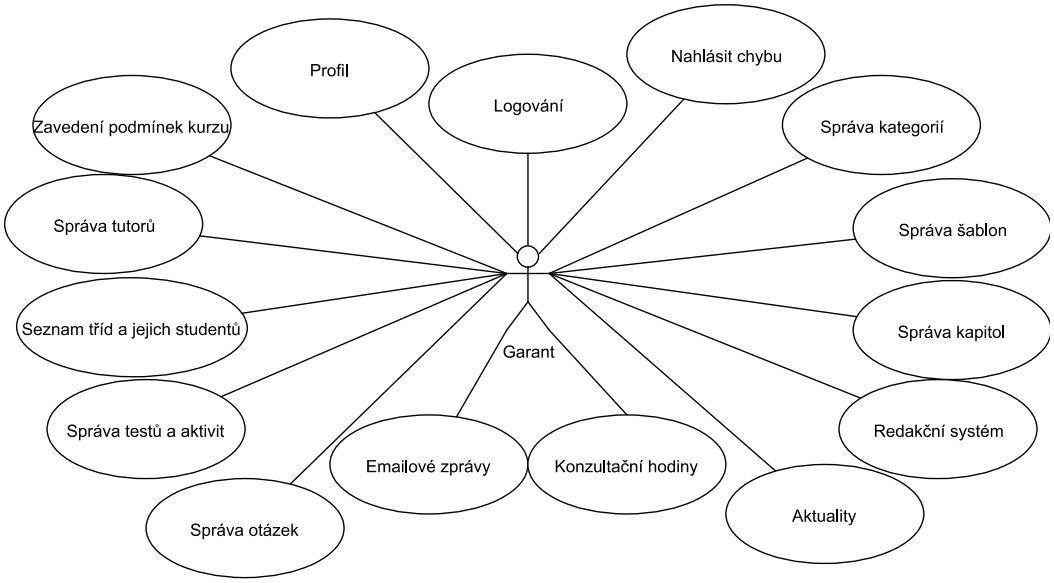
C *Diagramy případů užití*



Obrázek 18: *Diagram případů užití pro nepřihlášeného uživatele a studenta*



Obrázek 19: *Diagram případů užití pro uživatele v roli tutora*



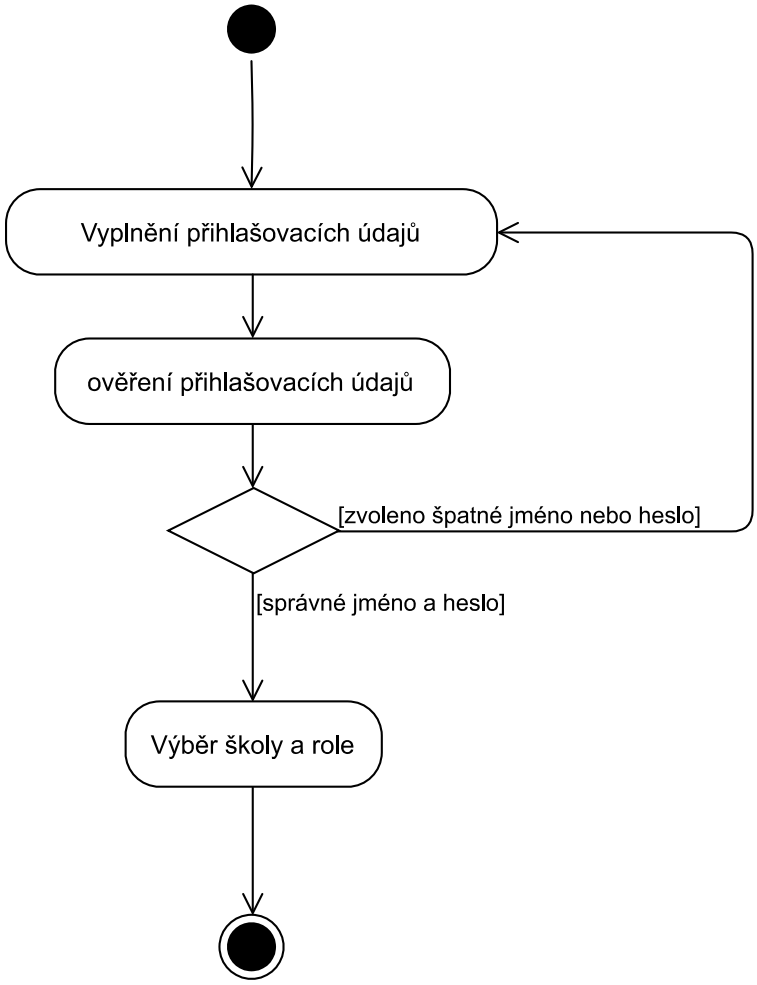
Obrázek 20: *Diagram případů užití pro uživatele v roli garanta*

D *Scénář případů užití*

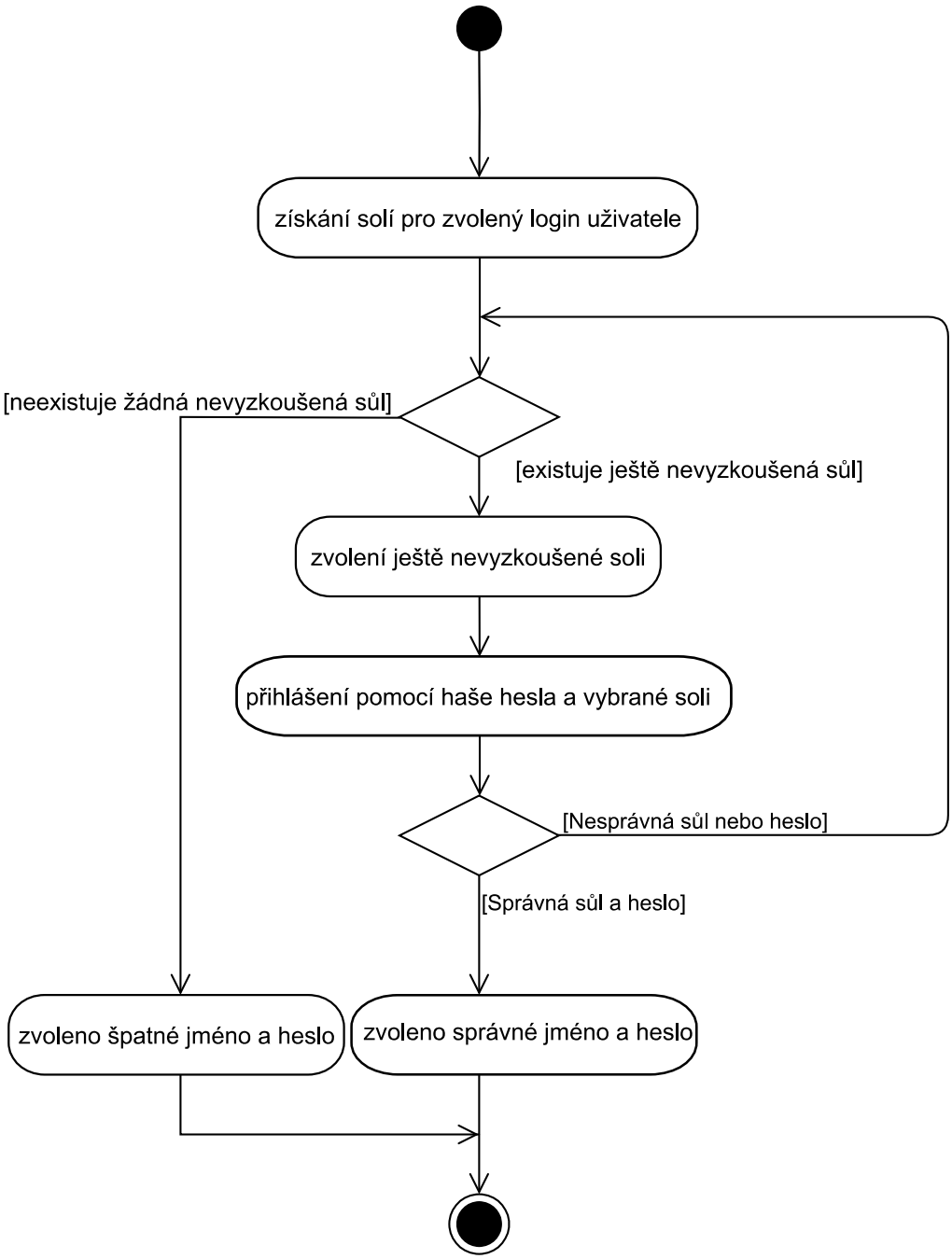
Tabulka 1: *Scénář případů užití pro případ užití „Nahlásit chybu“*

Aktéři	*Student, Garant, Tutor*
Předpoklady	*Přihlášení do systému, přístup k internetu*
Průběh	***1) aplikace** zobrazí lištu s možností „Nahlásit chybu“* ***2) uživatel** může zvolit v liště možnost „Nahlásit chybu“* ***3) aplikace** zobrazí dialogové okno s možností popisu chyby a jejího odeslání nebo zavření dialogového okna* ***4a) uživatel** popíše chybu a zvolí tlačítko pro její odeslání* ***5) aplikace** zavře dialogové okno a informuje uživatele o tom, zda byla chyba v pořádku odeslána či nikoliv*
Výjimky	- internetové připojení není aktivní v době odesílání chyby na server ***3b) aplikace** uzavře dialogové okno a informuje uživatele, že spojení se serverem selhalo* - internetové připojení je aktivní, ale server neodpovídá ***3c) aplikace** uzavře dialogové okno a informuje uživatele, že se nepodařilo nahlásit chybu*
Alternativní průběh	***4b) uživatel** zvolí možnost pro uzavření dialogového okna a chybu nenahlásí* ***5) aplikace** uzavře dialogové okno*
Alternativní průběh	***4c) uživatel** nepopíše chybu a zvolí tlačítko pro odeslání chyby* ***5) aplikace** uzavře dialogové okno*

E *Diagramy aktivit*

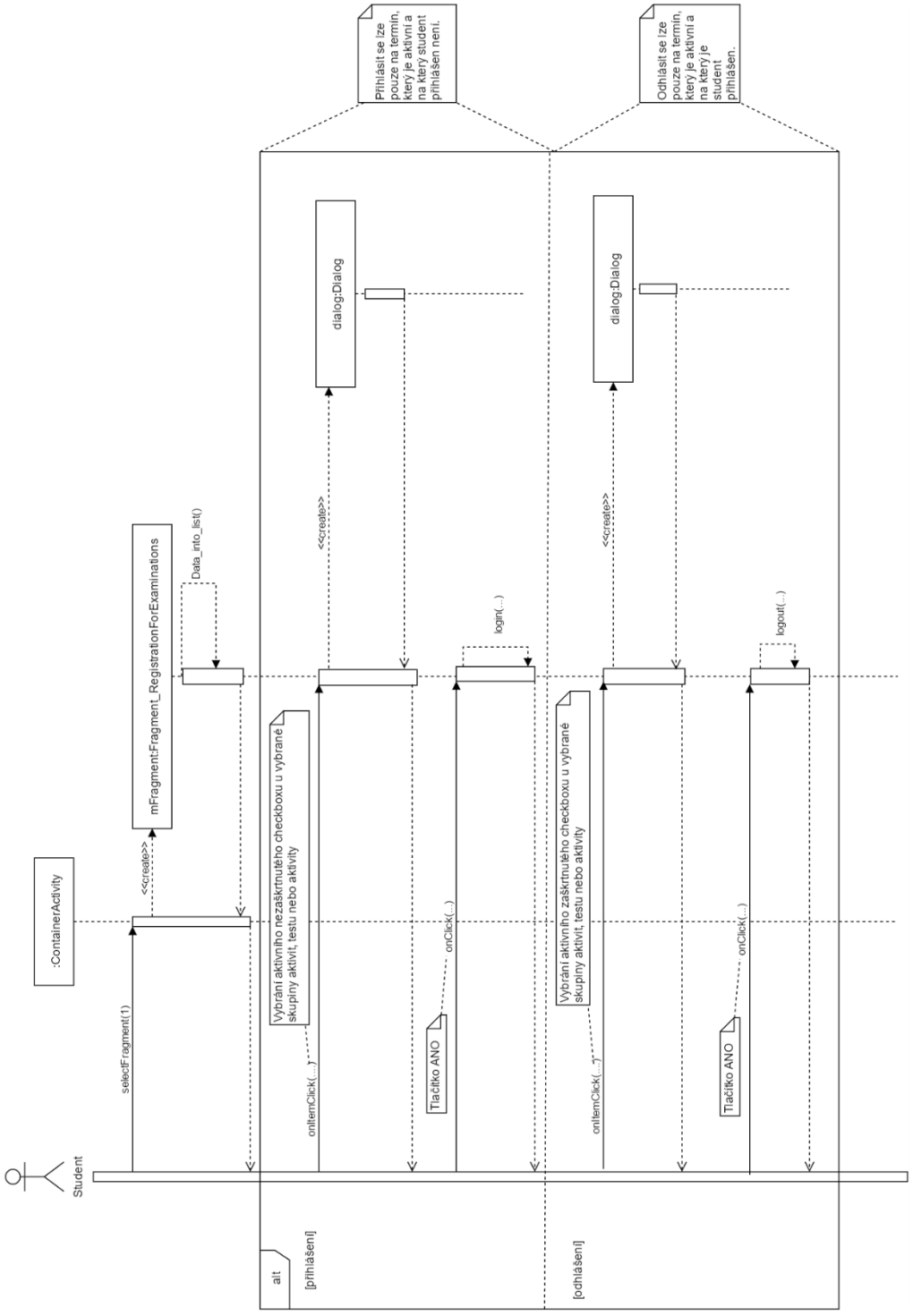


Obrázek 21: *Diagram aktivit procesu přihlášení*

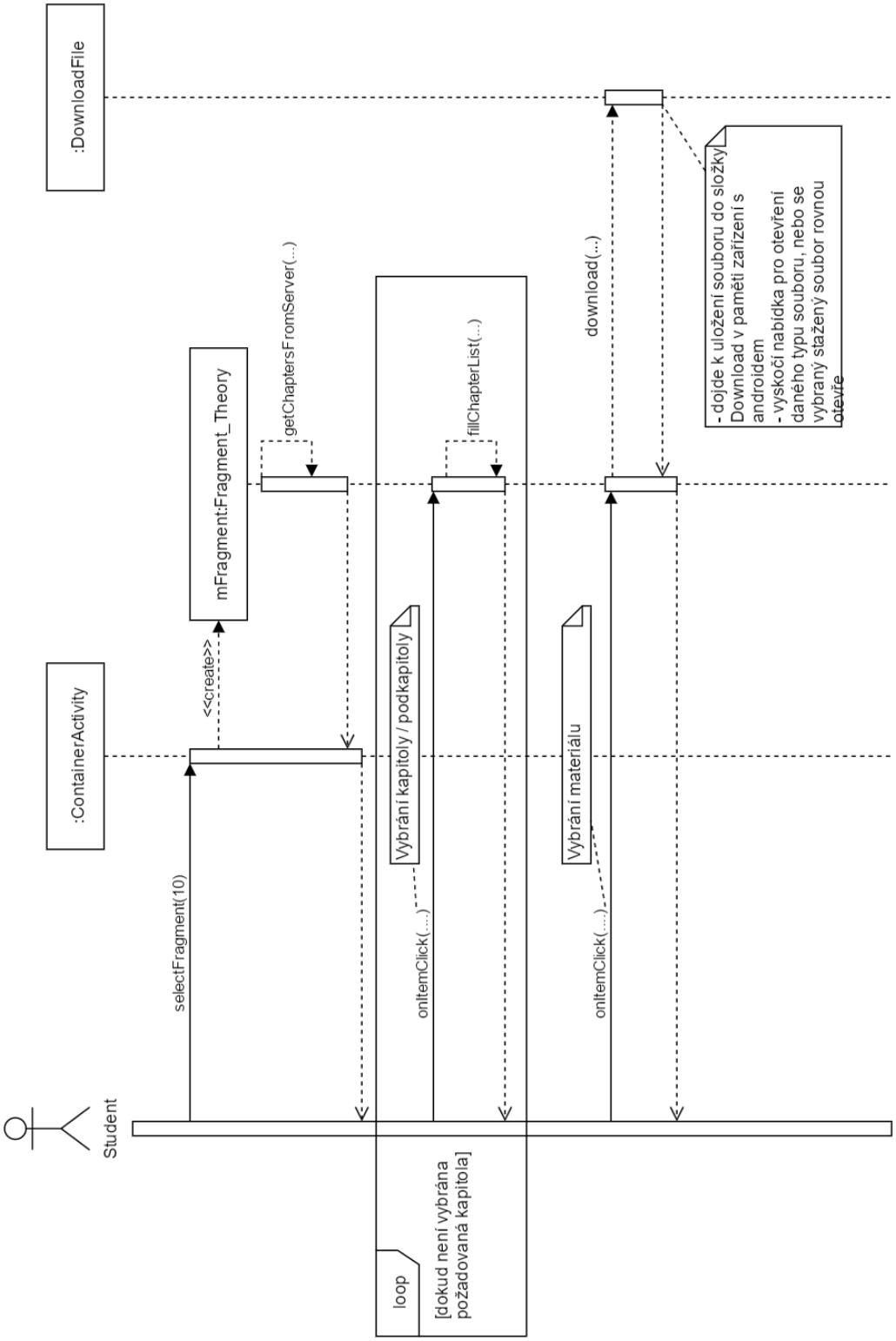


Obrázek 22: *Aktivitní diagram pro ověření přihlašovacích údajů*

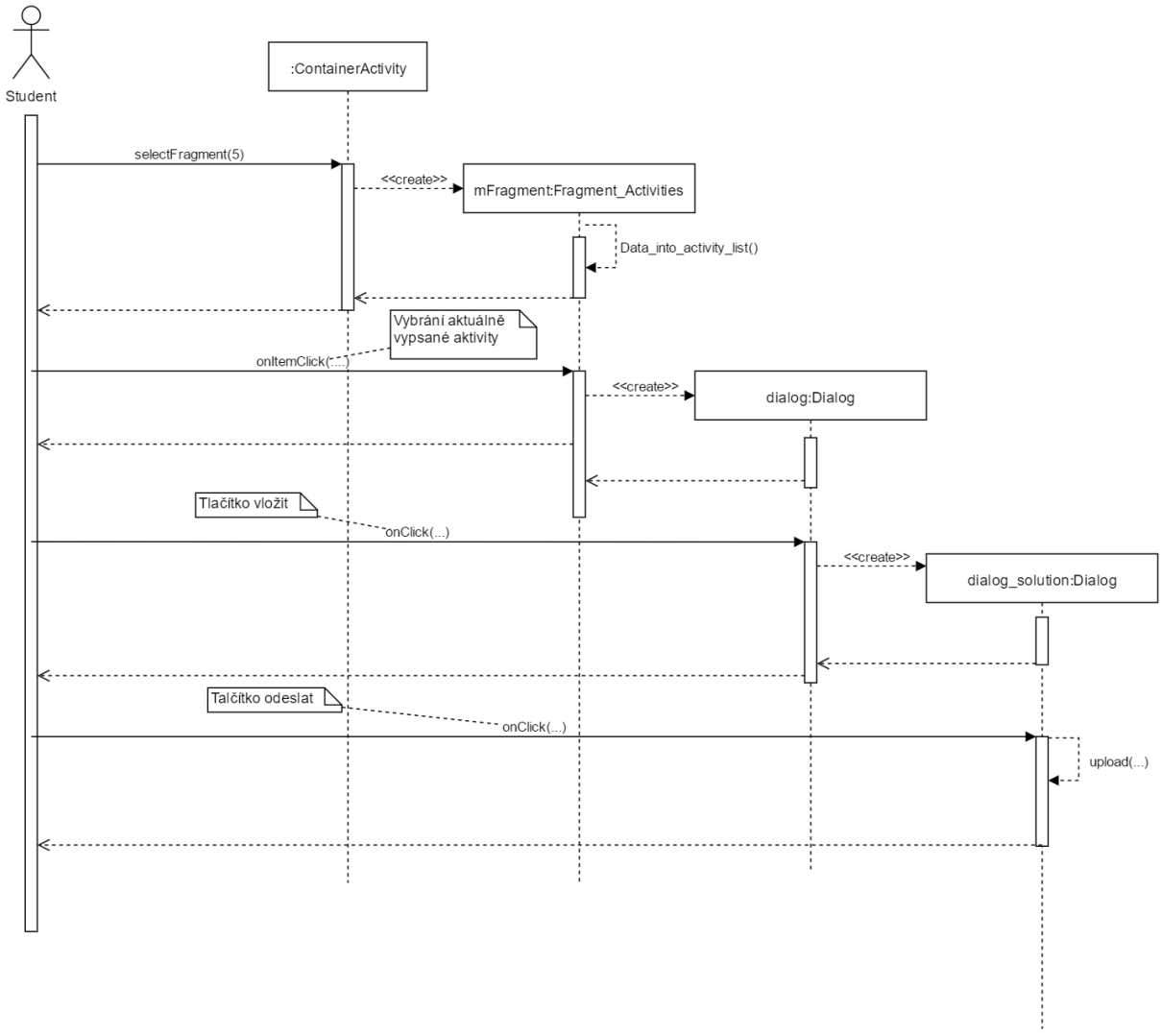
F *Sekvenční diagramy*



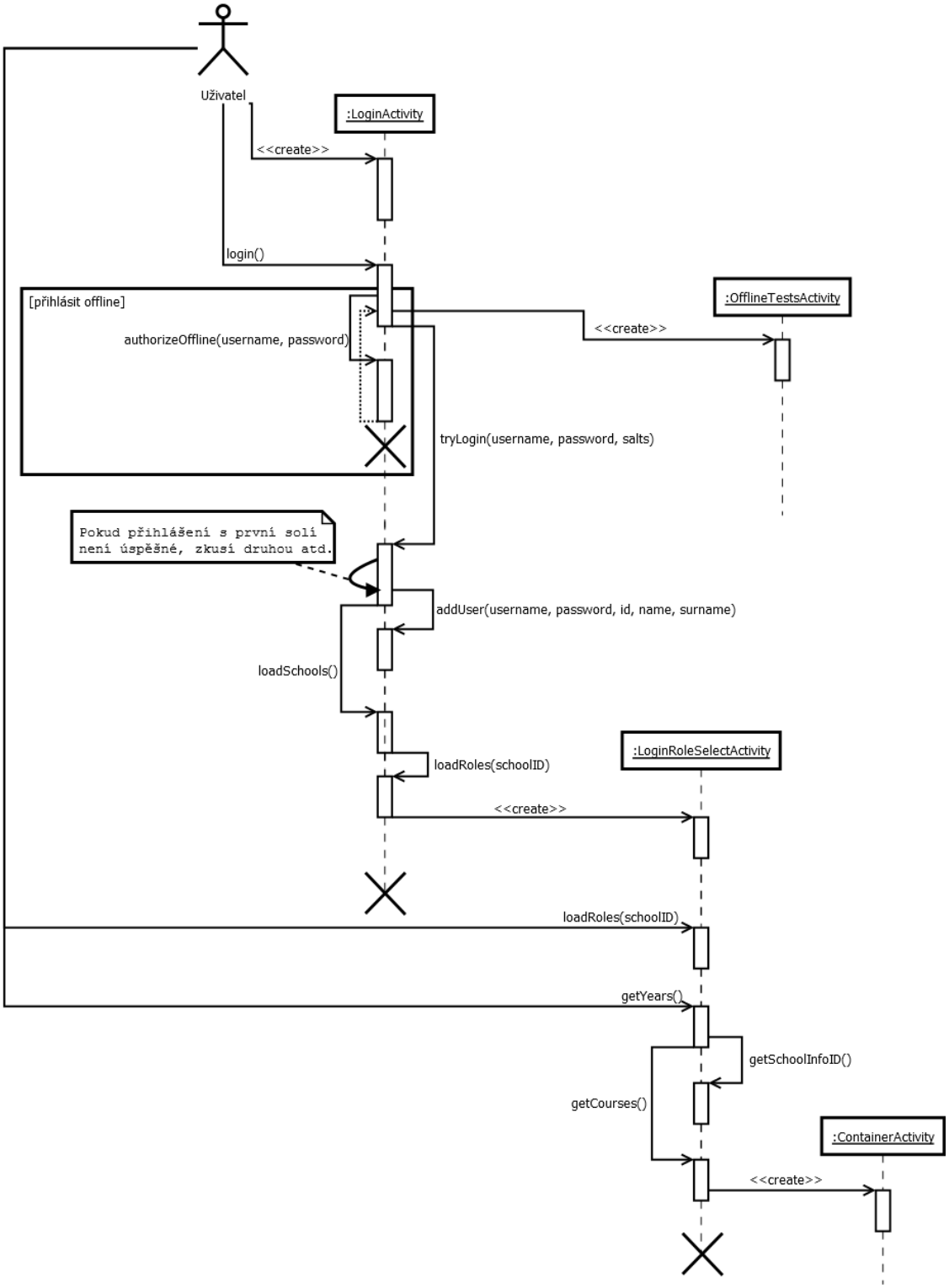
Obrázek 23: *Sekvenční diagram popisující přihlášení na termín a odhlášení z termínu uživatele v roli studenta*



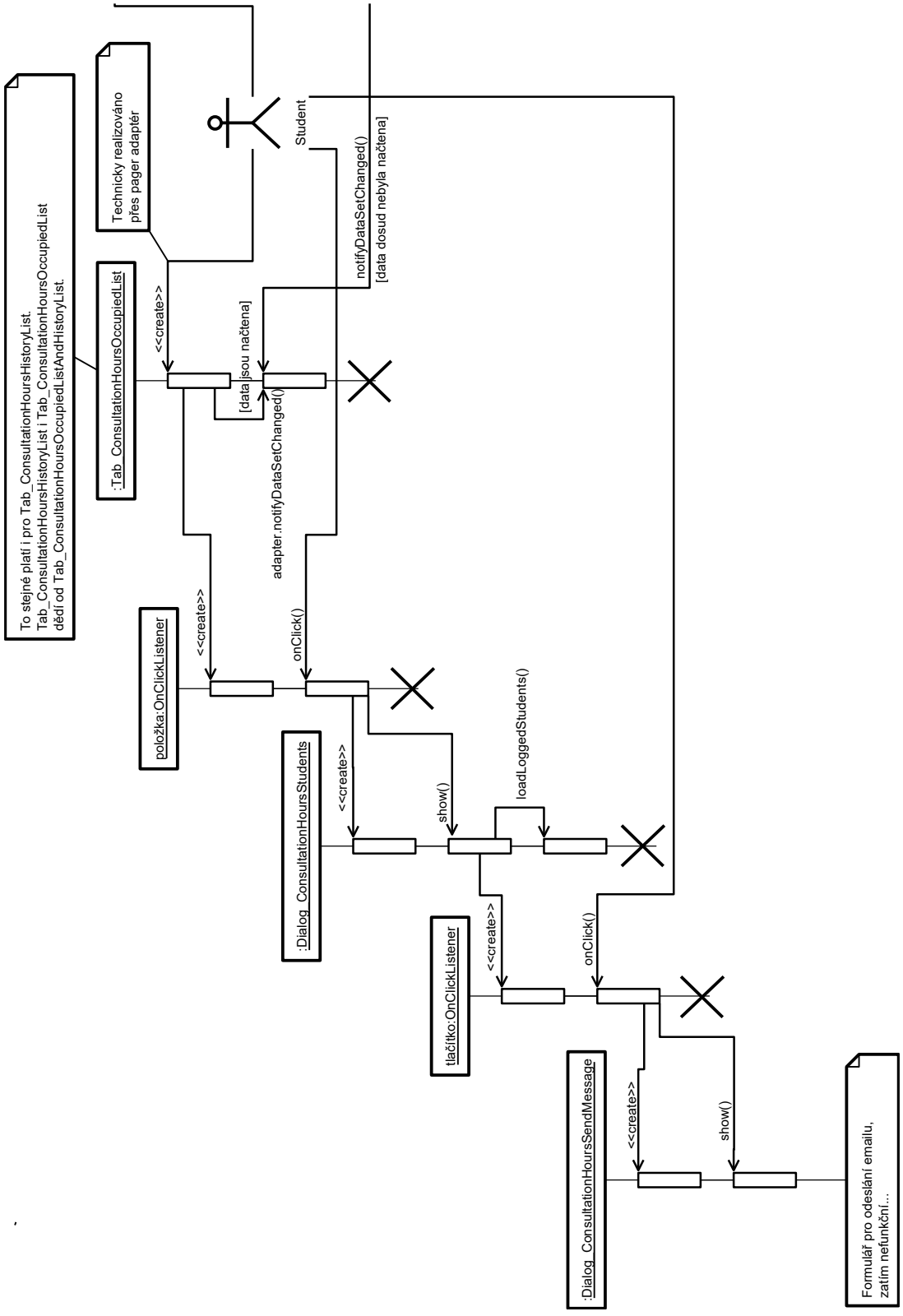
Obrázek 24: *Sekvenční diagram zachycující stažení materiálu.*



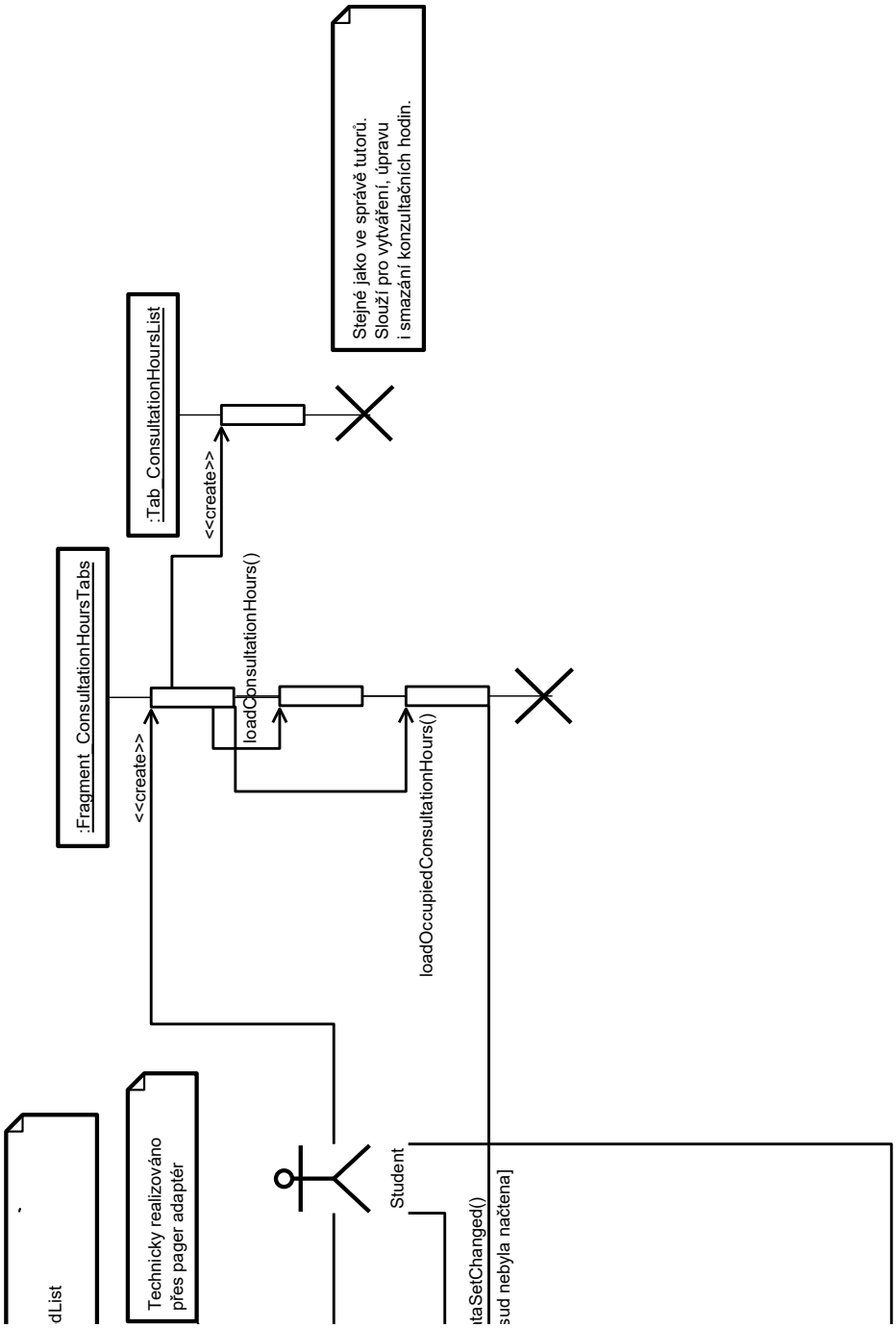
Obrázek 25: *Sekvenční diagram popisující odevzdání aktivity uživatelem v roli studenta*



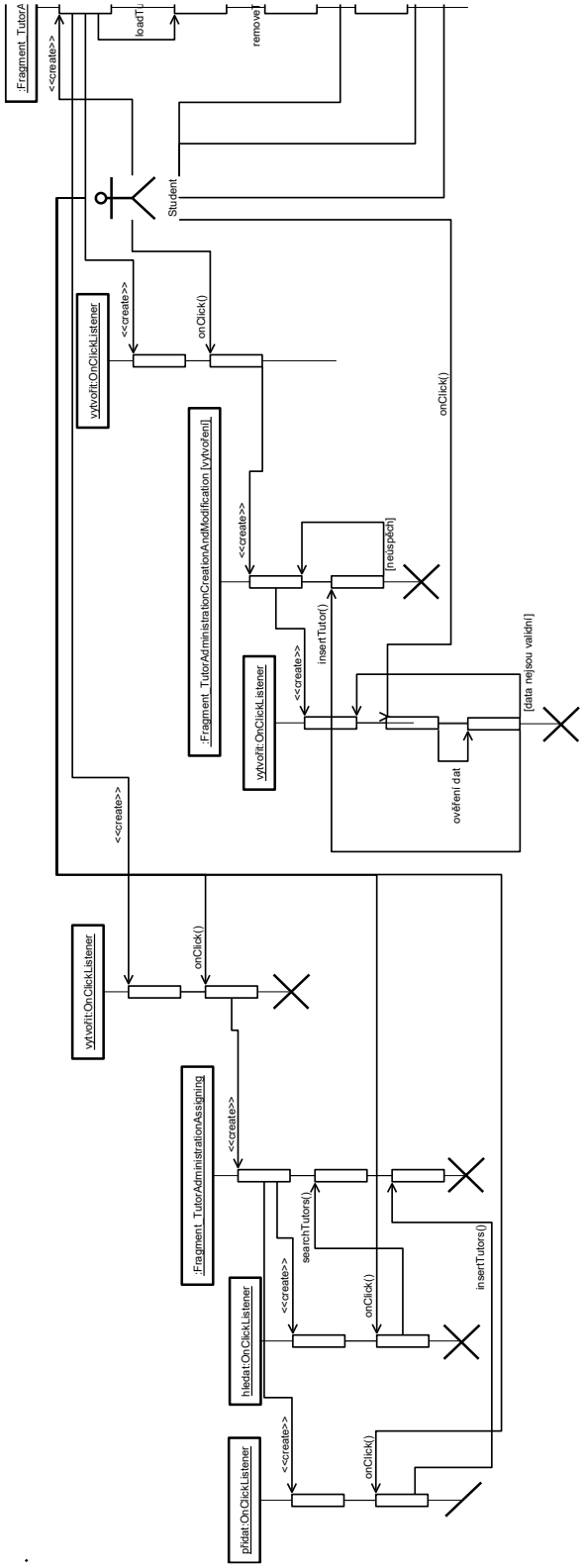
Obrázek 26: *Sekvenční diagram zachycující přihlášení do aplikace s výběrem školy a role*



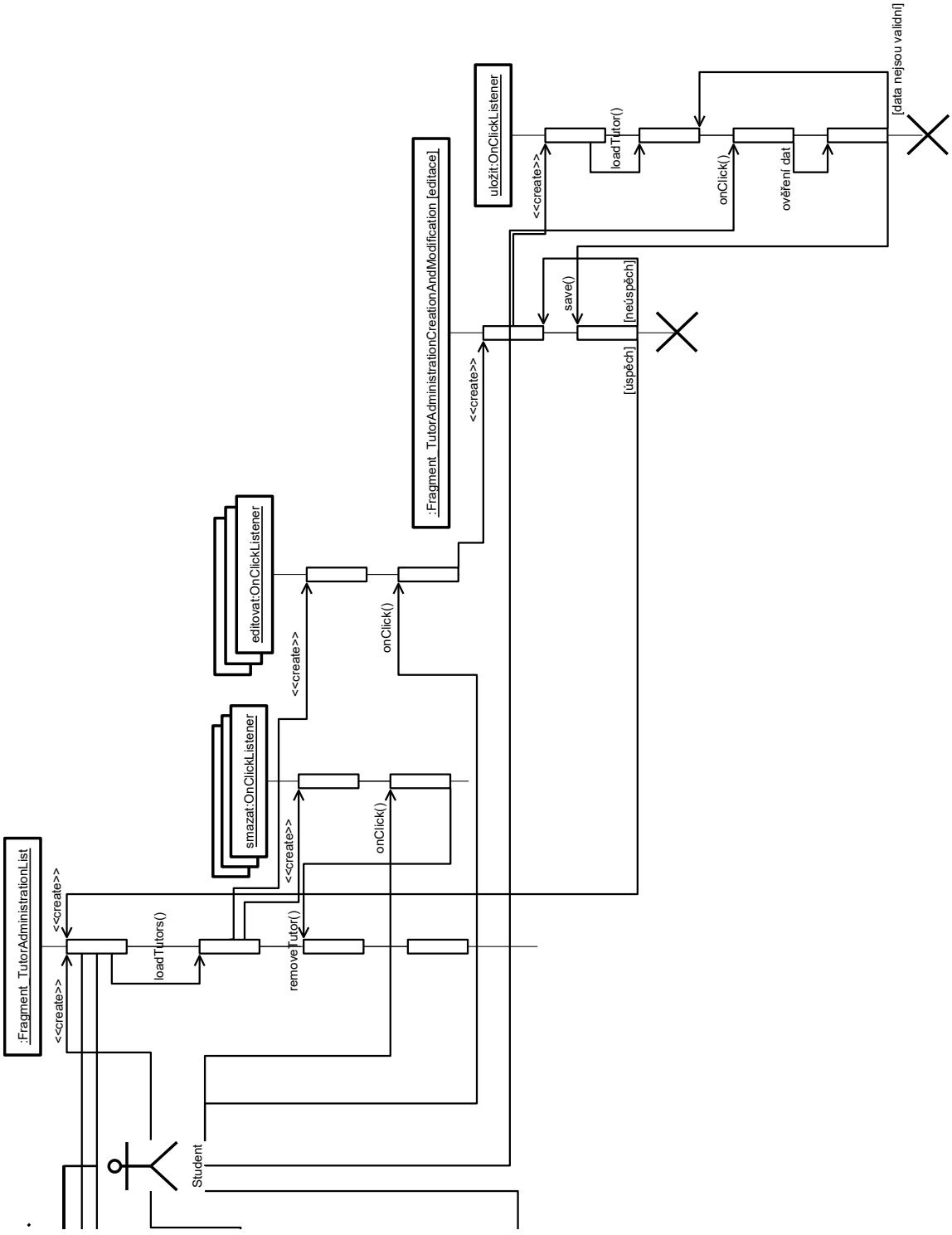
Obrázek 27: *Levá polovina sekvenčního diagramu ke správě konzultačních hodin*



Obrázek 28: *Pravá polovina sekvenčního diagramu ke správě konzultačních hodin*

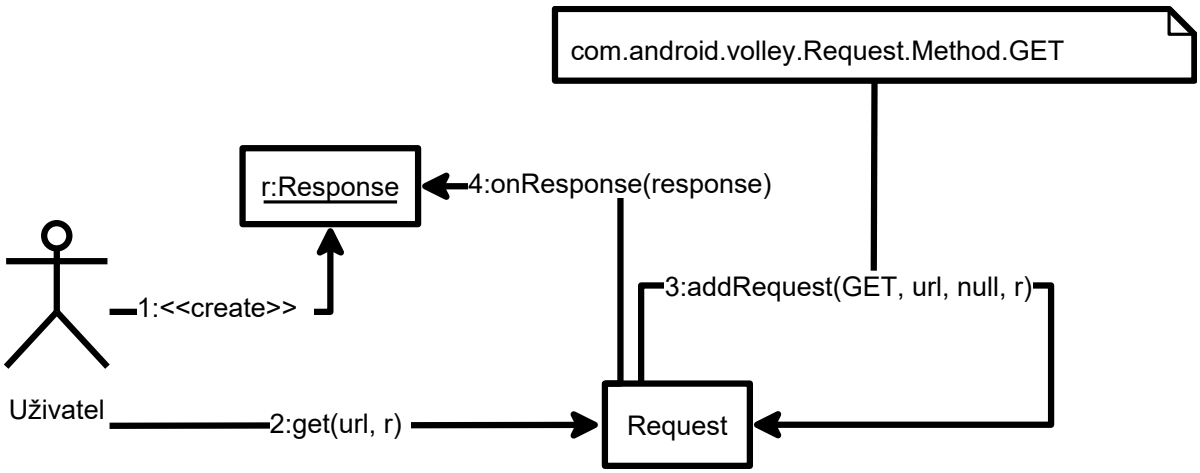


Obrázek 29: *Levá polovina sekvenčního diagramu ke správě tutorů*

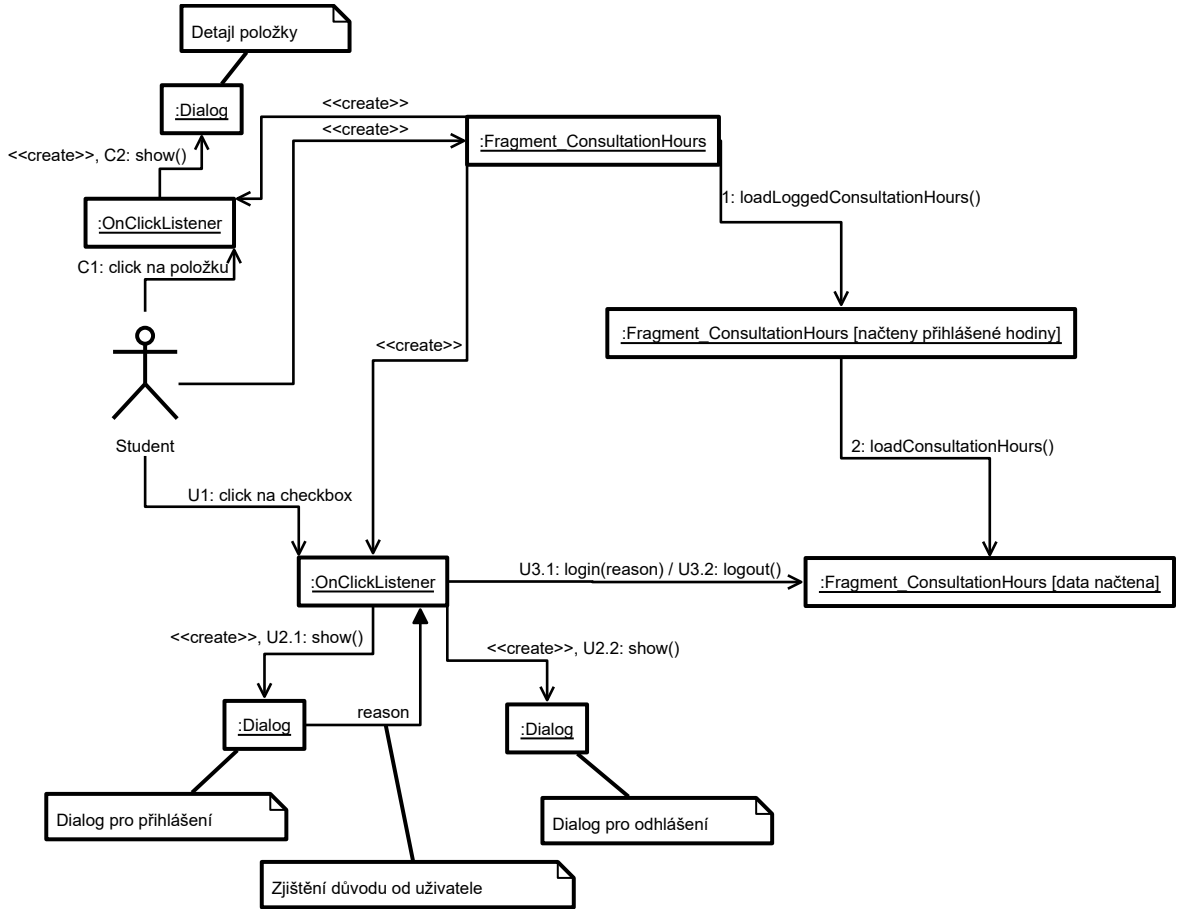


Obrázek 30: *Pravá polovina sekvenčního diagramu ke správě tutorů*

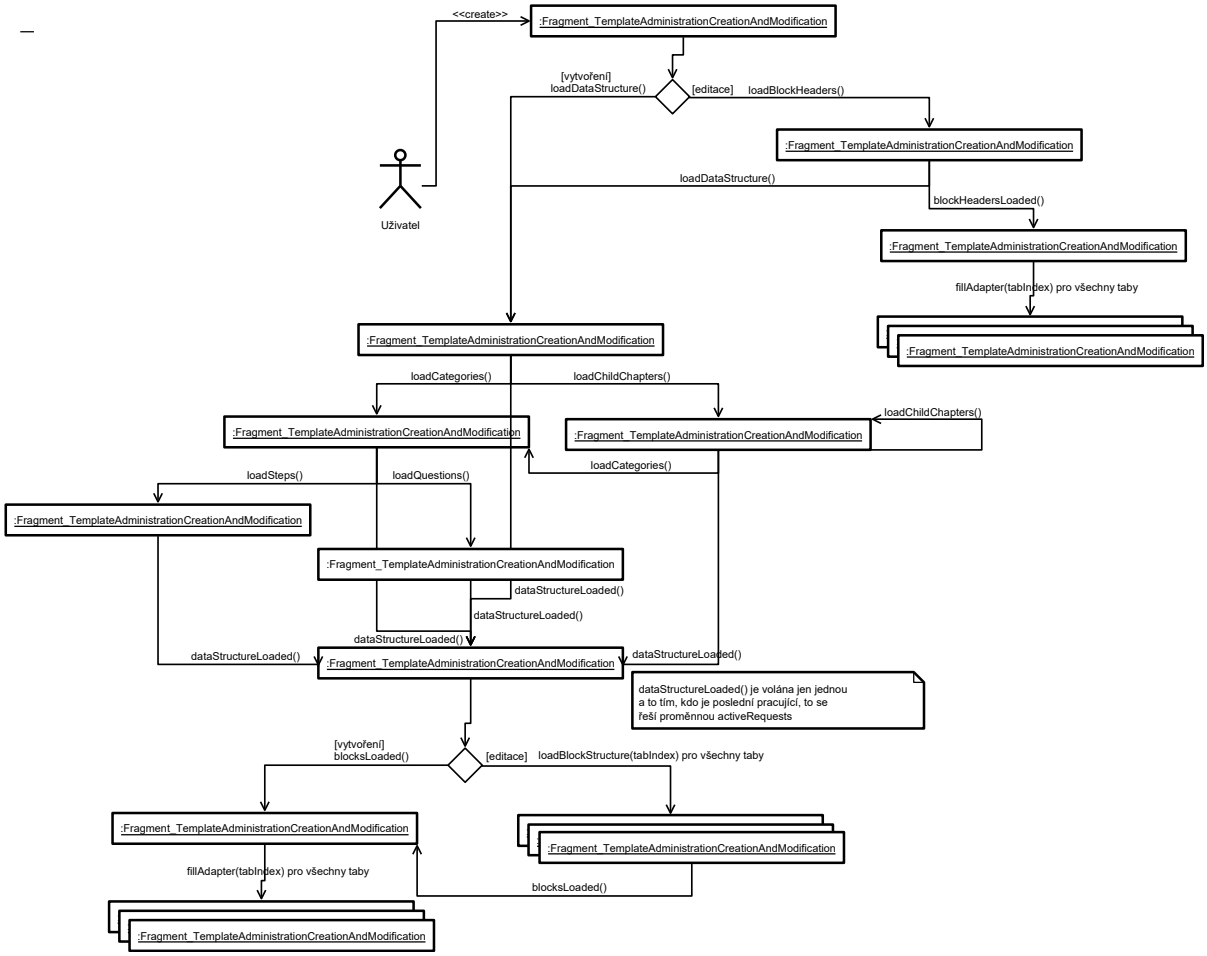
G *Diagram komunikace*



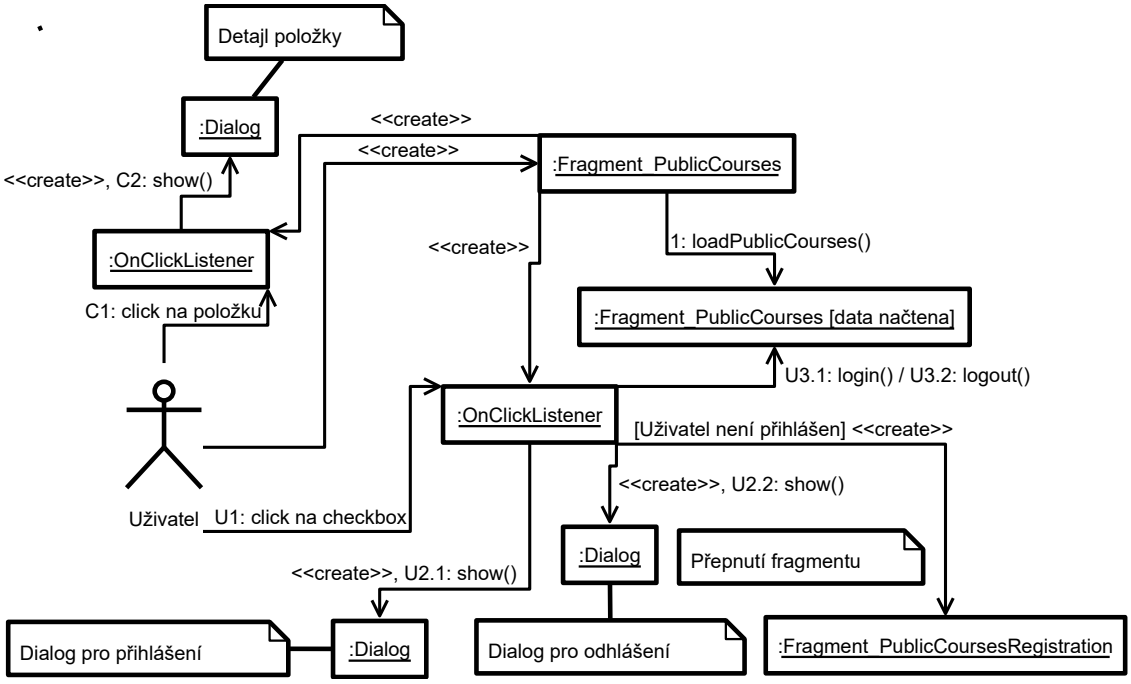
Obrázek 31: *Diagram komunikace ukazující komunikaci s knihovnou Volley*



Obrázek 32: *Diagram komunikace pro případ užití „Konzultační hodiny“ pro uživatele v roli studenta*

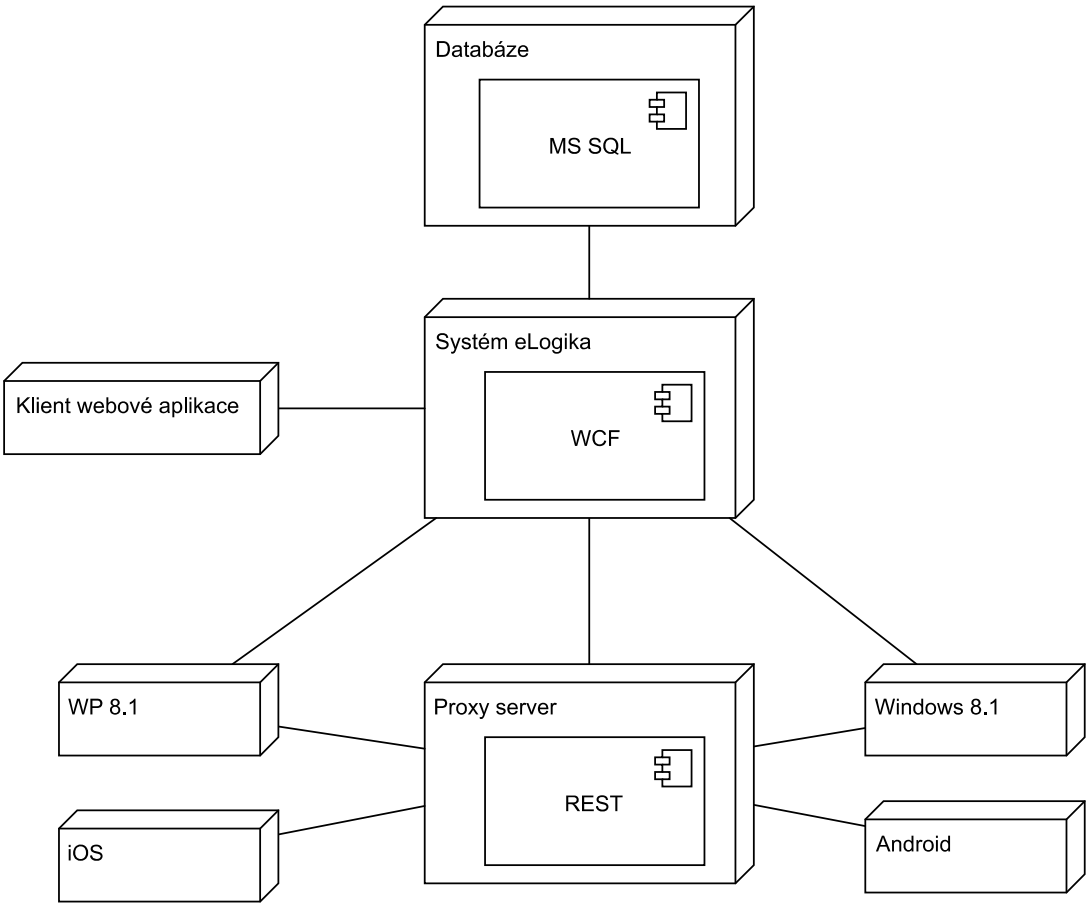


Obrázek 33: *Diagram komunikace zachycující vytváření a editaci šablon*



Obrázek 34: *Diagram komunikace pro zobrazování veřejných kurzů, přihlašování na ně a odhlašování z veřejného kurzu, na který je uživatel přihlášen*

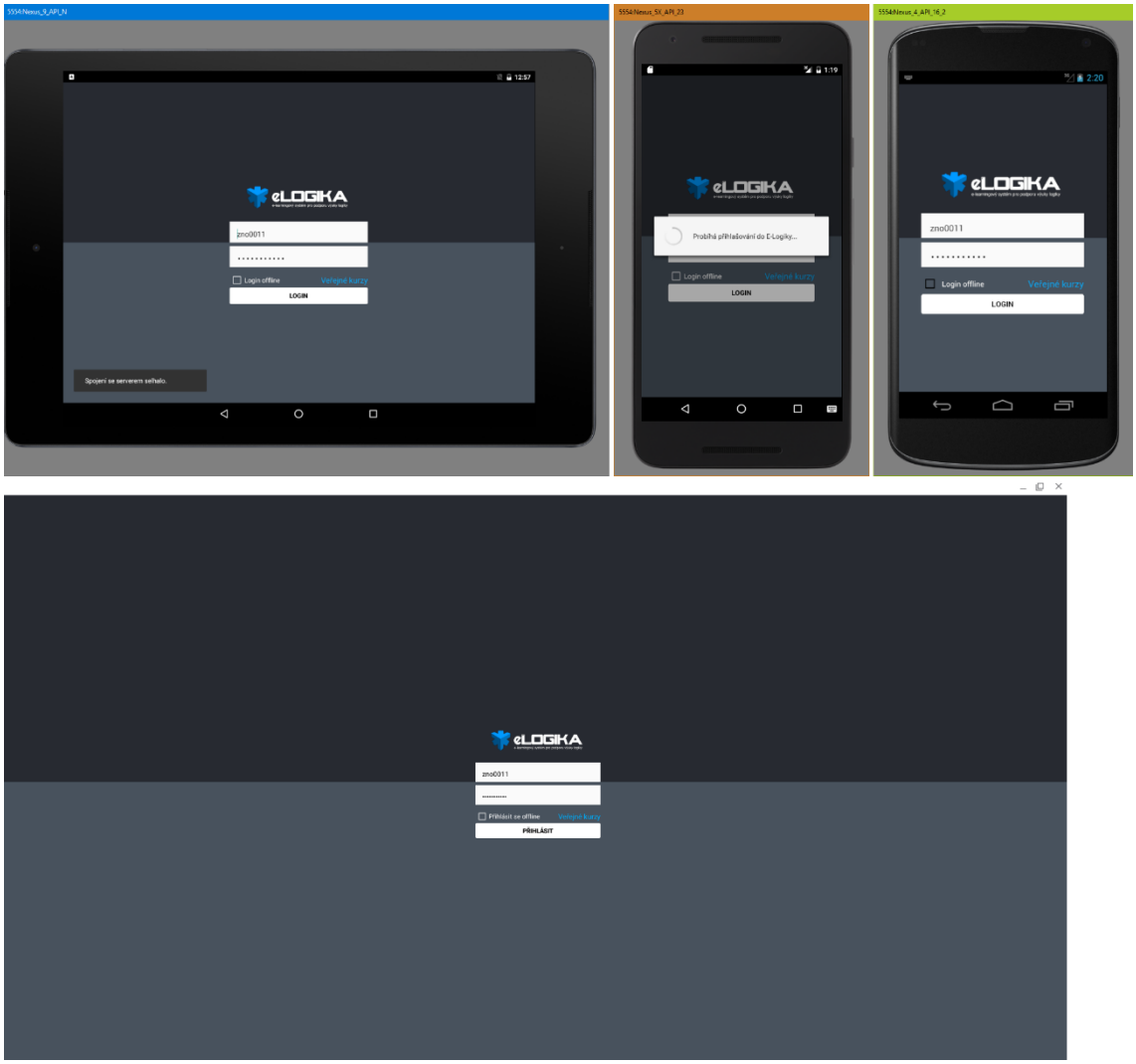
H *Diagram nasazení*



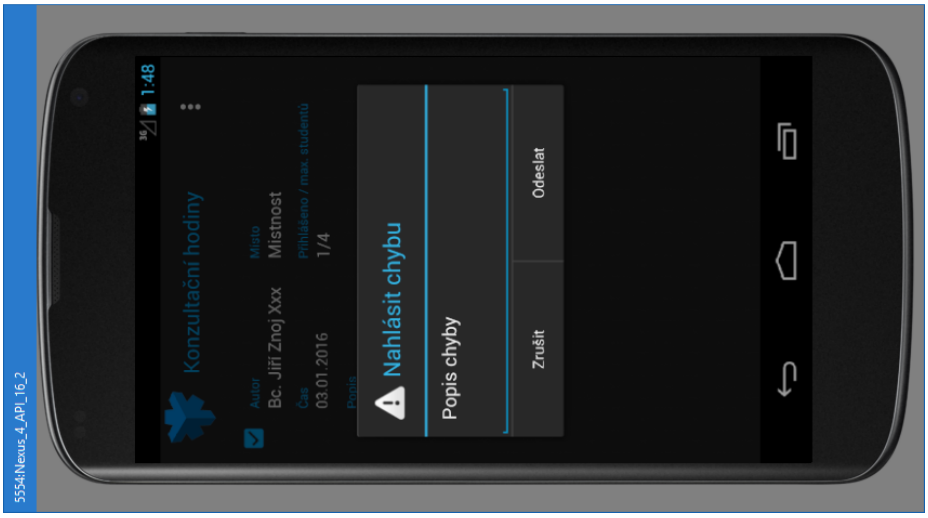
Obrázek 35: *Diagram nasazení pro systém eLogika*

I *Ukázky uživatelského rozhraní*

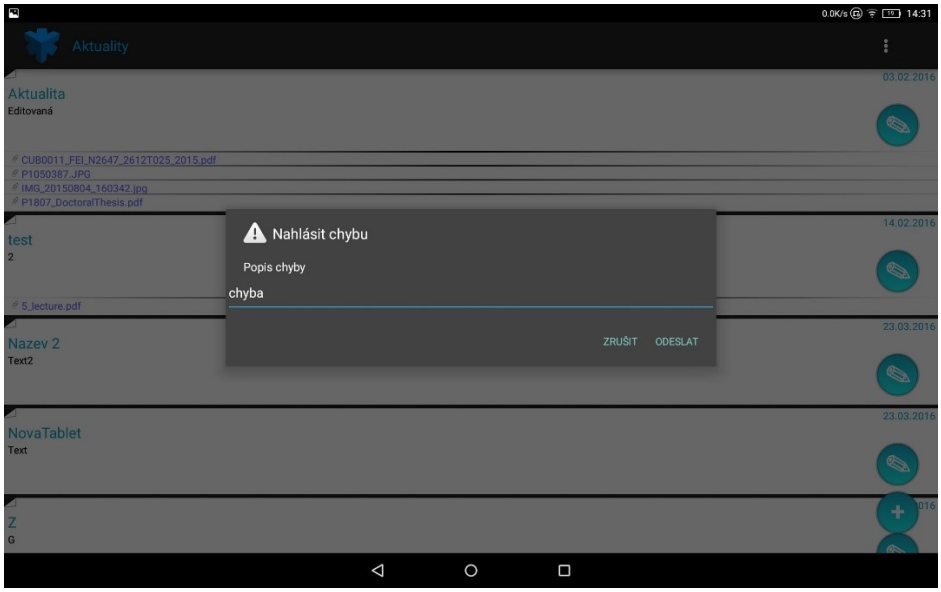
Obrázek 36: *Zleva: LG Optimus 2x (Android 4.4), Xiaomi Mi5 (Android 6.0), Lenovo Yoga 2 (Android 5.0)*



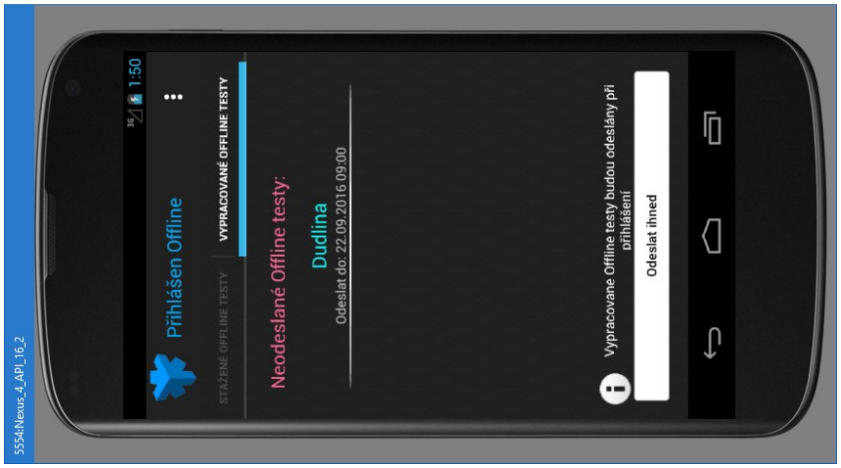
Obrázek 37: *Nahoře jsou znázorněny emulátory Android Studia (Android N, Android 6.0, Android 4.1), dole je pak ukázka okna ARC*



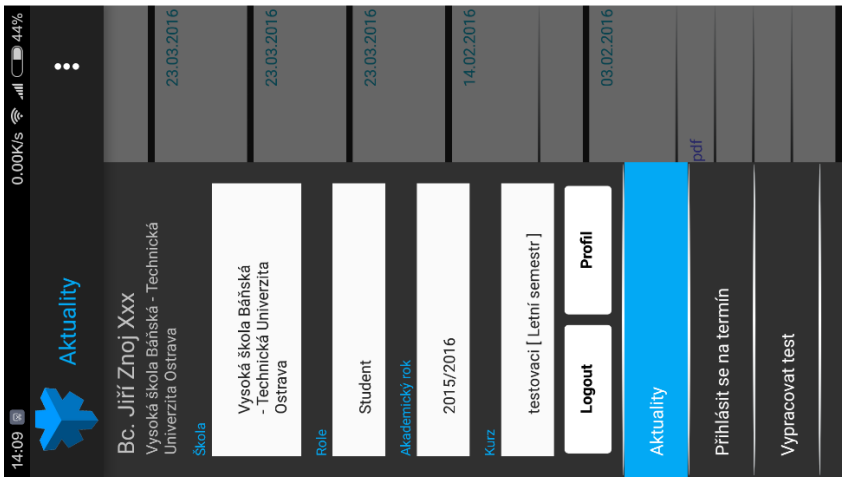
Obrázek 38: *Okno s hlášením chyby v OS Android 4.1*



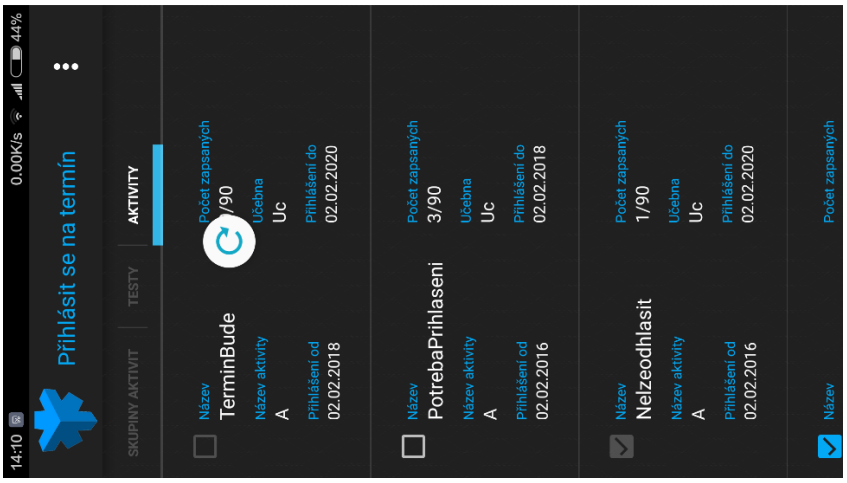
Obrázek 39: *Okno s hlášením chyby v OS Android 5.0*



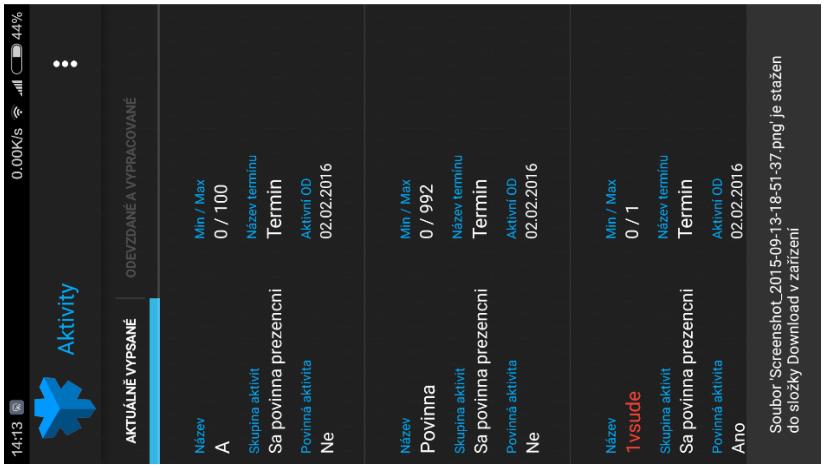
Obrázek 40: *Okno vypracovaného online testu*



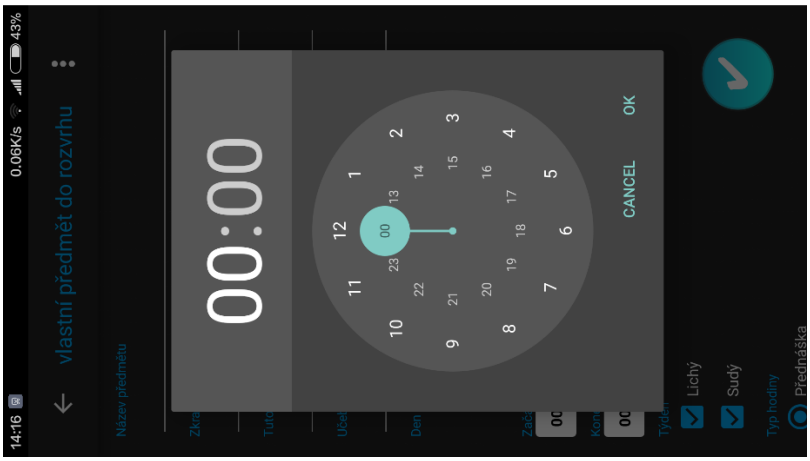
Obrázek 41: *Ukázka postranního menu aplikace*



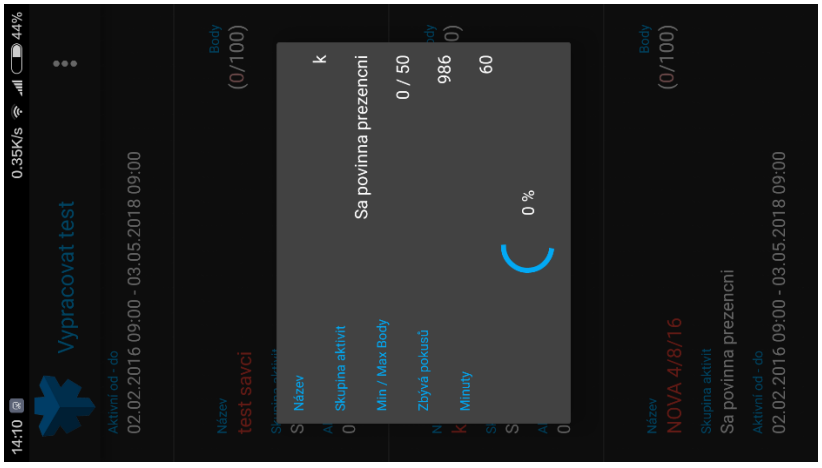
Obrázek 42: *ViewPager s ListView a ukázkou načítání SwipeRefreshLayoutu v rámci šablony list_view_swipe*



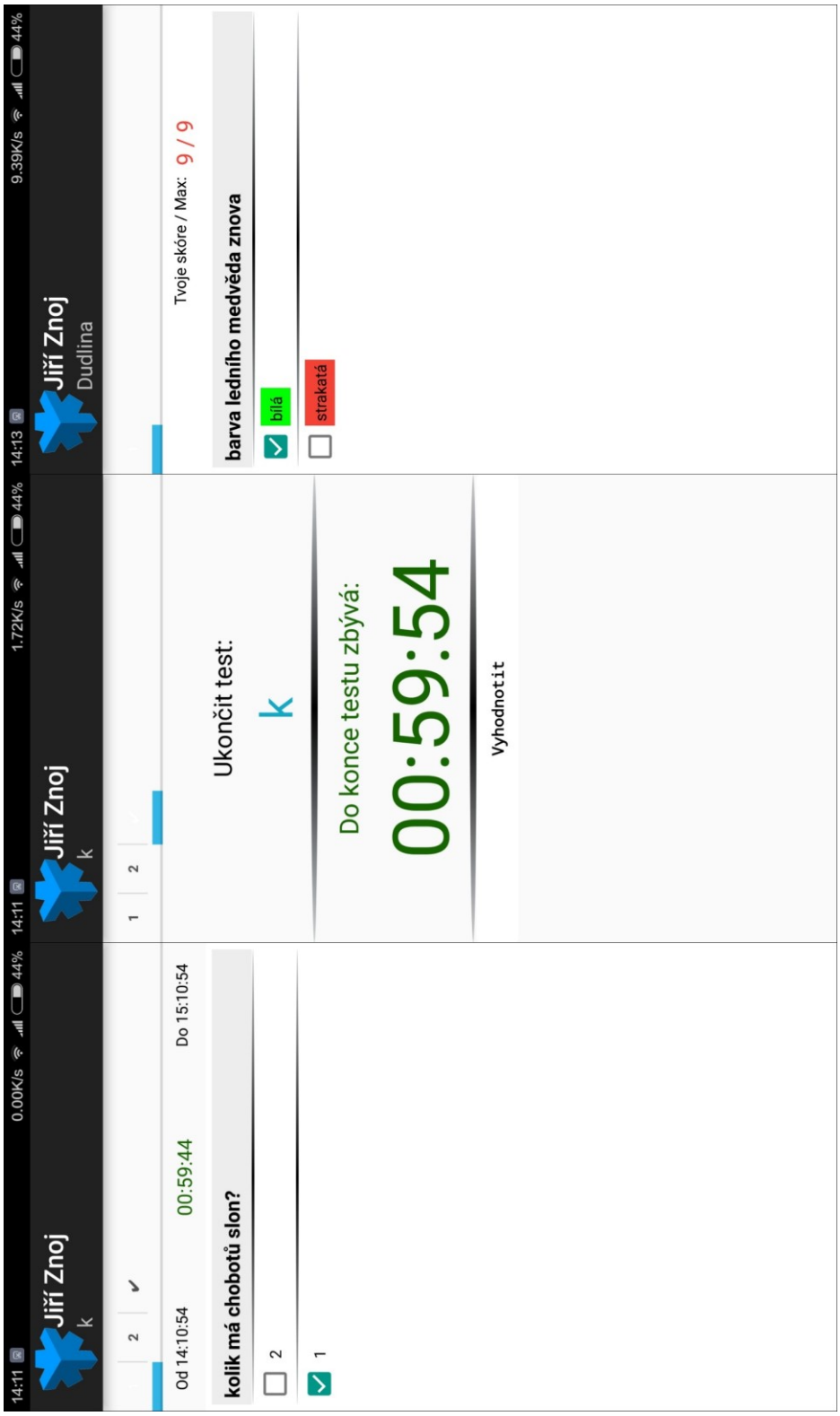
Obrázek 43: *Ukázka Snackbaru překrývajícího ListView s aktivitami*



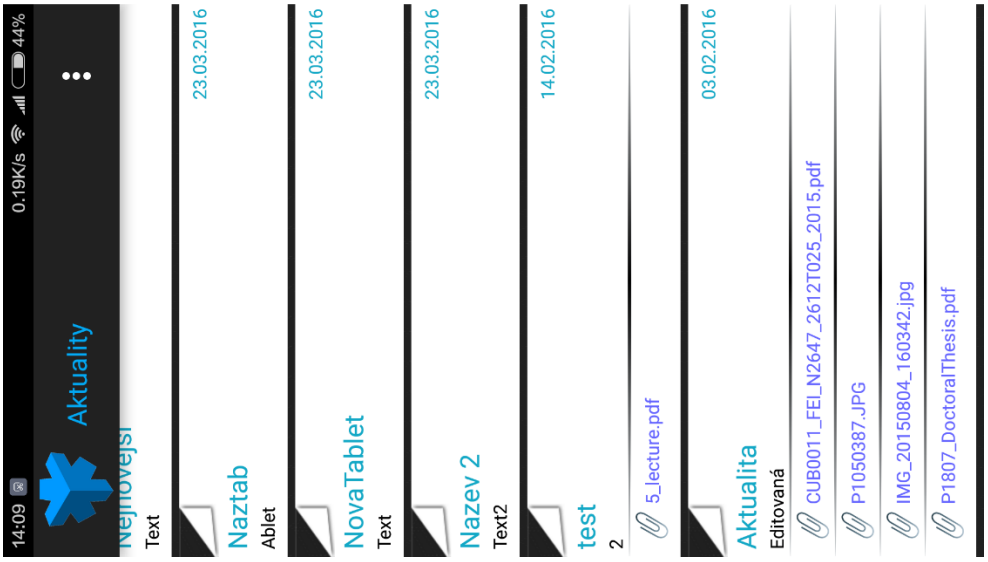
Obrázek 44: *Volba času uvnitř knihovny TimePickerDialogOnButton*



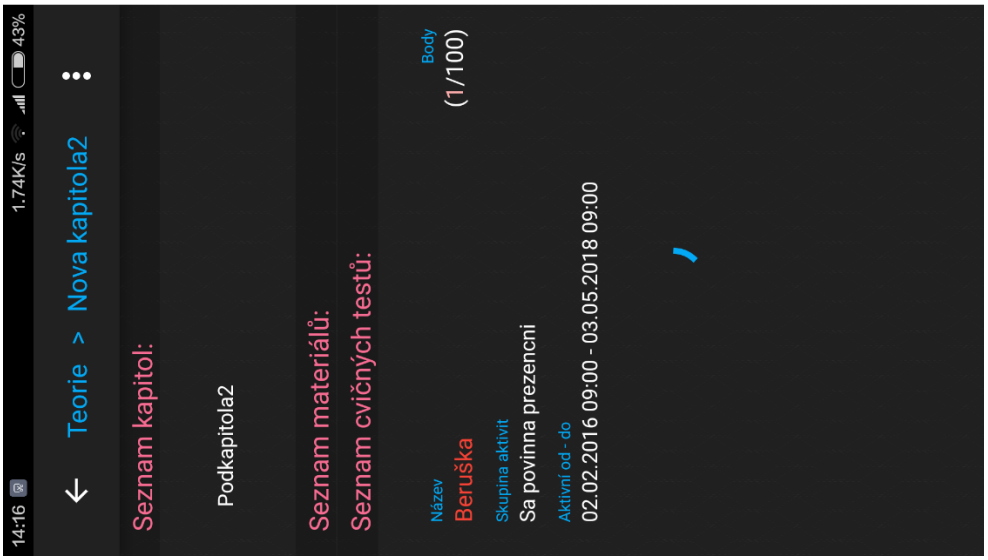
Obrázek 45: *Ukázka načítání testu a otázek*



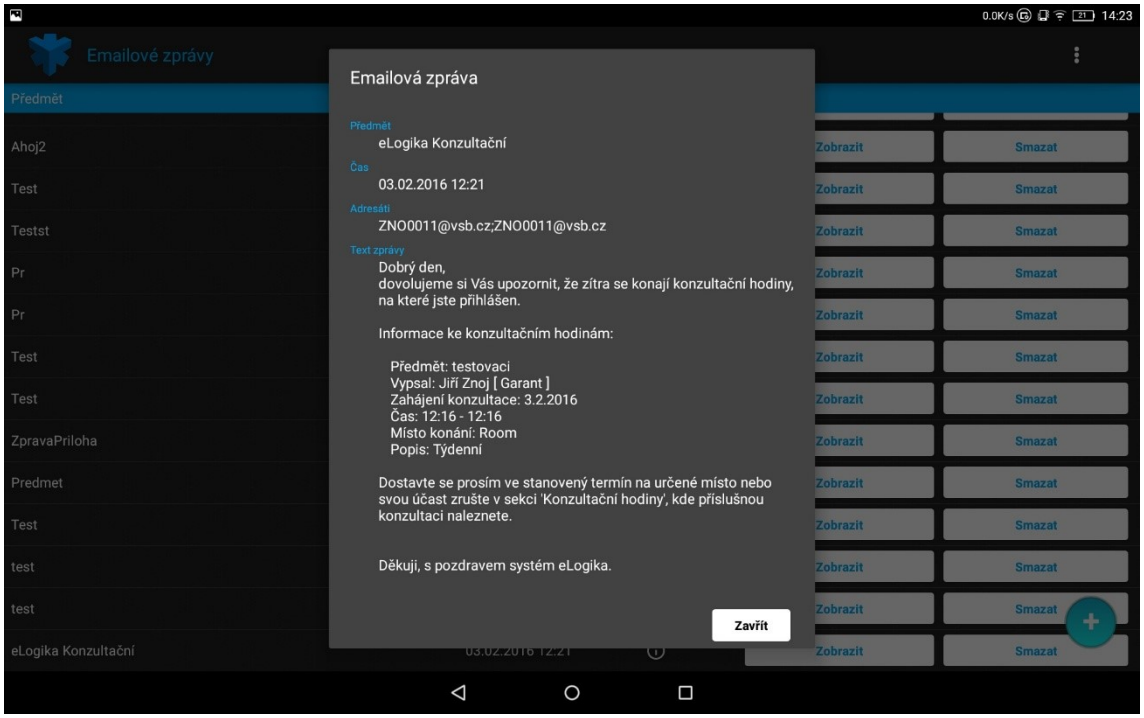
Obrázek 46: *Ukázka vypracování, vyhodnocení a zobrazení testu*



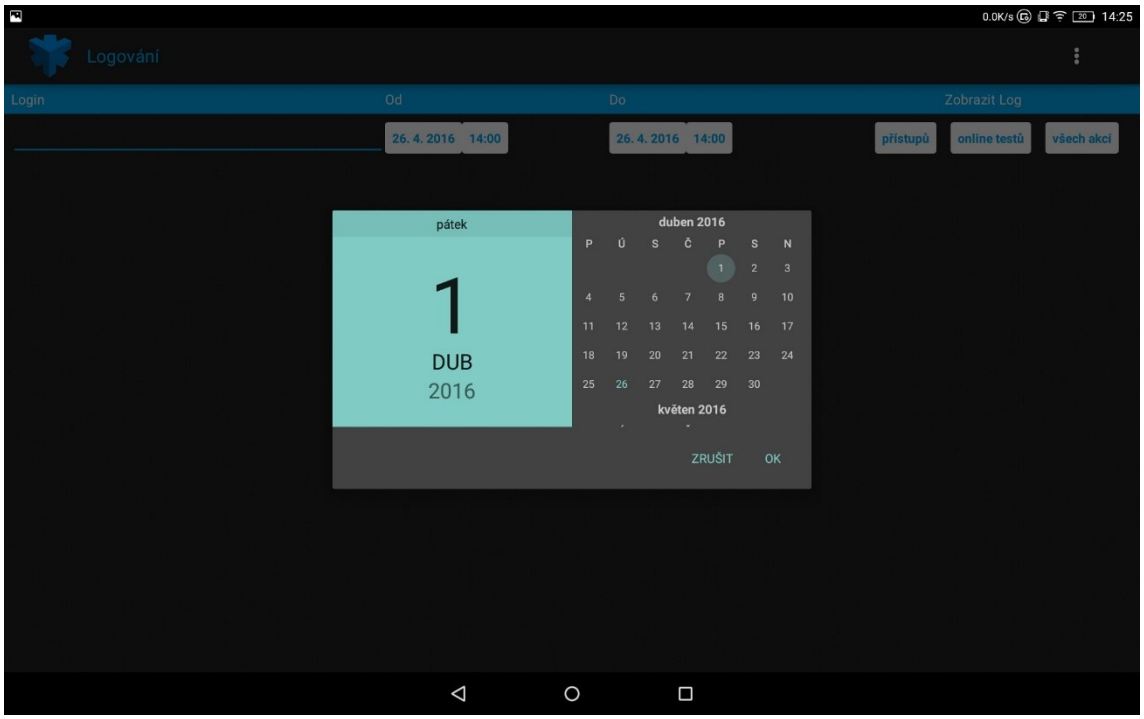
Obrázek 47: *Aktuality zobrazované uživateli v roli studenta*



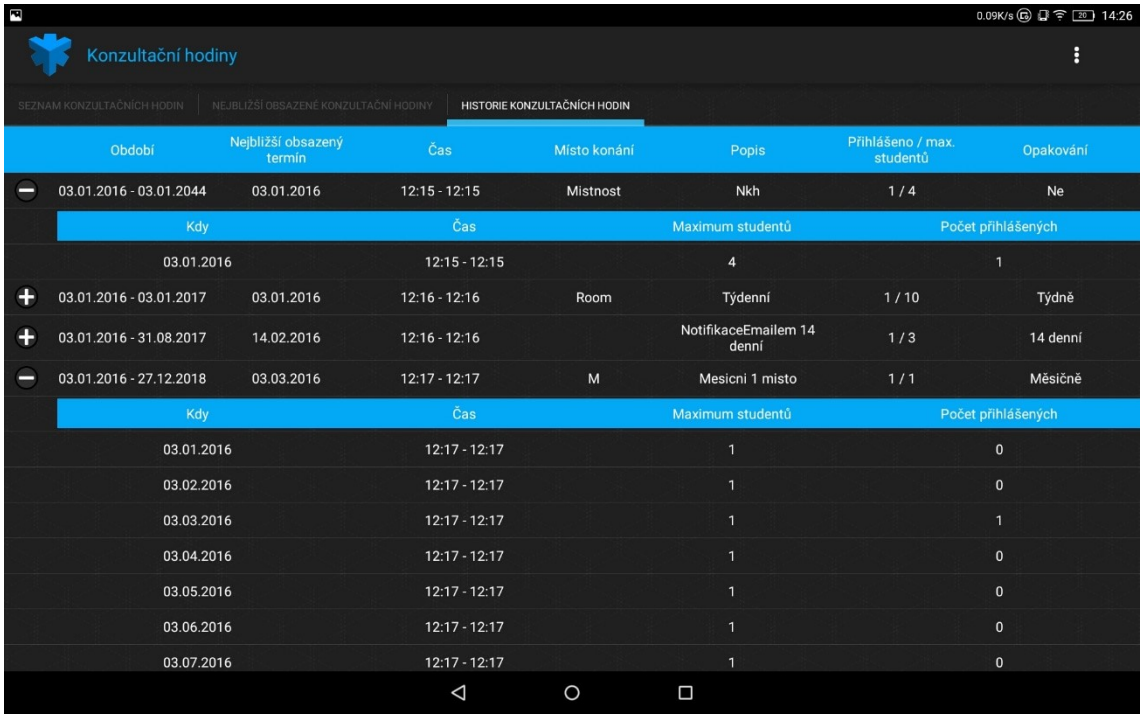
Obrázek 48: *SeparatedListAdapter*



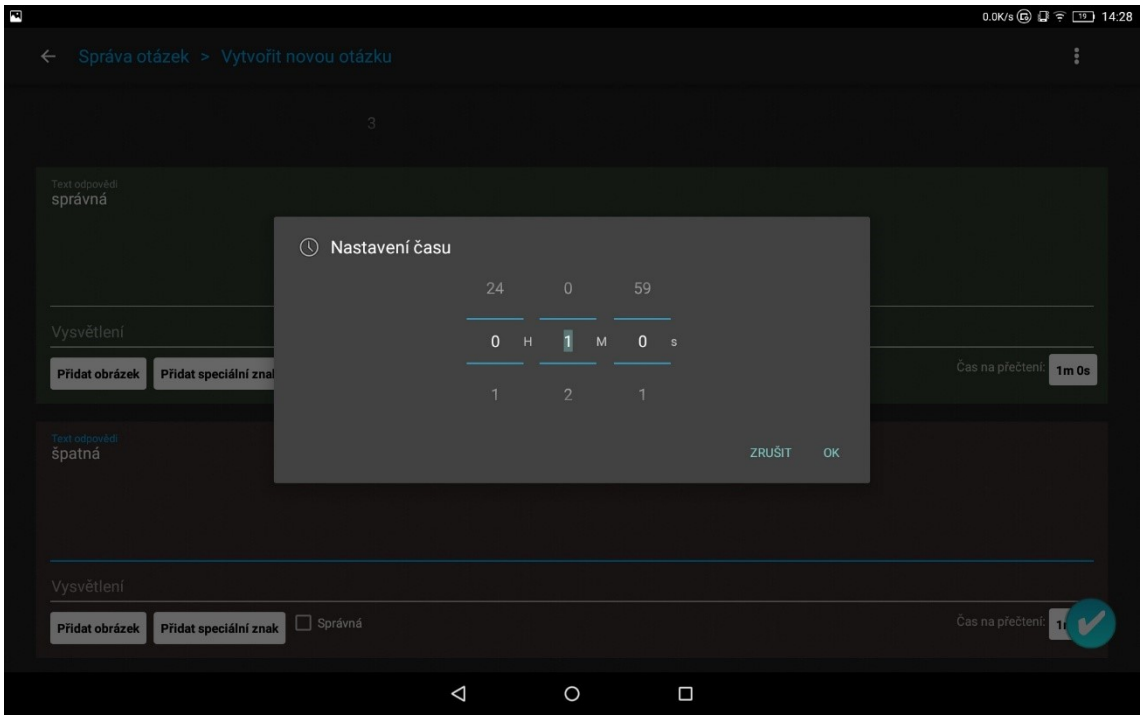
Obrázek 49: *Emailové zprávy*



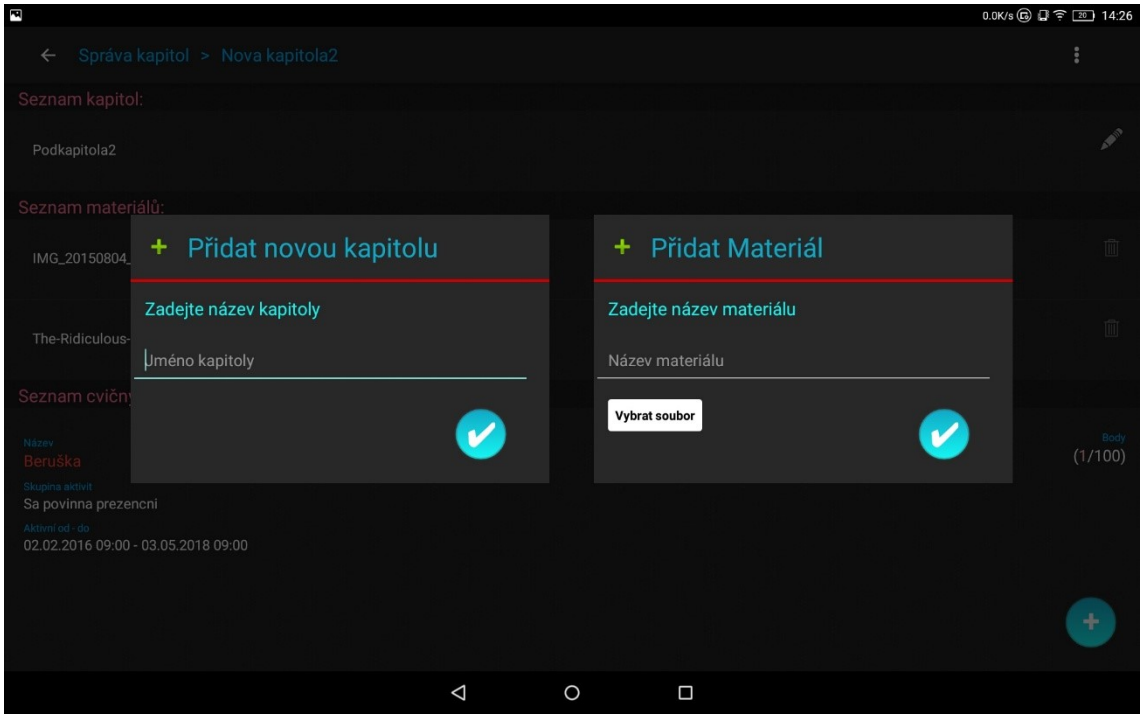
Obrázek 50: *Výběr data uvnitř knihovny DataPickerDialogOnButton*



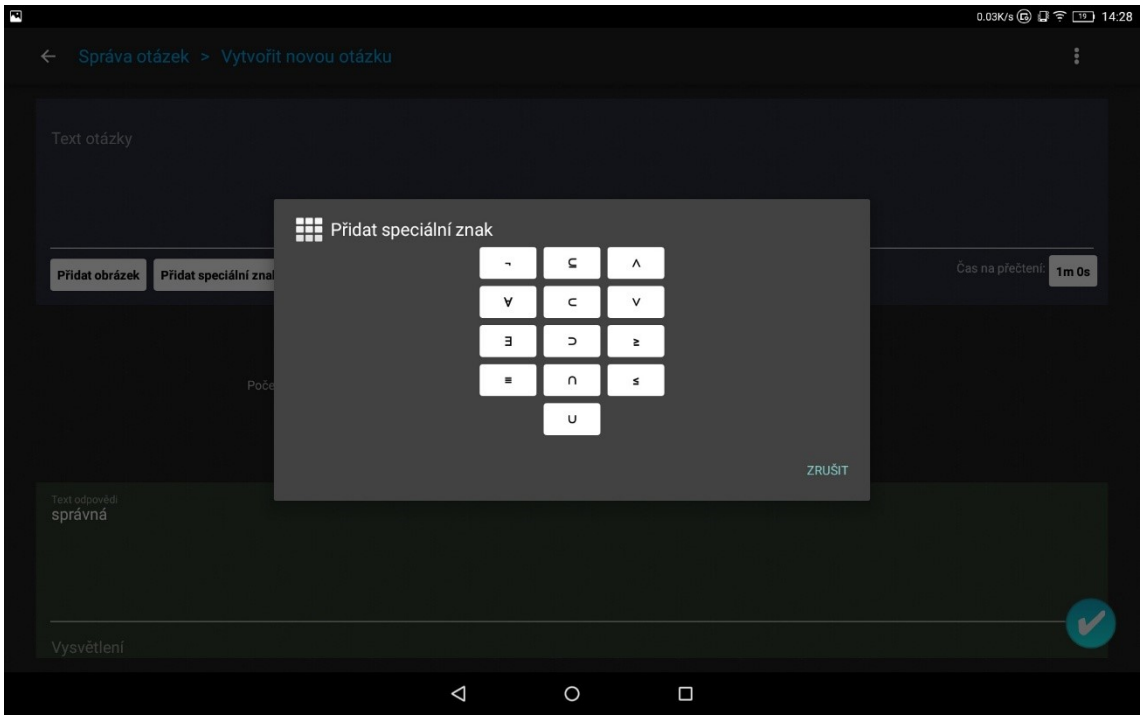
Obrázek 51: *Ukázka použití knihovny HierarchicalAdapter*



Obrázek 52: *Ukázka použití knihovny TimePickerDialogButton_HMS*

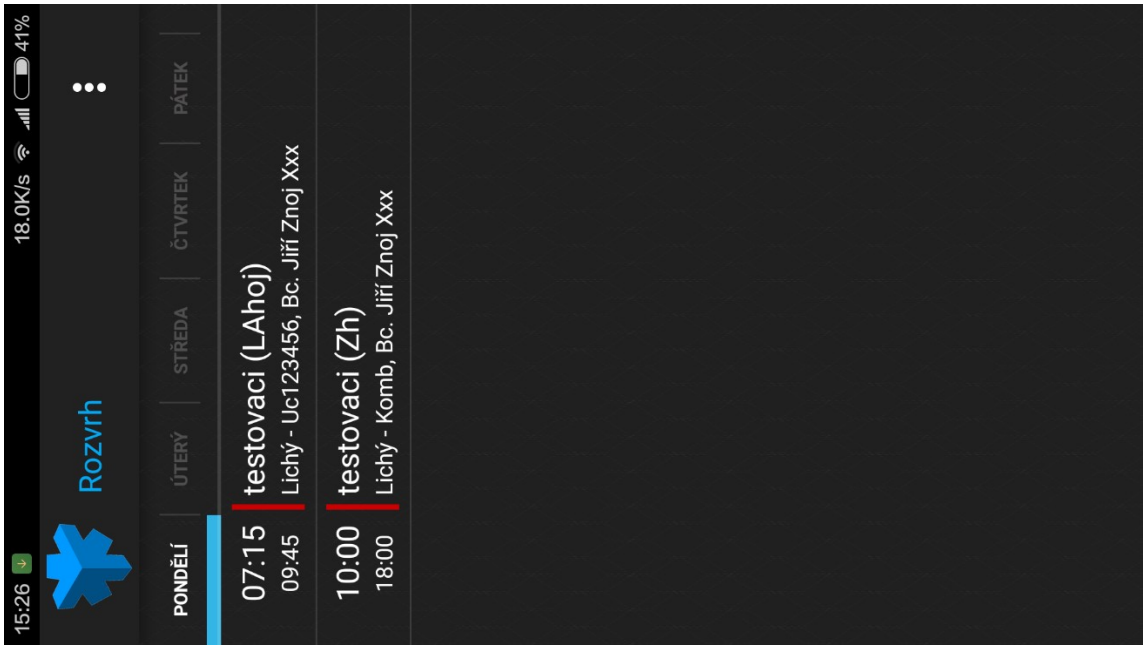


Obrázek 53: *Ukázka případu užití Správa kapitol pro uživatele v roli Garanta*

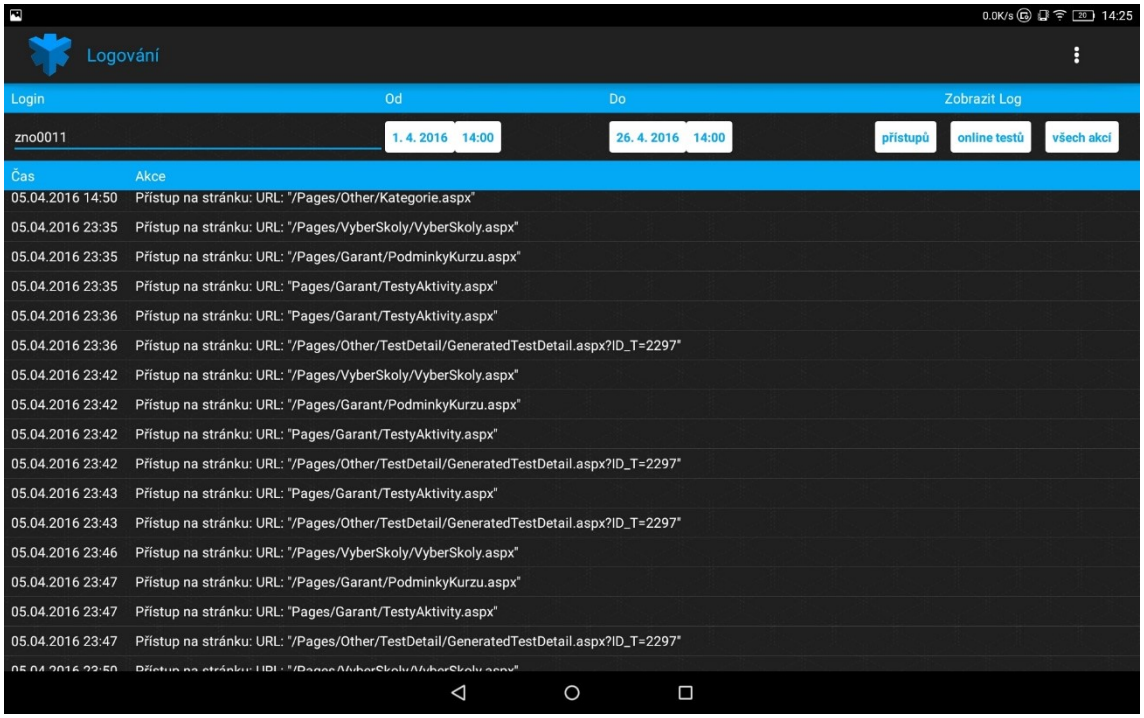


Obrázek 54: *Ukázka použití knihovny SpecialCharDialogButton*

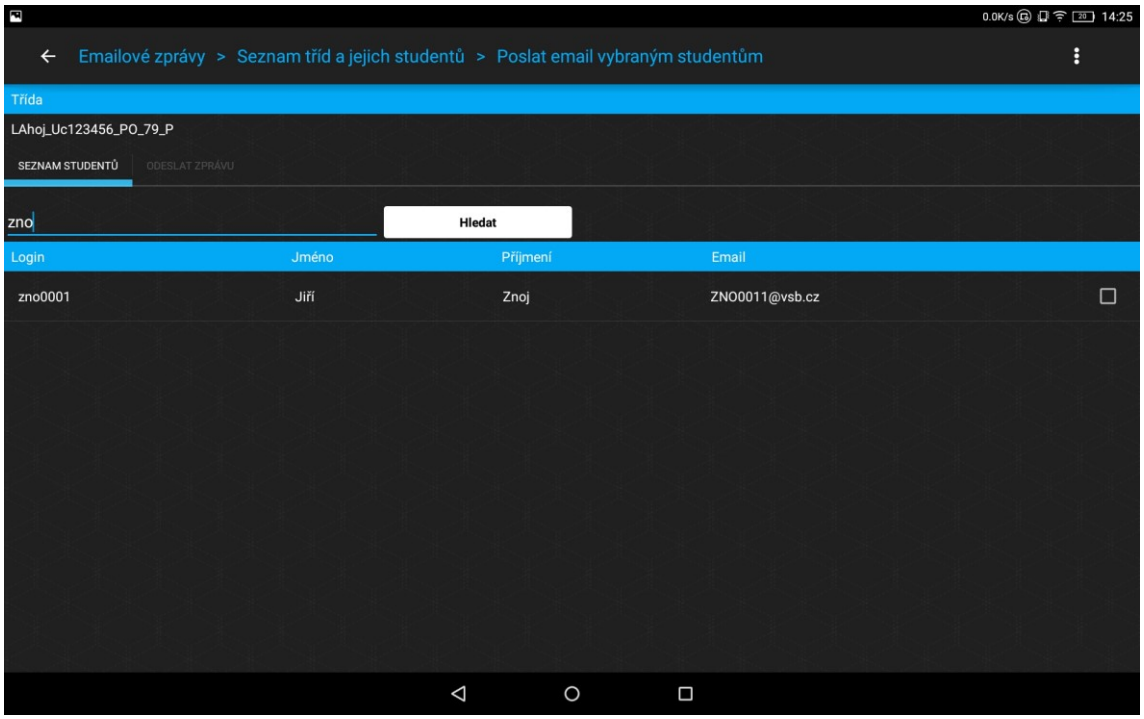
Obrázek 55: *Ukázka rovrzhu pro uživatele v roli Garanta*



Obrázek 56: *Ukázka rozvrhu pro uživatele v roli studenta*



Obrázek 57: *Ukázka použití RecyclerView*



Obrázek 58: *Ukázka ActionBaru a filtrování pomocí loginu*

J *Použité REST API*

1 User/

a) Get/{id}

GET metoda pro získání detailních informací o uživateli.

parametry GET metody:

- „id„ – id uživatele, jehož detailní informace chceme získat

b) loginsecureobject

POST metoda pro přihlášení uživatele

parametry POST metody:

- „Login„ – login uživatele
- „Password„ – haš hesla z metody J 1c)

c) getsaltbylogin?login={login}

GET metoda pro získání soli pro hašování hesla pro parametr metody J 1b)

parametry GET metody:

- „login„ – login uživatele, pro kterého chceme sůl získat

d) login?login={login}&password={password}

GET metoda pro přihlášení uživatele

- „login„ – login uživatele
- „password„ – haš hesla z metody J 1c)

e) registrationexistuser?userId={userId}&courseInfoId={courseInfoId}

POST metoda pro přihlášení existujícího uživatele na vybraný veřejný kurz.

parametry adresy POST metody:

- „userId“ – id uživatele
- „courseInfoId“ – id kurzu v daném roce na který se chce uživatel přihlásit

f) unregistrationexistuser?userId={userId}&courseInfoId={courseInfoId}

POST metoda pro odhlášení existujícího uživatele z vybraného veřejného kurzu.

parametry adresy POST metody:

- „userId“ – id uživatele
- „courseInfoId“ – id kurzu v daném roce na který se chce uživatel přihlásit

g) registrationuser?courseInfoId={courseInfoId}&loginSkola={loginSkola}

POST metoda pro vytvoření nového uživatele, vytvoření jeho loginu a následné přihlášení k vybranému veřejnému kurzu.

parametry adresy POST metody:

- „courseInfoId“ – id kurzu v daném roce na který se chce uživatel přihlásit
- „loginSkola“ – školní login uživatele

parametry POST metody:

- „Jmeno“ – jméno uživatele
- „Prijmeni“ – příjmení uživatele
- „PrijmeniRodne“ – rodné příjmení uživatele
- „Email“ – email uživatele
- „Jazyk“ – jazyk uživatele
- „Ulice“ – ulice uživatele
- „CisloPop“ - číslo popisné uživatele
- „PSC“ – PSČ uživatele
- „Mesto“ – město uživatele
- „TitulPred“ – titul uživatele uvedený před jménem
- „TitulZa“ – titul uživatele uvedený za jménem
- „Fotka“ – fotka uživatele v kódování Base64
- „Salt“ – náhodný řetězec v kódování Base64
- „Heslo“ – MD5 haš hesla v kódování Base64

h) Post

POST metoda pro úpravu dat existujícího uživatele.

parametry POST metody:

- „Login“ – login uživatele
- „Jmeno“ – jméno uživatele
- „Prijmeni“ – příjmení uživatele
- „PrijmeniRodne“ – rodné příjmení uživatele
- „Email“ – email uživatele
- „Jazyk“ – jazyk uživatele
- „Ulice“ – ulice uživatele
- „CisloPop“ - číslo popisné uživatele
- „PSC“ – PSČ uživatele
- „Mesto“ – město uživatele
- „TitulPred“ – titul uživatele uvedený před jménem

- „TitulZa“ – titul uživatele uvedený za jménem

i) generatelogin?p_prijmeni={p_prijmeni}

GET metoda pro vygenerování systémového loginu.

parametry GET metody:

- „p_prijmeni“ – příjmení uživatele, pro kterého chceme login nechat vygenerovat

j) updatepassword

POST metoda pro změnu hesla.

parametry POST metody:

- „IDUser“ – identifikátor uživatele
- „PasswordOld“ – MD5 haš původního hesla v kódování Base64
- „PasswordNew“ – MD5 haš nového hesla v kódování Base64

k) gettutorsbyidkurz?courseInfoId={courseInfoId}&userId={userId}

GET metoda pro získání seznamu tutorů kurzu.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id uživatele, který chce seznam tutorů získat

l) insertuser

POST metoda pro vytvoření nového účtu tutora (včetně vygenerování jeho loginu).

parametry POST metody:

- „Login“ – login tutora, kterého chceme vytvořit
- „Jmeno“ – jméno tutora
- „Prijmeni“ – příjmení tutora
- „PrijmeniRodne“ – rodné příjmení tutora
- „Email“ – email tutora
- „Jazyk“ – jazyk tutora
- „Ulice“ – ulice tutora
- „CisloPop“ - číslo popisné tutora
- „PSC“ – PSČ tutora
- „Mesto“ – město tutora
- „TitulPred“ – titul tutora uvedený před jménem
- „TitulZa“ – titul tutora uvedený za jménem
- „Fotka“ – fotka tutora v kódování Base64
- „Salt“ – náhodný řetězec v kódování Base64

- „Heslo“ – MD5 haš hesla v kódování Base64
- „Smazany“ – příznak o existenci tutora

m) RemoveTutor?tutorId={tutorId}&courseInfoId={courseInfoId}&schoolInfoId={schoolInfoId}

GET metoda pro odstranění vybraného tutora z konkrétního kurzu.

parametry GET metody:

- „tutorId“ – id tutora, kterého chceme z kurzu odebrat
- „courseInfoId“ – id kurzu ve vybraném roce
- „schoolInfoId“ – id školy ve vybraném roce

n) inserttutor?tutorId={tutorId}&schoolInfoId={schoolInfoId}&courseInfoId={courseInfoId}&schoolInfoId={schoolInfoId}

POST metoda pro přiřazení tutora ke konkrétnímu kurzu ve vybrané škole.

parametry adresy POST metody:

- „tutorId“ – id tutora, kterého chceme k vybranému kurzu přiřadit
- „courseInfoId“ – id kurzu ve vybraném roce
- „schoolInfoId“ – id školy ve vybraném roce
- „schoolLogin“ – školní login tutora, kterého chceme k vybranému kurzu přiřadit

o) inserttutors?schoolInfoId={schoolInfoId}&courseInfoId={courseInfoId}&schoolLogin={schoolLogin}

POST metoda pro přiřazení tutorů ke konkrétnímu kurzu ve vybrané škole.

parametry adresy POST metody:

- „courseInfoId“ – id kurzu ve vybraném roce
- „schoolInfoId“ – id školy ve vybraném roce
- „schoolLogin“ – „defaultní školní login“, pokud některý z tutorů nemá žádný login zadán

parametry POST metody:

- „[tutorId, tutorId, …]“ – pole id tutorů oddělených čárkami

p) FindUserBySchool?text={text}&schoolId={schoolId}&courseInfoId={courseInfoId}

GET metoda pro získání tutorů v konkrétní škole a kurzu, kteří vyhovují zadanému příjmení.

parametry GET metody:

- „text“ – příjmení tutorů, které chceme ze serveru získat
- „schoolId“ – id školy
- „courseInfoId“ – id kurzu ve vybraném roce

q) SendBugReport

POST metoda pro možnost nahlásit chybu aplikace.

parametry POST metody:

- „OSName“ – jméno operačního systému, ze kterého je chyba hlášena
- „OSVersion“ – verze operačního systému, ze kterého je chyba hlášena
- „Device“ – zařízení, ze kterého je chyba hlášena
- „IdUser“ – id uživatele, který hlásí chybu v aplikaci
- „IdCourse“ – id kurzu v daném roce, na který je uživatel přihlášen
- „IdRole“ – id role uživatele, který hlásí chybu v aplikaci
- „IdInfoSkola“ – id školy ve vybraném roce, na který je uživatel přihlášen
- „BugText“ – popis chybu, kterou uživatel hlásí

r) LogAccess

POST metoda pro odesílání informací o používání aplikace (logů).

- „UserId“ – id uživatele, který aplikaci používá
- „RoleId“ – id role uživatele, který aplikaci používá
- „SkolaInfoId“ – id školy ve vybraném roce, na který je uživatel přihlášen
- „KurzInfoId“ – id kurzu v daném roce, na který je uživatel přihlášen
- „Cas“ – aktuální čas
- „URL“ – logovací informace ve tvaru webové aplikace – url adresa webové aplikace na které se nachází akce logovaná v aplikaci
- „IP“ – IP adresa uživatele aplikace
- „Flag“ – dodatečné informace upřesňující log z aplikace (oproti webové aplikaci)
- „OS“ – operační systém, ze kterého je log odeslán

s) GetStudentOfTrida?classId={classId}

GET metoda pro získání studentů vybrané třídy.

parametry GET metody:

- „classId“ – id třídy, ze které chceme studenty získat

t) RemoveStudent?studentId={studentId}&classId={classId}&schoolId={schoolId}

GET metoda pro odstranění vybraného studenta z vybrané třídy daného kurzu.

parametry GET metody:

- „studentId“ – id studenta, kterého chceme odebrat
- „classId“ – id třídy, ze které chceme studenta odebrat
- „schoolId“ – id školy ve kterém se třída a vybraný student nachází

u) InsertStudent?studentId={studentId}&schoolInfoId={schoolInfoId}&classId={classId}&schoolLogin={schoolLogin}

POST metoda pro přiřazení konkrétního studenta do vybrané třídy.

parametry adresy POST metody:

- „studentId“ – id studenta, kterého chceme do třídy přidat
- „schoolInfoId“ – id školy v daném roce, ve kterém se vybraný kurz nachází
- „classId“ – id třídy, do které chceme studenta přidat
- „schoolLogin“ – školní login vybraného studenta

v) getlogaccessbyiduzivatel?idUzivatel={idUzivatel}&casOD={casOD}&casDO={casDO}

GET metoda pro získání seznamu záznamů přístupů do systému.

parametry GET metody:

- „idUzivatel“ – id uživatele, jehož logy chceme zobrazit
- „casOD“ – datum a čas nejstaršího požadovaného záznamu
- „casDO“ – datum a čas nejnovějšího požadovaného záznamu

w) getlogactionsbyiduzivatel?idUzivatel={idUzivatel}&casOD={casOD}&casDO={casDO}

GET metoda pro získání seznamu záznamů přístupů do systému a záznamů z online testů.

parametry GET metody:

- „idUzivatel“ – id uživatele, jehož logy chceme zobrazit
- „casOD“ – datum a čas nejstaršího požadovaného záznamu
- „casDO“ – datum a čas nejnovějšího požadovaného záznamu

x) getlogonlinetestbyiduzivatel?idUzivatel={idUzivatel}&casOD={casOD}&casDO={casDO}

GET metoda pro získání seznamu záznamů z online testů.

parametry GET metody:

- „idUzivatel“ – id uživatele, jehož logy chceme zobrazit
- „casOD“ – datum a čas nejstaršího požadovaného záznamu
- „casDO“ – datum a čas nejnovějšího požadovaného záznamu

y) GetUserIdBySchoolsLogin?schoolId={schoolId}&schoolLogin={schoolLogin}

GET metoda pro získání id uživatele se zadaným školním loginem pro zadanou školu.

- „schoolId“ – id školy
- „schoolLogin“ – školní login uživatele

z) GetStudentOfKurz?idCourseInfo={idCourseInfo}

GET metoda pro získání všech studentů vybraného kurzu.

parametry GET metody:

- „idCourseInfo“ – id kurzu ve vybraném roce

2 School

a) schoolsbyuser?userId={userId}

GET metoda pro získání všech dostupných škol daného studenta

parametry GET metody:

- „userId“ – id uživatele, pro kterého chceme školy získat

b) schoolinfobyyear?yearId={yearId}&schoolId={schoolId}

GET metoda pro získání jednoznačného identifikátoru konkrétní školy v konkrétním roce.

parametry GET metody:

- „yearId“ – id akademického roku
- „schoolId“ – id školy

3 Role

a) getroles?userId={userId}&schoolId={schoolId}

GET metoda pro získání seznamu rolí vybraného uživatele dané školy.

parametry GET metody:

- „userId“ – id uživatele, pro kterého chceme role získat
- „schoolId“ – id školy

4 Year

a) yearbyuser?userId={userId}&schoolId={schoolId}&roleId={roleId}

GET metoda pro získání roků, ve kterých byl daný uživatel v zadané škole a roli.

parametry GET metody:

- „userId“ – id uživatele, pro kterého chceme roky získat
- „schoolId“ – id školy

- „roleId“ – id role, ve které se daný uživatel nachází

5 Semester

a) semestersbycourseinfo?courseInfoId={courseInfoId}

GET metoda pro získání semestrů pro vybraný kurz.

parametry GET metody:

- „courseInfoId“ – id kurzu ve vybraném roce

6 Timetable

a) timetableget?semesterId={semesterId}&userId={userId}

GET metoda pro načtení všech hodin vybraného uživatele ve vybraném semestru.

parametry GET metody:

- „semesterId“ – id semestru
- „userId“ – id uživatele, pro kterého chceme rozvrh stáhnout

b) timetableentry?courseInfoId={courseInfoId}&userId={userId}

GET metoda pro načtení všech hodin vybraného uživatele ve vybraném kurzu.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id uživatele, pro kterého chceme rozvrh stáhnout

c) timetablelogin?studentId={studentId}&schoolClassId={schoolClassId}

POST metoda pro přihlášení vybraného studenta do vybrané třídy.

parametry adresy POST metody:

- „studentId“ – id studenta, kterého chceme do vybrané třídy přihlásit
- „schoolClassId“ – id vybrané třídy

d) timetablelogout?studentId={studentId}&schoolClassId={schoolClassId}

POST metoda pro odhlášení vybraného studenta z vybrané třídy.

parametry adresy POST metody:

- „studentId“ – id studenta, kterého chceme z vybrané třídy odhlásit
- „schoolClassId“ – id vybrané třídy

e) timetableinsert?userId={userId}&semesterId={semesterId}

POST metoda pro vytvoření hodiny v rozvrhu ve vybraném semestru.

parametry adresy POST metody:

- „userId“ – id uživatele, který hodinu vytváří
- „semesterId“ – id semestru

parametry POST metody:

- „Predmet“ – jméno předmětu
- „Zkratka“ – zkratka předmětu
- „Jmeno“ – jméno vyučujícího předmětu
- „Ucebna“ – učebna, ve kterém předmět bude probíhat
- „Den“ – den, ve kterém bude předmět probíhat
- „HodinaOD“ – začátek vyučovací hodiny
- „HodinaDO“ – konec vyučovací hodiny
- „Tyden“ – „Sudý“, „Lichý“ nebo „Každý“ týden
- „Typ“ – přednáška (hodnota „0“) nebo cvičení (hodnota „1“)

f) timetableupdate

POST metoda pro úpravu vybrané hodiny v rozvrhu.

parametry POST metody:

- „IDRozvrh“ – id vybrané hodiny v rozvrhu
- „Predmet“ – jméno předmětu
- „Zkratka“ – zkratka předmětu
- „Jmeno“ – jméno vyučujícího předmětu
- „Ucebna“ – učebna, ve kterém předmět bude probíhat
- „Den“ – den, ve kterém bude předmět probíhat
- „HodinaOD“ – začátek vyučovací hodiny
- „HodinaDO“ – konec vyučovací hodiny
- „Tyden“ – „Sudý“, „Lichý“ nebo „Každý“ týden
- „Typ“ – přednáška (hodnota „0“) nebo cvičení (hodnota „1“)

g) RemoveTimetable?timetableId={timetableId}

GET metoda pro odstranění vybrané hodiny z rozvrhu.

parametry GET metody:

- „timetableId“ – id vybrané hodiny v rozvrhu, kterou chceme odstranit

7 SolvedActivities

a) getall?userId={userId}&courseInfoId={courseInfoId}

GET metoda pro získání všech vyřešených aktivit daného uživatele ve vybraném kurzu.

parametry GET metody:

- „userId“ – id uživatele, pro kterého chceme vyřešené aktivity získat
- „courseInfoId“ – id kurzu v daném roce

b) Exists?activityId={activityId}&dateId={dateId}&studentId={studentId}

GET metoda pro zjištění, jestli vybraná aktivita byla již odevzdána.

parametry GET metody:

- „activityId“ – id aktivity
- „dateId“ – id termínu, ve kterém se aktivita nachází
- „studentId“ – id studenta, pro kterého stav aktivity chceme zjistit

c) getsolvedactivityfile?activityId={activityId}

GET metoda pro stažení souboru s řešením pro vybranou aktivitu.

parametry GET metody:

- „activityId“ – id aktivity

d) Post

POST metoda pro odevzdání nebo úpravu vyřešené aktivity.

parametry POST metody:

- „IDAktivitaVyresene“ – id již vyřešené aktivity, nebo nekladná číselná hodnota v případě prvního odevzdání vybrané aktivity
- „IDVlozil“ – id uživatele, který vkládá hodnocení
- „IDUzivatel“ – id uživatele, který chce aktivitu odevzdat
- „IDTermin“ – id termínu, ve kterém se aktivita nachází
- „IDAktivita“ – id aktivity
- „Hodnoceni“ – hodnocení aktivity
- „HodnoceniDatum“ – datum udělení hodnocení aktivity
- „DatumVlozeni“ – datum odevzdání aktivity
- „LoginSkola“ – školní login uživatele
- „LoginSystem“ – systémový login uživatele
- „Jmeno“ – jméno uživatele
- „Prijmeni“ – příjmení uživatele
- „TitulPred“ – titul uživatele uvedený před jménem
- „TitulZa“ – titul uživatele uvedený za jménem
- „NazevAktivita“ – název aktivity
- „DoporHodnoceni“ – maximální možné hodnocení
- „NazevTermin“ – název termínu
- „NazevSkupinaAktivit“ – název skupiny aktivit ve které se aktivita nachází

- „Reseni“ – text řešení
- „Poznamka“ – poznámka
- „ReseniSouborNazev“ – název souboru s řešením
- „ReseniSouborContent“ – typ souboru s řešením
- „ReseniSoubor“ – soubor s řešením v kódování Base64

e) postEvaluation

metoda POST pro udělení hodnocení aktivit.

parametry POST metody:

- „IDAktivitaVyresene“ – id aktivity, které chceme udělit hodnocení
- „IDVlozil“ – id uživatele, který vkládá hodnocení
- „Hodnoceni“ – hodnocení aktivity
- „HodnoceniDatum“ – datum udělení hodnocení aktivity

8 CourseConditions

a) activitydates?courseInfoId={courseInfoId}&studentId={studentId}

GET metoda pro získání termínů skupin aktivit, aktivit a testů.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „studentId“ – id studenta, pro kterého chceme termíny získat

b) currentaktivita?courseInfoId={courseInfoId}&userId={userId}

GET metoda pro získání aktuálních aktivit.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id uživatele, pro kterého chceme aktuality získat

c) logintodate?studentId={studentId}&dateId={dateId}

POST metoda pro přihlášení na termín.

parametry addresy POST metody:

- „studentId“ – id studenta
- „dateId“ – id termínu, na který se chce student přihlásit

d) logoutfromdate?studentId={studentId}&dateId={dateId}

POST metoda pro přihlášení na termín.

parametry adresy POST metody:

- „studentId“ – id studenta
- „dateId“ – id termínu, na který se chce student přihlásit

e) addloglist

POST metoda pro odeslání statistik z testu.

parametry POST metody:

- „TextLogs“ – pole následujících parametrů:
  - „QuestionId“ – id otázky
  - „QuestionOrder“ – pořadí otázky
  - „AnswerId“ – id odpovědi
  - „AnswerOrder“ – pořadí odpovědi
  - „CreatedAt“ – aktuální čas
  - „Checked“ – pravdivostní hodnota o zaškrtnutí dané otázky
  - „UserGeneratedTestId“ – id aktuálně vypracovaného testu

f) groupactivites?courseInfoId={courseInfoId}&userId={userId}&roleId={roleId}

GET metoda pro získání skupin aktivit.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id uživatele, pro kterého chceme skupiny aktivit získat
- „roleId“ – id role uživatele

g) currenttestbychapterrt?chapterId={chapterId}&studentId={studentId}

GET metoda pro získání cvičných testů.

parametry GET metody:

- „chapterId“ – id kurzu v daném roce
- „studentId“ – id studenta, pro kterého chceme cvičné testy získat

h) availabletests?courseInfoId={courseInfoId}&studentId={studentId}

GET metoda pro získání testů.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „studentId“ – id studenta, pro kterého chceme testy získat

9 ConsultationHours

a) ConsultationHoursWhichIsLogged?courseInfoId={courseInfoId}&studentId={studentId}

GET metoda pro získání konzultací, na které je student přihlášen.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „studentId“ – id studenta, pro kterého chceme konzultace získat

b) ConsultationHoursDatesToLogin?courseInfoId={courseInfoId}&studentId={studentId}

GET metoda pro získání dostupných konzultací.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „studentId“ – id studenta, pro kterého chceme konzultace získat

c) ConsultationHours/studentsconsultationhours?consultationHoursId={consultationHoursId}&exactDate={exactDate}

GET metoda pro získání studentů přihlášených na vybranou konzultační hodinu.

parametry GET metody:

- „consultationHoursId“ – id konzultační hodiny
- „exactDate“ – datum a čas opakující se konzultace (nepovinný parametr)

d) login

POST metoda sloužící k přihlášení studenta na vybranou konzultační hodinu.

parametry POST metody:

- „ConsultationHoursId“ – id konzultační hodiny
- „StudentId“ – id studenta, který se chce na konzultační hodinu přihlásit
- „ExactDate“ – datum a čas konzultace
- „Reason“ – důvod přihlášení na termín

e) logout

POST metoda sloužící k odhlášení studenta z vybrané konzultační hodiny.

parametry POST metody:

- „ConsultationHoursId“ – id konzultační hodiny
- „StudentId“ – id studenta, který se chce na konzultační hodinu přihlásit
- „ExactDate“ – datum a čas konzultace

f) getconsultationhoursdates?courseInfoId={courseInfoId}&userId={userId}

GET metoda pro získání seznamu konzultačních hodin pro vybraný kurz v daném roce.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id uživatele, pro kterého chceme konzultační hodiny získat

g) getoccupiedconsultationhoursdates?courseInfoId={courseInfoId}&userId={userId}

GET metoda pro získání seznamu nejbližších obsazených konzultačních hodin pro vybraný kurz v daném roce.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id uživatele, pro kterého chceme nejbližší obsazené konzultačních hodin získat

h) save

POST metoda pro vytvoření nebo úpravu konzultačních hodin.

parametry POST metody:

- „Id“ – id existující konzultační hodiny
- „GarantId“ – id garanta, který konzultační hodinu vyváří
- „CourseInfoId“ - id kurzu v daném roce
- „Description“ – popis konzultační hodiny
- „From“ – datum začátku konzultační hodiny
- „To“ – datum konce konzultační hodiny
- „FromHours“ – čas začátku konzultační hodiny
- „ToHours“ – čas konce konzultační hodiny
- „Place“ – místo konání konzultační hodiny
- „RepeatableType“ – typ opakování konzultační hodiny (týdně, měsíčně, 14 denní opakování, žádné opakování)
- „AlertByEmail“ – příznak určující notifikaci emailem
- „MaxStudents“ – kapacita konzultační hodiny

i) Remove?consultationHoursId={consultationHoursId}

GET metoda odstranění vybrané konzultační hodiny.

parametry GET metody:

- „consultationHoursId“ – id konzultační hodiny

10 News

a) newsbycourse?courseInfoId={courseInfoId}&ownerId={ownerId}&studentId={studentId}

GET metoda pro získání novinek v kurzu daného roku.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „ownerId“ – id autora novinky (nepovinný parametr)
- „studentId“ – id studenta (nepovinný parametr)

b) newsattachment?newsId={newsId}&attachmentId={attachmentId}

GET metoda pro získání přílohy vybrané novinky.

parametry GET metody:

- „newsId“ – id novinky ke které chceme přílohy získat
- „attachmentId“ – id přílohy

c) deleteattachment?attachmentId={attachmentId}

POST metoda pro smazání vybrané přílohy.

parametry adresy POST metody:

- „attachmentId“ – id přílohy

d) update

POST metoda pro úpravu novinky.

parametry POST metody:

- „IDNews“ – id novinky, kterou chceme upravit
- „IDUserInserted“ – id uživatele, který novinku vytvořil
- „DateInserted“ – datum a čas kdy byla novinky vytvořena
- „ValidFrom“ – datum a čas od kdy je novinka validní
- „ValidTo“ – datum a čas do kdy je novinka validní
- „Name“ – jméno novinky
- „Text“ – text novinky
- „IDUserModified“ – id uživatele, který novinku poslední modifikoval
- „DateModified“ – datum a čas kdy byla novinka naposledy modifikovaná
- „Attachments“ – pole příloh (volitelný parametr)
- „IDSchoolInfos“ – pole id škol v konkrétních letech, kde je novinka viditelná (volitelný parametr)

e) insert?schoolInfoId={schoolInfoId}&courseInfoId={courseInfoId}

POST metoda pro vytvoření novinky.

parametry adresy POST metody:

- „schoolInfoId“ – id školy v daném roce
- „courseInfoId“ – id kurzu v daném roce ve kterém chceme novinku vytvořit

parametry POST metody:

- „IDUserInserted“ – id uživatele, který novinku vytváří
- „DateInserted“ – datum a čas kdy je novinky vytvořena
- „ValidFrom“ – datum a čas od kdy je novinka validní
- „ValidTo“ – datum a čas do kdy je novinka validní
- „Name“ – jméno novinky
- „Text“ – text novinky
- „IDUserModified“ – id uživatele, který novinku poslední modifikoval
- „DateModified“ – datum a čas kdy byla novinka naposledy modifikovaná
- „Attachments“ – pole příloh (volitelný parametr)
- „IDSchoolInfos“ – pole id škol v konkrétních letech, ve kterých je novinka viditelná (volitelný parametr)

f) insertattachments?newsId={newsId}

POST metoda pro přidání příloh k novince.

parametry adresy POST metody:

- „newsId“ – id novinky, ke které chceme přílohy přidat

parametry POST metody:

- „[novinka, novinka, …]“ – pole novinek s parametry:
  - „IDNewsAttachment“ – id novinky, ke které chceme přílohy přidat
  - „File“ – soubor v kódování Base64
  - „FileName“ – jméno souboru
  - „FieContent“ – typ souboru

g) Remove?newsId={newsId}&schoolInfoId={schoolInfoId}

GET metoda pro odstranění novinky z vybrané školy v daném roce.

parametry GET metody:

- „newsId“ – id novinky
- „schoolInfoId“ – id školy v daném roce

11 GeneratedTest

a) generatedquestions?generatedTestId={generatedTestId}

GET metoda pro získání vygenerovaných otázek.

parametry GET metody:

- „generatedTestId“ – id vygenerovaného testu

b) generatedanswers?generatedQuestionId={generatedQuestionId}

GET metoda pro získání vygenerovaných odpovědí k vygenerované otázce.

parametry GET metody:

- „generatedQuestionId“ – id vygenerované otázky

c) getgeneratedquestionbyid?generatedQuestionId={generatedQuestionId}

GET metoda pro získání textu vygenerovaných otázek dle zadaného id.

parametry GET metody:

- „generatedQuestionId“ – id vygenerované otázky

d) generatetest

POST metoda pro vygenerování testu a odebrání pokusu.

parametry POST metody:

- „Test“:
  - „TestId“ – id cvičného, vypracovaného testu nebo „-1“
  - „Attempts“ – počet pokusů
  - „TemplateId“ – id šablony
  - „InclusionId“ – id zařazení testu (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
  - „Name“ – jméno testu
  - „TimeMinutes“ – čas na vypracování testu v minutách
  - „Max“ – maximum dosažitelných bodů
  - „Min“ – minimum potřebných bodů
  - „IPRange“ – rozsah IP adres ze kterých může být test vykonán
  - „ShowResult“ – příznak udávající, jestli mají být výsledky viditelné studentům
  - „CanSendResult“ – příznak udávající, jestli mohou být výsledky odeslány
  - „PaperTest“ – příznak značící, jestli se jedná o papírový test
  - „Required“ – příznak udávající, jestli je test povinný

    - „Deleted“ – příznak udávající, jestli je test smazaný
    - „DateId“ – id termínu
    - „DateName“ – název termínu
    - „DateAvailableFrom“ – datum dostupnosti testu OD
    - „DateAvailableTo“ – datum dostupnosti testu DO
    - „DateSendTo“ – datum pro nejpozdější odeslání testu
    - „GroupActivityName“ – jméno skupiny aktivit
    - „IsEditable“ – příznak udávající, jestli je test editovatelný
    - „Offline“ – příznak značící, jestli se jedná o offline test
- „StartAt“ – aktuální čas (čas vygenerování testu)
- „IPAddress“ – IP
- „StudentId“ – id studenta požadujícího vygenerování testu
- „CourseInfoId“ – id kurzu v daném roce

e) savegeneratetestkeyvalue

POST metoda pro uložení mezivýsledku, nebo výsledku testu.

parametry POST metody:

- „GeneratedTestId“ – id vygenerovaného testu
- „UserGeneratedTestId“ – id aktuálně vypracovaného testu
- „AnsweredQuestions“ – pole zodpovězených otázek s následujícími parametry:
    - „Key“:
        - „AuthorId“ – id autora otázky
        - „BlockId“ – id bloku
        - „CategoryId“ – id kategorie
        - „CategoryName“ – název kategorie
        - „ChapterName“ – název kapitoly
        - „Checked“ – příznak udávající, jestli je otázka překontrolována
        - „CheckedAt“ – datum kontroly
        - „CreatedAt“ – datum evidence
        - „Deleted“ – příznak udávající, jestli je otázka smazána
        - „EstimatedTimeForQuestion“ – předpokládaný čas strávený čtením otázky
        - „GeneratedQuestionId“ – id vygenerované otázky
        - „GeneratedTestId“ – id vygenerovaného testu
        - „Images“ – pole obrázků s atributy:
            - „IDObrazekOtazkaOdpoved“ – id obrázku
            - „IDOtazka“ – id otázky
            - „IDOdpoved“ – id odpovědi
            - „Soubor“ – obrázek v kódování Base64

- „SouborNazev“ – název souboru
- „SouborContent“ – typ souboru

- „InclusionId“ – id zařazení testu (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
- „IsAuthor“ – příznak udávající, jestli je uživatel autor testu
- „Question“ – text otázky
- „QuestionId“ – id otázky
- „AnswerType“ – typ odpovědi
- „RoleId“ – id role
- „UserIdWhoChecked“ – id uživatele, který test kontroloval

- „Value“:
  - „Answer“ – text odpovědi
  - „AnswerId“ – id odpovědi
  - „Deleted“ – příznak, jestli je odpověď smazaná
  - „EstimatedTimeForAnswer“ – předpokládaný čas strávený nad řešením otázky
  - „Help“ – vysvětlení
  - „Images“ – pole obrázků s atributy:
    - „IDObrazekOtazkaOdpoved“ – id obrázku
    - „IDOtazka“ – id otázky
    - „IDOdpoved“ – id odpovědi
    - „Soubor“ – obrázek v kódování Base64
    - „SouborNazev“ – název souboru
    - „SouborContent“ – typ souboru
  - „IsCorrect“ – příznak, jestli je otázka zodpovězena správně
  - „TextAnswerCorrect“ – správná odpověď

- „StudentId“ – id studenta vypracovávajícího test
- „SchoolInfoId“ – id školy v daném roce

f) viewgeneratedtest?generatedTestId={generatedTestId}&viewed={viewed}

POST metoda, která vrátí pokus za nevypracovaný test.

parametry adresy POST metody:

- „generatedTestId“ – id vygenerovaného testu
- „viewed“ – příznak značící, jestli student test viděl (hodnota „true“) nebo jestli se má odebrat pokus (hodnota „false“)

g) answersbygeneratedquestion?userGeneratedTestId={userGeneratedTestId}&generatedQuestionId={generatedQuestionId}

GET metoda pro získání co uživatel odpověděl na konkrétní otázku konkrétního testu.

parametry GET metody:

- „userGeneratedTestId“ – id testu vygenerovaného studentem
- „generatedQuestionId“ – id vygenerované otázky

h) generatedtestimages?userGeneratedTestId={userGeneratedTestId}

GET metoda pro získání oskenovaných testů.

parametry GET metody:

- „userGeneratedTestId“ – id testu vygenerovaného studentem

i) setvygenerovanytestuzivatelcorrect?userGeneratedTestId={userGeneratedTestId}&correct={correct}

POST metoda pro ohlášení chybně rozpoznaného testu studentem, nebo jeho opravení garantem.

parametry adresy POST metody:

- „userGeneratedTestId“ – id testu vygenerovaného studentem
- „correct“ – příznak, jestli je test správný. Při nahlášení chybně rozpoznaného testu student posílá hodnotu „false“ a po opravení garant posílá hodnotu „true“

j) cangenerate?testId={testId}&userId={userId}&terminId={terminId}

GET metoda značící, jestli se vybranému studentovi může vygenerovat další varianta testu – jestli má ještě volný pokus.

parametry GET metody:

- „testId“ – id testu
- „userId“ – id uživatele, který žádá o další vygenerování testu
- „terminId“ – id termínu, ve kterém se daný test nachází

k) getgeneratedtestbytestid?testId={testId}&dateId={dateId}

GET metoda pro získání vygenerovaných testů pro daný termín.

parametry GET metody:

- „testId“ – id testu
- „dateId“ – id termínu, pro který chceme testy získat

l) generatetestsprava?idUzivatel={idUzivatel}&idSablona={idSablona}&hlavni={hlavni}

POST metoda pro vygenerování testů.

parametry adresy POST metody:

- „idUzivatel“ – id uživatele, který test generuje
- „idSablona“ – id šablony, podle které se test generuje
- „hlavni“ – příznak, jestli se jedná o hlavní test či nikoli (vychází z id zařazení testu, ze kterého se generovaný test generuje)

parametry POST metody:

- „IDTest“ – id testu, ze kterého se generovaný test generuje
- „CasSpusteni“ – čas zavolání metody pro vygenerování testu
- „IP“ – IP adresa uživatele, který test generuje
- „IDTermin“ – id termínu, do kterého se test vygeneruje
- „DatumSpusteni“ – datum zavolání metody pro vygenerování testu

12 TestDrawn

a) completedtests?studentId={studentId}&courseInfoId={courseInfoId}

GET metoda pro získání testů, které student vypracoval v daném kurzu, v daném roce.

parametry GET metody:

- „studentId“ – id studenta, jehož testy chceme získat
- „courseInfoId“ – id kurzu v daném roce, ve kterém se nachází testy, které chceme získat

13 Course

a) publiccourses?userId={userId}

GET metoda pro načtení veřejných kurzů.

parametry GET metody:

- „userId“ – id studenta, což slouží k identifikaci, jestli je student ke kurzu přihlášen

b) coursebystudent?yearId={yearId}&studentId={studentId}

GET metoda pro získání kurzů v daném roce, na které je student přihlášen.

parametry GET metody:

- „yearId“ – id akademického roku ve kterém se kurzy nachází
- „studentId“ – id studenta, pro něhož chceme kurzy získat

c) coursebygarant?yearId={yearId}&garantId={garantId}

GET metoda pro získání kurzů v daném roce, ve kterých je uživatel v roli garanta.

parametry GET metody:

- „yearId“ – id akademického roku ve kterém se kurzy nachází
- „garantId“ – id garanta, pro něhož chceme kurzy získat

d) coursebytutor?yearId={yearId}&tutorId={tutorId}

GET metoda pro získání kurzů v daném roce, ve kterých je uživatel v roli tutora.

parametry GET metody:

- „yearId“ – id akademického roku ve kterém se kurzy nachází
- „tutorId“ – id tutora, pro něhož chceme kurzy získat

e) GetClassTimeLimitsFromTo?courseInfoId={courseInfoId}

GET metoda pro zjištění časového omezení přihlašování do tříd.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce

f) SetKurzLimits?courseInfoId={courseInfoId}&studyForm={studyForm}&min={min}&max={max}

GET metoda úpravu minima a maxima daného kurzu pro vybranou formu studia.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „studyForm“ – forma studia (prezenční – hodnota „0“, nebo kombinovaná – hodnota „1“)
- „min“ – minimum
- „max“ – maximum

g) GetKurzLimits?courseInfoId={courseInfoId}&studyForm={studyForm}

GET metoda pro získání minima a maxima kurzu pro danou formu.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „studyForm“ – forma studia (prezenční – hodnota „0“, nebo kombinovaná – hodnota „1“)

h) SetClassTimeLimits?courseInfoId={courseInfoId}

POST metoda pro časové omezení pro zápis do tříd.

parametry adresy POST metody:

- „courseInfoId“ – id kurzu v daném roce, pro který chceme časové limity nastavit

parametry POST metody:

- „zapisOD“ – čas začátku zápisu do tříd
- „zapisDO“ – čas konce zápisu do tříd

14 StudentEvaluation

a) getdata?courseInfoId={courseInfoId}&userId={userId}

GET metoda pro získání celkových výsledků konkrétního studenta ve vybraném kurzu v daném roce.

parametry GET metody:

- „courseInfoId“ – id kurzu v daném roce
- „userId“ – id studenta, pro kterého chceme celkové výsledky získat

15 EvaluationQuestion

a) GetEvaluationByGeneratedTest?userGeneratedTestId={userGeneratedTestId}

GET metoda pro získání výsledů všech otázek z vygenerovaného testu jednoho studenta.

parametry GET metody:

- „userGeneratedTestId“ – id testu vygenerovaného konkrétním studentem

16 Block

a) GetBlocks?templateId={templateId}

GET metoda pro získání bloků vybrané šablony testů.

parametry GET metody:

- „templateId“ – id šablony, pro kterou chceme bloky získat

b) GetBlockStructure?blockId={blockId}

GET metoda pro získání obsahu vybraného bloku konkrétní šablony testů.

parametry GET metody:

- „blockId“ – id bloku, jehož obsah chceme získat

c) InsertBlocks

POST metoda pro vytvoření šablony testů a jejích bloků.

parametry POST metody:

- „Template“:
  - „Nazev“ – název šablony
  - „Popis“ – popis šablony
  - „MichaniBloku“ – příznak udávající, jestli se mají bloky míchat
  - „MichatCele“ – příznak udávající, jestli se má zamíchat vše

    - „PocetBloku“ – počet bloků v šabloně
    - „IDKurzInfo“ – id kurzu v daném roce
- „Blocks“ – pole bloků s následujícími parametry:
    - „Block“:
        - „IDBlok“ – id bloku
        - „TypHodnoceni“ – typ hodnocení (1. nebo 2. způsob)
        - „VahaBloku“ – váha bloku v procentech
        - „SlozitostOD“ – spodní hranice obtížnosti v procentech
        - „SlozitostDO“ – horní hranice obtížnosti v procentech
        - „PocetOtazek“ – počet otázek
        - „Minimum“ – minimum
        - „TypOdpovedi“ – typ odpovědi („Právě 1 správná“, „Alespoň 1 správná“, „Odpověď do formuláře“, „Alespoň 1 správná a alespoň 1 špatná“)
        - „PocetOdpovedi“ – celkový počet odpovědí
        - „MinusProcento“ – procento, které se odečte za špatnou odpověď
        - „Nazev“ – název bloku
        - „MichaniVBloku“ – příznak udávající, jestli má být míchání v bloku povoleno
        - „NazevZapnout“ – příznakznačící, jestli má být zobrazen název bloku
        - „Smazany“ – příznak udávající, jestli je blok smazaný, či nikoliv
    - „BlockStructures“ – pole kategorií, kroků a otázek bloku s parametry:
        - „IDBlok“ – id bloku
        - „IDKategorie“ – id kategorie
        - „PocetOtazekKategorie“ – počet otázek v kategorii – parametr je povinný pouze v kategoriích
        - „IDKrok“ – id kroku – parametr je povinný pouze v krocích
        - „Podminka“ – podmínka – parametr je povinný pouze v krocích
        - „IDOtazka“ – id otázky – parametr je povinný pouze v otázkách
- „CourseInfoId“ – id kurzu v daném roce
- „UserId“ – id studenta, pro kterého chceme celkové výsledky získat

d) UpdateBlocks?templateId={templateId}

POST metoda pro úpravu bloků šablony testů.

parametry adresy POST metody:

- „templateId“ – identifikátor šablony, ve které chceme bloky upravit

parametry POST metody:

- pole s následujícími parametry:
    - „Block“:
        - „IDBlok“ – id bloku
        - „TypHodnoceni“ – typ hodnocení (1. nebo 2. způsob)
        - „VahaBloku“ – váha bloku v procentech
        - „SlozitostOD“ – spodní hranice obtížnosti v procentech
        - „SlozitostDO“ – horní hranice obtížnosti v procentech
        - „PocetOtazek“ – počet otázek
        - „Minimum“ – minimum
        - „TypOdpovedi“ – typ odpovědi („Právě 1 správná“, „Alespoň 1 správná“, „Odpověď do formuláře“, „Alespoň 1 správná a alespoň 1 špatná“)
        - „PocetOdpovedi“ – celkový počet odpovědí
        - „MinusProcento“ – procento, které se odečte za špatnou odpověď
        - „Nazev“ – název bloku
        - „MichaniVBloku“ – příznak udávající, jestli má být míchání v bloku povoleno
        - „NazevZapnout“ – příznakznačící, jestli má být zobrazen název bloku
        - „Smazany“ – příznak udávající, jestli je blok smazaný, či nikoliv
    - „BlockStructures“ – pole kategorií, kroků a otázek bloku s parametry:
        - „IDBlok“ – id bloku
        - „IDKategorie“ – id kategorie
        - „PocetOtazekKategorie“ – počet otázek v kategorii – parametr je povinný pouze v kategoriích
        - „IDKrok“ – id kroku – parametr je povinný pouze v krocích
        - „Podminka“ – podmínka – parametr je povinný pouze v krocích
        - „IDOtazka“ – id otázky – parametr je povinný pouze v otázkách

17 Chapter

a) chaptersbycourse?courseInfoId={courseInfoId}

GET metoda pro získání kapitol v kurzu v daném roce.

parametry GET metody:

- „courseInfoId“ – id kurzu v konkrétním roce

b) materials?chapterId={chapterId}

GET metoda pro získání materiálů pro vybranou kapitolu.

parametry GET metody:

- „chapterId“ – id kapitoly, pro kterou chceme materiály získat

c) insertchapter?courseInfoId={courseInfoId}

POST metoda pro vytvoření nové kapitoly.

parametry adresy POST metody:

- „courseInfoId“ – id kurzu v konkrétním roce

parametry POST metody:

- „IDKapitola“ – id kapitoly (kapitola v bez předka má hodnotu id „-1“)
- „Nazev“ – název kapitoly
- „ParentID“ – id rodičovské kapitoly, parametr je pro kapitolu bez rodiče volitelný
- „Smazana“ – příznak značící, jestli je kapitola smazaná

d) RemoveMaterial/{id}

GET metoda pro smazání materiálu.

parametry GET metody:

- „id“ – id materiálu, který má být odstraněn

e) updatechapter

POST metoda pro úpravu kapitoly.

parametry POST metody:

- „IDKapitola“ – id kapitoly (kapitola v bez předka má hodnotu id „-1“)
- „Nazev“ – název kapitoly
- „ParentID“ – id rodičovské kapitoly, parametr je pro kapitolu bez rodiče volitelný
- „Smazana“ – příznak značící, jestli je kapitola smazaná

f) deletechapter/{id}

POST metoda pro smazání kapitoly.

parametry adresy POST metody:

- „id“ – id kapitoly, která má být odstraněna

g) insertmaterial

POST metoda pro vytvoření nového materiálu.

parametry POST metody:

- „IDKapitola“ – id kapitoly, ve které se materiál nachází
- „Nazev“ – název kapitoly, ve kterém se materiál nachází
- „Soubor“ – materiál v kódování Base64

- „SouborNazev“ – název materiálu
- „SouborContent“ – typ materiálu

18 OnlineStatistics

a) InsertStatOtazkaUzivatelList

POST metoda pro vložení statistik pro otázky na server.

parametry POST metody:

- „QuestionStats“ – pole otázek s parametry
  - „IDOtazka“ – id otázky
  - „IDStatOtazkaUzivatel“ – id konkrétní statistiky dané otázky
  - „PocetZobrazeni“ – počet zobrazení otázky
  - „StravenyCas“ – čas strávený nad otázkou

b) InsertStatOdpovedUzivatelList

POST metoda pro vložení statistik pro odpovědi na server.

parametry POST metody:

- „AnswerStats“ – pole odpovědí s parametry
  - „IDOdpoved“ – id odpovědi
  - „PocetKliku“ – počet kliků ve výběru odpovědi
  - „IDStatOdpovedUzicatel“ – id konkrétní statistiky dané odpovědi
  - „IDVygenerovanyTestUzivatel“ - id aktuálně vypracovávaného testu

19 Date

a) GetDates?groupActivityId={groupActivityId}&testId={testId}&activityId={activityId}

GET metoda pro získání termínů skupiny aktivit, testů, nebo aktivit.

parametry GET metody:

- „groupActivityId“ – id skupiny aktivit, parametr je povinný pouze pro termíny skupin aktivit
- „testId“ – id testu, parametr je povinný pouze pro termíny testů
- „activityId“ – id aktivity, parametr je povinný pouze pro termíny aktivit

b) Remove/{id}

POST metoda pro smazání termínu.

parametry adresy POST metody:

- „id“ – id termínu, který chceme pomocí metody smazat

c) Update

POST metoda pro úpravu již existujícího termínu skupiny aktivit / aktivity / testu.

parametry POST metody:

- „IDTermin“ – id termínu, který chceme pomocí metody smazat
- „IDSkupinaAktivit“ – id skupiny aktivit, parametr je povinný pouze pro úpravu termínu skupiny aktivit, pro ostatní se používá hodnota „-1“
- „IDTest“ - id testu, parametr je povinný pouze pro úpravu termínu testu, pro ostatní se používá hodnota „-1“
- „IDAktivita“ - id aktivity, parametr je povinný pouze pro úpravu termínu aktivity, pro ostatní se používá hodnota „-1“
- „Nazev“ – název termínu
- „Ucebna“ - učebna
- „NazevTyp“ – název spolu s typem termínu (test / aktivita / skupina aktivit)
- „AktivniOD“ – datum a čas od kdy je termín aktivní
- „AktivniDO“ – datum a čas do kdy je termín aktivní
- „PrihlaseniDO“ - datum a čas do kdy je možné přihlášení na daný termín
- „OdevzdaniDO“ - datum a čas do kdy je možné vypracování (testu, aktivity, skupiny aktivit)
- „IDAutor“ – id autora termínu
- „DatumPrihlaseni“ – datum a čas od kdy je možné se na termín přihlásit
- „DatumOdhlaseni“ – datum a čas do kdy je možné se z termínu odhlásit
- „NutnePrihlaseni“ – příznak značící, jestli je na daný termín nutné přihlášení
- „PocetStudentu“ – maximální počet studentů, kteří se mohou na daný termín zapsat
- „Smazany“ – příznak značící, jestli je termín smazaný
- „PocetZapsanych“ – počet aktuálně zapsaných studentů na daný termín
- „AktivitaNazev“ – název skupiny aktivit ve které se daný termín nachází
- „PocetPokusu“ – počet pokusů na daný termín
- „Zapsan“ – příznak udávající, jestli je termín zapsán

d) Insert

POST metoda pro vložení nového termínu skupiny aktivit / aktivity / testu.

parametry POST metody:

- „IDTermin“ – id termínu, který chceme pomocí metody smazat
- „IDSkupinaAktivit“ – id skupiny aktivit, parametr je povinný pouze pro úpravu termínu skupiny aktivit, pro ostatní se používá hodnota „-1“

- „IDTest“ - id testu, parametr je povinný pouze pro úpravu termínu testu, pro ostatní se používá hodnota „-1“
- „IDAktivita“ - id aktivity, parametr je povinný pouze pro úpravu termínu aktivity, pro ostatní se používá hodnota „-1“
- „Nazev“ – název termínu
- „Ucebna“ - učebna
- „AktivniOD“ – datum a čas od kdy je termín aktivní
- „AktivniDO“ – datum a čas do kdy je termín aktivní
- „PrihlaseniDO“ - datum a čas do kdy je možné přihlášení na daný termín
- „OdevzdaniDO“ - datum a čas do kdy je možné vypracování (testu, aktivity, skupiny aktivit)
- „IDAutor“ – id autora termínu
- „DatumPrihlaseni“ – datum a čas od kdy je možné se na termín přihlásit
- „DatumOdhlaseni“ – datum a čas do kdy je možné se z termínu odhlásit
- „NutnePrihlaseni“ – příznak značící, jestli je na daný termín nutné přihlášení
- „PocetStudentu“ – maximální počet studentů, kteří se mohou na daný termín zapsat
- „AktivitaNazev“ – název skupiny aktivit ve které se daný termín nachází
- „PocetPokusu“ – počet pokusů na daný termín

e) LoginAllStudentsToDate?courseInfoId={courseInfoId}&dateId={dateId}&fulltime={fulltime}&combined={combined}

POST metody pro přihlášení všech prezenčních, kombinovaných, nebo všech studentů na termín vybraného kurzu v daném roce.

parametry adresy POST metody:

- „courseInfoId“ – id kurzu v konkrétním roce
- „dateId“ – id termínu, na který chceme studenty přihlásit
- „fulltime“ – příznak značící, jestli na termín budou přihlášeni všichni studenti prezenční formy studia
- „combined“ – příznak značící, jestli na termín budou přihlášeni všichni studenti prezenční formy studia

f) LogoutFromDateImmediately?studentId={studentId}&dateId={dateId}

POST metoda pro odhlášení z termínu pro potřeby garanta – umožňuje odhlásit studenta i po uzavření termínu.

parametry adresy POST metody:

- „studentId“ – id studenta, kterého chceme z termínu odhlásit
- „dateId“ – id termínu, ze kterého chceme studenta odhlásit

g) GetStudentsQRCodes?dateId={dateId}&skolaInfoId={skolaInfoId}

GET metoda pro získání PDF souboru s QR kódy studentů přihlášených na termín ve vybrané škole pro konkrétní školní rok.

parametry GET metody:

- „dateId“ – id termínu, ze kterého chceme získat PDF soubor s QR kódy studentů
- „skolaInfoId“ – id školy v konkrétním roce, kde se termín nachází

h) GetStudentspdf?dateId={dateId}&skolaInfoId={skolaInfoId}

GET metoda pro získání PDF souboru se seznamem studentů přihlášených na termín ve vybrané škole pro konkrétní školní rok.

parametry GET metody:

- „dateId“ – id termínu, ze kterého chceme získat PDF soubor se seznamem studentů
- „skolaInfoId“ – id školy v konkrétním roce, kde se termín nachází

i) GetStudents?dateId={dateId}&skolaInfoId={skolaInfoId}

GET metoda pro získání seznamu studentů přihlášených na termín ve vybrané škole pro konkrétní školní rok.

parametry GET metody:

- „dateId“ – id termínu, ze kterého chceme seznam studentů získat
- „skolaInfoId“ – id školy v konkrétním roce, kde se termín nachází

20 StudentSummary

a) GetSummaryByGroupActivity?groupActivityId={groupActivityId}

GET metoda pro získání souhrnných výsledků studentů pro danou skupinu aktivit.

parametry GET metody:

- „groupActivityId“ – id skupiny aktivit, ze které chceme souhrnné výsledky získat

b) GetSummaryByGroupActivityAndStudent?groupActivityId={groupActivityId}&userId={userId}

GET metoda pro získání výsledků vybraného studenta z dané skupiny aktivit.

parametry GET metody:

- „groupActivityId“ – id skupiny aktivit, ze které chceme souhrnné výsledky získat
- „userId“ – id studenta, pro kterého chceme souhrnné výsledky získat

c) GetSummaryByCourse?courseInfoId={courseInfoId}

GET metoda pro získání souhrnných výsledků studentů z daného kurzu v konkrétním roce.

parametry GET metody:

- „courseInfoId“ – id kurzu v konkrétním roce, odkud chceme souhrnné výsledky získat

d) GetSchoolClassesByStudent?studentId={studentId}&courseInfoId={courseInfoId}

GET metoda pro získání tříd z daného kurzu v konkrétním roce, do kterých je vybraný student zapsán.

parametry GET metody:

- „studentId“ – id studenta, pro kterého chceme získat seznam tříd do kterých je zapsán
- „courseInfoId“ – id kurzu v konkrétním roce, odkud chceme třídy získat

e) GetSummaryBySchoolClassFromTest?schoolClassId={schoolClassId}&testId={testId}

GET metoda pro získání souhrnných výsledků třídy z vybraného testu.

parametry GET metody:

- „schoolClassId“ – id třídy
- „testId“ – id testu, ze kterého chceme souhrnné výsledky třídy získat

21 EvaluationTest

a) PrioritizeEvaluationTest?evaluationTestId={evaluationTestId}

POST metoda pro upřednostnění vybraného hodnocení testu.

parametry adresy POST metody:

- „evaluationTestId“ – id hodnocení testu

22 GroupActivities

a) updategroupactivity

POST metoda pro úpravu skupiny aktivit.

parametry POST metody:

- „IDSkupinaAktivit“ – id skupiny aktivit, kterou chceme upravit
- „PocetPokusu“ – počet pokusů na danou skupinu aktivit
- „Nazev“ – název skupiny aktivit
- „IDRole“ – id role toho, kdo skupinu aktivit spravuje (Garant – hodnota „3“, nebo Tutor – hodnota „4“)

- „IDKurzInfo“ – id kurzu v konkrétním roce, ve kterém se skupina aktivit nachází
- „Minimum“ – minimum bodů skupiny aktivit
- „Maximum“ – maximum bodů skupiny aktivit
- „Povinny“ – příznak, jestli je skupina aktivit povinná
- „ZapocitatDoVysledku“ – příznak, jestli se výsledek skupiny aktivity započítává do celkového hodnocení
- „FormaStudia“ – forma studia (Prezenční – hodnota „0“, Kombinovaná – hodnota „1“)
- „Nulovat“ – příznak značící, jestli se mají negativní výsledky nahradit nulou
- „Smazana“ – příznak značící, jestli je skupina aktivit smazaná
- „IsEditable“ – příznak značící, jestli je skupina aktivit dále editovatelná

b) getgroupactivitiesbycourse?courseInfoId={courseInfoId}&userId={userId}&roleId={roleId}

GET metoda pro získání skupin aktivit vybraného kurzu v konkrétním roce pro daného uživatele v zadané roli.

parametry GET metody:

- „courseInfoId“ – id kurzu v konkrétním roce, odkud chceme skupiny aktivit získat
- „userId“ – id uživatele, pro kterého chceme skupiny aktivit získat
- „roleId“ – id role, ve které se uživatel nachází

c) insertgroupactivities?userId={userId}

POST metoda pro vytvoření nové skupiny aktivit.

parametry adresy POST metody:

- „userId“ – id uživatele, který chceme skupinu aktivit vytvořit

parametry POST metody:

- pole skupin aktivit s následujícími parametry:
    - „IDSkupinaAktivit“ – u nové skupinu aktivit se používá hodnota „-1“
    - „PocetPokusu“ – počet pokusů na danou skupinu aktivit
    - „Nazev“ – název skupiny aktivit
    - „IDRole“ – id role toho, kdo skupinu aktivit spravuje (Garant – hodnota „3“, nebo Tutor – hodnota „4“)
    - „IDKurzInfo“ – id kurzu v konkrétním roce, ve kterém se skupina aktivit nachází
    - „Minimum“ – minimum bodů skupiny aktivit
    - „Maximum“ – maximum bodů skupiny aktivit
    - „Povinny“ – příznak, jestli je skupina aktivit povinná

- „ZapocitatDoVysledku“ – příznak, jestli se výsledek skupiny aktivity započítává do celkového hodnocení
- „FormaStudia“ – forma studia (Prezenční – hodnota „0“, Kombinovaná – hodnota „1“)
- „Nulovat“ – příznak značící, jestli se mají negativní výsledky nahradit nulou
- „Smazana“ – příznak značící, jestli je skupina aktivit smazaná
- „IsEditable“ – příznak značící, jestli je skupina aktivit dále editovatelná

d) Remove/{id}

POST metoda pro smazání skupiny aktivit.

parametry adresy POST metody:

- „id“ – id skupiny aktivit, kterou chceme smazat

23 Activity

a) GetActivities?groupActivityId={groupActivityId}&userId={userId}&roleId={roleId}

GET metoda pro získání seznamu aktivit v dané skupině aktivit pro zadaného uživatele v dané roli.

parametry GET metody:

- „groupActivityId“ – id skupiny aktivit, ze které chceme aktivity získat
- „userId“ – id uživatele
- „roleId“ – id role uživatele

b) Remove/{id}

POST metoda pro odstranění aktivity.

parametry adresy POST metody:

- „id“ – id aktivity, kterou chceme odstranit

c) Insert?groupActivityId={groupActivityId}&userId={userId}

POST metoda pro vytvoření nové aktivity.

parametry adresy POST metody:

- „groupActivityId“ – id skupiny aktivit, ve které chceme aktivitu vytvořit
- „userId“ – id uživatele, který vytváří novou aktivitu

parametry POST metody:

- „IDAktivita“ – u nové aktivity se používá hodnota „-1“
- „PocetPokusu“ – počet pokusů na vypracování dané aktivity
- „Nazev“ – název aktivity
- „Popis“ – slovní popis aktivity
- „VzorVysl“ – vzorový výsledek aktivity
- „DoporHodnoceni“ – doporučené hodnocení aktivity
- „Minimum“ – minimum bodů aktivity
- „Povinny“ – příznak, jestli je aktivita povinná
- „Vybiratelnost“ – příznak možnosti výběru aktivity
- „VyberDoData“ – datum a čas do kdy je možné si aktivitu vybrat
- „OmezeniStudentu“ – maximální počet studentů na aktivitu
- „TypOmezeni“ – omezení se může týkat pouze třídy – hodnota „0“, nebo celého kurzu – hodnota „1“
- „PopisSoubor“ – popis aktivity v souboru zakódovaném Base64
- „PopisSouborNazev“ – název souboru pro popis aktivity
- „PopisSouborContType“ – typ souboru pro popis aktivity
- „VzorVyslSoubor“ – vzorový výsledek v souboru zakódovaném Base64
- „VzorVyslSouborNazev“ – název souboru se vzorovým výsledkem aktivity
- „VzorVyslSouborContType“ – typ souboru se vzorovým výsledkem aktivity
- „Smazana“ – příznak značící, jestli je aktivita smazaná
- „IsEditable“ – příznak značící, jestli je aktivita dále editovatelná

d) GetActivitySampleFile?activityId={activityId}

GET metoda pro získání souboru se vzorovým výsledkem pro danou aktivitu.

parametry GET metody:

- „activityId“ – id aktivity, pro kterou chceme soubor získat

e) GetActivityInfoFile?activityId={activityId}

GET metoda pro získání souboru s popisem dané aktivitu.

parametry GET metody:

- „activityId“ – id aktivity, pro kterou chceme soubor získat

f) Update

POST metoda pro úpravu již existující aktivity.

parametry POST metody:

- „IDAktivita“ – id upravované aktivity
- „PocetPokusu“ – počet pokusů na vypracování dané aktivity

- „Nazev“ – název aktivity
- „Popis“ – slovní popis aktivity
- „VzorVysl“ – vzorový výsledek aktivity
- „DoporHodnoceni“ – doporučené hodnocení aktivity
- „Minimum“ – minimum bodů aktivity
- „Povinny“ – příznak, jestli je aktivita povinná
- „Vybiratelnost“ – příznak možnosti výběru aktivity
- „VyberDoData“ – datum a čas do kdy je možné si aktivitu vybrat
- „OmezeniStudentu“ – maximální počet studentů na aktivitu
- „TypOmezeni“ – omezení se může týkat pouze třídy – hodnota „0“, nebo celého kurzu – hodnota „1“
- „PopisSoubor“ – popis aktivity v souboru zakódovaném Base64
- „PopisSouborNazev“ – název souboru pro popis aktivity
- „PopisSouborContType“ – typ souboru pro popis aktivity
- „VzorVyslSoubor“ – vzorový výsledek v souboru zakódovaném Base64
- „VzorVyslSouborNazev“ – název souboru se vzorovým výsledkem aktivity
- „VzorVyslSouborContType“ – typ souboru se vzorovým výsledkem aktivity
- „Smazana“ – příznak značící, jestli je aktivita smazaná
- „IsEditable“ – příznak značící, jestli je aktivita dále editovatelná

24 Test

a) GetTests?groupActivitiesId={groupActivitiesId}&userId={userId}&roleId={roleId}

GET metoda pro získání seznamu testů v dané skupině aktivit pro zadaného uživatele v dané roli.

parametry GET metody:

- „groupActivitiesId“ – id skupiny aktivit, ze které chceme testy získat
- „userId“ – id uživatele
- „roleId“ – id role uživatele

b) Remove/{id}

POST metoda pro odstranění testu.

parametry adresy POST metody:

- „id“ – id testu, který chceme odstranit

c) Insert?groupActivityId={groupActivityId}&userId={userId}

POST metoda pro vytvoření nového testu.

parametry adresy POST metody:

- „groupActivityId“ – id skupiny aktivit, ve které chceme test vytvořit
- „userId“ – id uživatele, který vytváří novou aktivitu

parametry POST metody:

- „IDTest“ – u nového testu se používá hodnota „-1“
- „PocetPokusu“ – počet pokusů na daný test
- „IDSablona“ – id šablony, podle které je test vytvořen
- „IDZarazení“ – id zařazení testu (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
- „Nazev“ – název testu
- „Cas“ – čas na vypracování testu
- „MaxBody“ – maximum bodů testu
- „MinBody“ – minimum bodů testu
- „RozsahIP“ – rozsah IP adres, ze kterých může být test vykonán
- „ZobrazHotovTest“ – příznak udávající, jestli mají být výsledky viditelné studentům
- „OdeslaniVysl“ – příznak udávající, jestli mohou být výsledky odeslány
- „PapirovyTest“ – příznak značící, jestli se jedná o papírový test
- „Offline“ – příznak značící, že lze vykonat test offline
- „Povinny“ – příznak udávající, jestli je test povinný
- „NazevSA“ – jméno skupiny aktivit
- „IsEditable“ – příznak udávající, jestli je test editovatelný

d) Update

POST metoda pro úpravu stávajícího testu.

parametry POST metody:

- „IDTest“ – id testu, který chceme upravit
- „PocetPokusu“ – počet pokusů na daný test
- „IDSablona“ – id šablony, podle které je test vytvořen
- „IDZarazení“ – id zařazení testu (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
- „Nazev“ – název testu
- „Cas“ – čas na vypracování testu
- „MaxBody“ – maximum bodů testu
- „MinBody“ – minimum bodů testu
- „RozsahIP“ – rozsah IP adres, ze kterých může být test vykonán

- „ZobrazHotovTest“ – příznak udávající, jestli mají být výsledky viditelné studentům
- „OdeslaniVysl“ – příznak udávající, jestli mohou být výsledky odeslány
- „PapirovyTest“ – příznak značící, jestli se jedná o papírový test
- „Offline“ – příznak značící, že lze vykonat test offline
- „Povinny“ – příznak udávající, jestli je test povinný
- „NazevSA“ – jméno skupiny aktivit
- „IsEditable“ – příznak udávající, jestli je test editovatelný

25 Step

a) GetSteps?categoryId={categoryId}

GET metoda pro získání všech kroků v dané kategorii otázek.

parametry GET metody:

- „categoryId“ – id kategorie otázek, ze kterých chceme kroky získat

b) Insert?categoryId={categoryId}

POST metoda pro vytvoření kroků v dané kategorii otázek.

parametry adresy POST metody:

- „categoryId“ – id kategorie do které máme v úmyslu kroky vkládat

parametry POST metody:

- pole kroků s parametry:
  - „Nazev“ – název kroku
  - „Obtiznost“ – složitost kroku
  - „Poradi“ – pořadí kroku

c) Update

POST metoda pro aktualizaci stávajících a současně vkládání nových kroků.

parametry POST metody:

- pole kroků s parametry:
  - „IDKrok“ – id kroku který aktualizujeme
  - „Nazev“ – název kroku
  - „Obtiznost“ – složitost kroku
  - „Poradi“ – pořadí kroku

26 Question

a) getquestions?categoryId={categoryId}&inclusionId={inclusionId}&chapterId={chapterId}&confirmed={confirmed}&userId={userId}&onlyOwn={onlyOwn}

GET metoda pro získání všech dostupných otázek v dané kategorii. Používá se v šablonách.

parametry GET metody:

- „categoryId“ – id kategorie, ze které chceme otázky získat
- „inclusionId“ – id zařazení (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
- „chapterId“ – id kapitoly
- „confirmed“ – příznak udávající, jestli mají být stažené pouze překontrolované otázky
- „userId“ – id uživatele, který chce otázky získat
- „onlyOwn“ – příznak udávající, jestli jsou požadovány pouze otázky, u kterých je uživatel vlastníkem

b) getquestionsallfull?categoryId={categoryId}&inclusionId={inclusionId}&chapterId={chapterId}&confirmed={confirmed}&unconfirmed={unconfirmed}&userId={userId}&onlyOwn={onlyOwn}

GET metoda pro získání všech dostupných otázek v dané kategorii. Používá se v otázkách.

parametry GET metody:

- „categoryId“ – id kategorie, ze které chceme otázky získat
- „inclusionId“ – id zařazení (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
- „chapterId“ – id kapitoly
- „confirmed“ – příznak udávající, jestli mají být stažené překontrolované otázky
- „unconfirmed“ – příznak udávající, jestli mají být stažené nepřekontrolované otázky
- „userId“ – id uživatele, který chce otázky získat
- „onlyOwn“ – příznak udávající, jestli jsou požadovány pouze otázky, u kterých je uživatel vlastníkem

c) insertquestion

POST metoda pro vytvoření nové otázky.

parametry POST metody:

- „question“:

- „IDZarazeni“ – id zařazení testu (Nevybírat podle zařazení – hodnota „-1“, Hlavní – hodnota „1“, Cvičný (á) – hodnota „2“, Ukázkový (á) – hodnota „3“)
- „IDKategorie“ – id kategorie
- „IDUzivatel“ – id uživatele, který otázku vytváří
- „IDRole“ – id role, ve které uživatel vystupuje
- „Zadani“ – text otázky
- "PredpCasPrecteni" – předpokládaný čas potřebný k přečtení otázky
- "Smazana" – příznak, jestli je otázka smazaná
- "DatumEvidence" – datum vytvoření otázky
- "IDUzivatelKontrola" – id uživatele, který otázku zkontroloval
- "DatumKontrola" – datum kontroly otázky
- "Prekontrolovano" – příznak, jestli je otázka překontrolována
- "KategorieNazev" – název kategorie, ve které se otázka nachází
- "KapitolaNazev" – název kapitoly, ve které se otázka nachází
- „Images“ – pole obrázků s atributy:
  - „IDObrazekOtazkaOdpoved“ – id obrázku
  - „IDOtazka“ – id otázky
  - „IDOdpoved“ – id odpovědi
  - „Soubor“ – obrázek v kódování Base64
  - „SouborNazev“ – název souboru
  - „SouborContent“ – typ souboru

- „stepsId“ – pole identifikátoru kroků pro danou otázku:
  - „id“ – id krokuu
- „images“ – pole obrázků s atributy:
  - „IDObrazekOtazkaOdpoved“ – id obrázku
  - „IDOtazka“ – id otázky
  - „IDOdpoved“ – id odpovědi
  - „Soubor“ – obrázek v kódování Base64
  - „SouborNazev“ – název souboru
  - „SouborContent“ – typ souboru

d) Remove/{id}

GET metoda pro odstranění otázky.

parametry GET metody:

- „id“ – id otázky, kterou chceme odstranit

27 Template

a) GetTemplates?courseInfoId={courseInfoId}&userId={userId}&roleId={roleId}

GET metoda pro získání šablon testů v daném kurzu v konkrétním roce pro zadaného uživatele ve vybrané roli.

parametry GET metody:

- „courseInfoId“ – id kurzu v konkrétním roce, odkud chceme šablony získat
- „userId“ – id uživatele, pro kterého chceme šablony testů získat
- „roleId“ – id role, ve které se uživatel nachází

b) GetTemplatesWithBlocks?courseInfoId={courseInfoId}&userId={userId}&roleId={roleId}

GET metoda pro získání šablon testů s počty bloků v daném kurzu v konkrétním roce pro zadaného uživatele ve vybrané roli.

parametry GET metody:

- „courseInfoId“ – id kurzu v konkrétním roce, odkud chceme šablony získat
- „userId“ – id uživatele, pro kterého chceme šablony testů získat
- „roleId“ – id role, ve které se uživatel nachází

c) Update

POST metoda pro úpravu již existující šablony.

parametry adresy POST metody:

- „IDSablona“ – id upravované šablony
- „Nazev“ – název šablony
- „Popis“ – popis šablony
- „MichaniBloku“ – příznak udávající, jestli se mají bloky míchat
- „MichatCele“ – příznak udávající, jestli se má zamíchat vše
- „PocetBloku“ – počet bloků v šabloně
- „IDKurzInfo“ – id kurzu v daném roce

d) Remove/{id}

POST metoda pro odstranění šablony.

parametry adresy POST metody:

- „id“ – id šablony, kterou chceme odstranit

28 Student

a) GetStudentSummaryByIDTermin?dateId={dateId}&skolaInfoId={skolaInfoId}&minPoints={minPoints}

GET metoda získání výsledků vybraného termínu ve zvolené škole v konkrétním roce.

parametry GET metody:

- „dateId“ – id termínu, ze kterého chceme výsledky získat
- „skolaInfoId“ – id školy v konkrétním roce, odkud chceme výsledky získat
- „minPoints“ – minimální počet bodů výsledků, které chceme získat

b) GetStudentHistoryByIDTermin?dateId={dateId}

GET metoda pro získání historie přihlašování na daný termín.

parametry GET metody:

- „dateId“ – id termínu, ze kterého chceme historii získat

29 Category

a) GetCategories?chapterId={chapterId}

GET metoda pro získání kategorií v dané kapitole.

parametry GET metody:

- „chapterId“ – id kapitoly, pro kterou chceme kategorie získat

b) Insert?chapterId={chapterId}

POST metoda pro vytvoření kategorie ve vybrané kapitole.

parametry adresy POST metody:

- „chapterId" – název kapitoly, ve které chceme kategorii vytvořit

parametry POST metody:

- „Nazev" – název kategorie
- "Popis" – popis kategorie
- "Rovnice" – složitost pro vkládanou kategorii
- "Hodnota" – hodnota maximální složitosti kroku pro danou kategorii

c) Update?chapterId={chapterId}

POST metoda pro úpravu existující kategorie ve vybrané kapitole.

parametry adresy POST metody:

- „chapterId" – název kapitoly, ve které chceme kategorii upravit

parametry POST metody:

- „IDKategorie" – id kategorie, kterou chceme upravit
- „Nazev" – název kategorie
- "Popis" – popis kategorie
- "Rovnice" – složitost pro upravovanou kategorii

- "Hodnota" – hodnota maximální složitosti kroku pro danou kategorii
- "Smazana" – příznak udávající, jestli je kategorie smazaná

d) Remove/{id}

POST metoda pro odstranění kategorie.

parametry adresy POST metody:

- „id“ – id kategorie, kterou chceme odstranit

30 Answer

a) Insert

POST metoda pro vytvoření odpovědí ke konkrétní otázce.

parametry POST metody:

- „answerList“ – pole odpovědí s parametry:
  - „IDOdpoved“ – pro nově vkládanou otázku se použije hodnota „-1“
  - „Spravna“ – příznak, jestli je otázka zodpovězena správně
  - „TextOdpovedi“ – text odpovědi
  - „Vysvetleni“ – vysvětlení, proč je odpověď označena jako správná skutečně správná
  - „PredpCasReseni“ – předpokládaný čas strávený nad řešením otázky
  - „Smazana“ – příznak, jestli je odpověď smazaná
- „questionId“ – id otázky
- „images“:
  - pořadí odpovědi: pole obrázků dané odpovědi s atributy:
    - „IDObrazekOtazkaOdpoved“ – id obrázku
    - „IDOtazka“ – id otázky
    - „IDOdpoved“ – id odpovědi
    - „Soubor“ – obrázek v kódování Base64
    - „SouborNazev“ – název souboru
    - „SouborContent“ – typ souboru

31 SchoolClass

a) getgarantclasses?courseInfoId={courseInfoId}

GET metoda pro získání tříd kurzu pro garanta kurzu.

parametry GET metody:

- „courseInfoId“ – id kurzu v konkrétním roce

b) insertschoolclass?courseInfoId={courseInfoId}

POST metoda pro vytvoření nové třídy.

parametry adresy POST metody:

- „courseInfoId“ – id kurzu v konkrétním roce

parametry POST metody:

- „IDTutor“ – id uživatele, který je pro danou třídu tutorem
- „Nazev“ – název předmětu
- „ZkratkaPredmet“ – zkratka předmětu
- „Ucebna“ – učebna
- „HodinaOD“ – čas začátku hodiny
- „HodinaDO“ – čas konce hodiny
- „Tyden“ – „Sudý“, „Lichý“ nebo „Každý“ týden
- „Typ“ – přednáška (hodnota „0“) nebo cvičení (hodnota „1“)
- „Omezeni“ – maximální počet studentů
- „FormaStudia“ – forma studia (Prezenční – hodnota „0“, Kombinovaná – hodnota „1“)
- „Den“ – den, v týdnu, kdy daná hodina probíhá (hodnoty „Pondělí“ - „Sobota“)
- „CombinedClasses“ – v případě, že se jedná o kombinovanou formu studia, tak je povinné pole tutoriálů s následujícím parametry:
    - „IDTutor“ – id uživatele, který je pro danou třídu tutorem
    - „TutorName“ – jméno tutora
    - „TutorSurname“ – příjmení tutora
    - „Nazev“ – název předmětu
    - „ZkratkaPredmet“ – zkratka předmětu
    - „Ucebna“ – učebna
    - „Den“ – den, v týdnu, kdy daná hodina probíhá (hodnoty „Pondělí“ - „Sobota“)
    - „HodinaOD“ – čas začátku hodiny
    - „HodinaDO“ – čas konce hodiny
    - „Tyden“ – „Sudý“, „Lichý“ nebo „Každý“ týden
    - „Typ“ – přednáška (hodnota „0“) nebo cvičení (hodnota „1“)
    - „Omezeni“ – maximální počet studentů
    - „FormaStudia“ – forma studia (Kombinovaná – hodnota „1“)
    - „Datum“ – datum, kdy daná hodina proběhne

c) updateschoolclass

POST metoda pro úpravu existující třídy.

parametry POST metody:

- „IDTrida“ – id třídy, jejíž hodnoty chceme upravit
- „IDTutor“ – id uživatele, který je pro danou třídu tutorem

- „Nazev“ – název předmětu
- „ZkratkaPredmet“ – zkratka předmětu
- „Ucebna“ – učebna
- „HodinaOD“ – čas začátku hodiny
- „HodinaDO“ – čas konce hodiny
- „Tyden“ – „Sudý“, „Lichý“ nebo „Každý“ týden
- „Typ“ – přednáška (hodnota „0“) nebo cvičení (hodnota „1“)
- „Omezeni“ – maximální počet studentů
- „FormaStudia“ – forma studia (Prezenční – hodnota „0“, Kombinovaná – hodnota „1“)
- „Den“ – den, v týdnu, kdy daná hodina probíhá (hodnoty „Pondělí“ - „Sobota“)
- „CombinedClasses“ – v případě, že se jedná o kombinovanou formu studia, tak je povinné pole tutoriálů s následujícím parametry:
    - „IDTutor“ – id uživatele, který je pro danou třídu tutorem
    - „TutorName“ – jméno tutora
    - „TutorSurname“ – příjmení tutora
    - „Nazev“ – název předmětu
    - „ZkratkaPredmet“ – zkratka předmětu
    - „Ucebna“ – učebna
    - „Den“ – den, v týdnu, kdy daná hodina probíhá (hodnoty „Pondělí“ - „Sobota“)
    - „HodinaOD“ – čas začátku hodiny
    - „HodinaDO“ – čas konce hodiny
    - „Tyden“ – „Sudý“, „Lichý“ nebo „Každý“ týden
    - „Typ“ – přednáška (hodnota „0“) nebo cvičení (hodnota „1“)
    - „Omezeni“ – maximální počet studentů
    - „FormaStudia“ – forma studia (Kombinovaná – hodnota „1“)
    - „Datum“ – datum, kdy daná hodina proběhne

d) Remove/{id}

GET metoda pro odstranění třídy.

parametry GET metody:

- „id“ – id třídy, kterou chceme odstranit

32 datamining

a) https://eLogika.vsb.cz/datamining/statistics.aspx?user={user}&course={course}&year={year}

GET metoda pro získání dat pro datamining.

parametry GET metody:

- „user“ – id uživatele, pro kterého chceme data získat
- „course“ – id kurzu, pro který chceme data získat
- „year“ – id roku ve kterém se nachází vybraný kurz

33 EmailMessage

a) Messages

POST metoda pro získání emailových zpráv z databáze.

parametry POST metody:

- „Id“ – id zprávy, kterou chceme získat – pro získání více zpráv se použije hodnota „-1“
- „UserId“ – id uživatele, pro kterého chceme data získat
- „CourseInfoId“ – id kurzu v konkrétním roce
- „From“ – datum a čas nejstarší možné požadované zprávy
- „To“ – datum a čas nejnovější možné požadované zprávy
- „Email“ – text zprávy

b) Remove?messageId={messageId}

POST metoda pro odstranění záznamu o zprávě.

parametry adresy POST metody:

- „messageId“ – id záznamu o zprávě, který chceme odstranit

c) SendToMultipleUsers

POST metoda pro odeslání emailové zprávy s přílohami jednomu či více uživatelům.

parametry POST metody:

- „Sender“ – email uživatele, který chce zprávu odeslat
- „Receiver“ – emaily uživatelů (oddělených „;“, kterým má být zpráva odeslána)
- „UserId“ – id uživatele, který zprávu odesílá
- „CourseInfoId“ – id kurzu v konkrétním roce
- „SentAt“ – datum a čas odeslání
- „Subject“ – předmět zprávy
- „Body“ – text zprávy
- „Attachments“ – pole příloh s následujícími parametry:
    - "Id“ – id nové přílohy je označeno hodnotou „-1“
    - "Name" – jméno souboru (příloha)

- "Content" – soubor (příloha) v kódování Base64
- "ContentType" – typ souboru (příloha)

d) SendToMultipleClasses

POST metoda pro odeslání emailové zprávy s přílohami jedné či více třídám.

parametry POST metody:

- „Sender“ – email uživatele, který chce zprávu odeslat
- „UserId“ – id uživatele, který zprávu odesílá
- „CourseInfoId“ – id kurzu v konkrétním roce
- „SentAt“ – datum a čas odeslání
- „Subject“ – předmět zprávy
- „Body“ – text zprávy
- „Attachments“ – pole příloh s následujícími parametry:
  - "Id“ – id nové přílohy je označeno hodnotou „-1“
  - "Name" – jméno souboru (příloha)
  - "Content" – soubor (příloha) v kódování Base64
  - "ContentType" – typ souboru (příloha)
- „IdClasses“ – pole s identifikátory tříd, jejímž studentům se má zpráva odeslat

34 ExportResult

a) ExportResult?testId={testId}&dateId={dateId}&schoolInfoId={schoolInfoId}

POST metoda pro export výsledků z vybraného termínu

parametry POST metody:

- „testId“ – id testu, ze kterého mají být výsledky exportovány
- „dateId“ – id termínu, ze kterého mají být výsledky exportovány
- „schoolInfoId“ – id školy v konkrétním roce