NÁZEV PŘÍKLADNÉ ZÁVĚREČNÉ PRÁCE

Bc. Tomáš Sládek

Diplomová práca
Fakulta informačních technologií
České vysoké učení technické v Praze
Katedra softwarového inženýrství
Študijný program: Informatika
Špecializácia: Webové inženýrství
Vedúci: Ing. Josef Pavlíček, Ph.D.
26. januára 2026

Replace the contents of this file with official assignment.
Místo tohoto souboru sem patří list se zadáním závěrečné práce.

Chtěl bych poděkovat především sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Vyhlásenie

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použitéinformační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravěvysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

Prohlašuji, že jsem v průběhu příprav a psaní závěrečné práce použil/-a nástroje umělé inteligence. Vygenerovaný obsah jsem ověřil/-a. Stvrzuji, že jsem si vědom/-a, že za obsah závěrečné práce plně zodpovídám.

V Prahe dňa 26. januára 2026

Abstrakt

Fill in the abstract of this thesis in Czech. Lorem ipsum dolor sit amet. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Kľúčové slová enter, comma, separated, list, of, keywords, in, CZECH

Abstract

Fill in the abstract of this thesis in English. Lorem ipsum dolor sit amet. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Keywords enter, comma, separated, list, of, keywords, in, ENGLISH

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov kódu

Zoznam skratiek

DFA	Deterministic Finite Automaton
FA	Finite Automaton
LPS	Labelled Prüfer Sequence
NFA	Nondeterministic Finite Automaton
NPS	Numbered Prüfer Sequence
XML	Extensible Markup Language
XPath	XML Path Language
XSLT	eXtensible Stylesheet Language Transformations
W3C	World Wide Web Consortium

Kapitola 1

Teoretický základ

1.1 Softvérové inžinierstvo

1.1.1 Životný cyklus vývoja softvéru SDLC

Výber správnej metodológie pre SDLC kladie dôležitú rolu na úspech projektu. SDLC metodológie ponúkajú kostru ktorá sprevádza projekt naprieč návrhom, vývojom a implementáciou softvérového riešenia. SDLC je proces ktorý načŕta fázy a aktivity zapojené do tvorby softvéru od kenceptu cez development po údržbu. Medzi typické fázy SDLC patrí:

Plánovanie definuje rozsah a požiadavky projektu, indetifikuje kľúčových členov a ich role a vytvára plán projektu.

Analýza získava a okumentuje podrobné požiadavky od používateľov a kľúčových členov. Analyzuje zozbierané potreby na pochopenie funkčných a nefunkčných požiadavkov systému.

Dizajn vytvára architektúru aplikácie, definujúc vzájomné interakcie komponent. Navrhuje podobu používateľského rozhrania a používateľských skúsenosti.

Implementácia obsahuje vlastnú tvorbu kódu podľa výstupov z predchádzajúcich fáz napr., požiadavky, návrh UI a ďalších.

Testovanie prebieha viacerých úrovniach od testovania jednotlivých častí až po systémové testy, ktoré testujú systém ako celok. V rámci testovania sa verifikuje splnenie požiadaviek.

Nasadenie prebieha primárne v dvoch fázach, nasadenie do testovacieho prostredia a do ostrého prostredia. V rámci nasadenia sa rieši finálne prostredie v ktorom budú softvér využívať používatelia.

Údržba a podpora sa zameriava na monitorovanie a údržbu softvéru v produkčnom prostredí. Prebiehajú v nej opravy na chyby ktoré sa nenašli v rámci predchádzajúcich fáz. Prípadne sem môžu patriť vylepšenia na základe spätnej väzby od používateľov.

Metodológie SDLC sa líšia primárne v prístupe k týmto fázam. Medzi najnámejšie patrí vodopádová metóda, špirálová metóda, iteratívna metóda, inkrementálna metóda, prototypová metóda, metóda v tvare V, metóda rýchlej aplikácie a agilné metódy.

Vodopád je typický lineárnym priechodom naprieč fázami, kde každá fáza začína po skončení predchádzajúcej. Tento prístup je vhodný na projekty s dobre definovanými požiadavkami na začiatku projektu, a prápadné zmeny sú len minimálné. Pevná štruktúra tejto metódy sa ťažko prispôsobuje zmenám po skončení fáze. Výhodou tejto metódy je dobrá odhadnuteľnosť prostriedkov a času na projekt.

Špiralová metóda rozdeľuje projekt do série iterativných cyklov. V každom cykle sa postupne prejde cez všetky fáze. Na konci cyklu sa spraví vyhodnotenie a spolu so spätnou väzbou vstupujú do plánovania ďalšieho cyklu. Toto umožňuje vyššiu flexibiltu na zmeny. Hlavnou vlastnosťou špirálového modelu je jeho zemranie sa na analýzu rizík. Nevýhodou špirálového modelu je jeho komplexnosť a náročnosť na prostriedky. Zároveň mu chýbajú dobre stavené ciele oproti vodopádovej metóde.

Itertívna metóda sa zameriava na postupný vývoj a zdokonalenie softvéru pomocou iterativných cyklov. Umožňuje kontinuálne zlepšovanie a adaptáciu na zmenu požiadavkov. Výhodou je vytvorenie čiastočného no funkčného produktu na konci každej iterácie. Nevýhodou však môže byť prerastanie rozsahu projektu počas vývojového procesu čo môže viesť k prekročeniu rozpočtu a neskorým časom dodania.

Inkrementálna metóda rozdelí na začiatku projekt na ucelené celky. Tieto celky sú následne vyvíjané nezávisle a inrementy sú sekvenčne integrované do projektu. Hlavnou výhodou je možnosť paralalelizmu, no toto zároveň prináša väčsie nároky na koordinovanie tímu. Táto metóda je vhodná len na projekty ktoré umožňujú rozdelenie na nezávislé časti.

Prototypová metóda je charakterizovaná čo najrýchlejším vytvorením prototypuna získanie spätnej väzby od používateľov. Prototypová metóda obsahuje takisto viacero iterácií, kde spätná väzba z predchádzajúcej iterácie pomáha dotvárať požiadavky na ďalšiu iteráciu. Pravidelná spätná väzba prispieba k zlepšeniu komunikácie medzi vývojovým tímom a koncovými používateľmi. Medzi nevýhody patrí náročný odhad rozpočtu a dĺžky práce, s neznámim počtom iterácií vopred. Zároveň môžu používatelia nadobudnúť pocit že sa jedná o hotový systém aj keď v realite ide iba o prototyp.

Metóda v tvare V je nadstavbou nad klasickou vodopádovou metódou. Oproti vodopádovej metóde obsahuje proces verifikácie a validácie ktorý je aktívny paralelne s hlavnými fázami vodopádovej metódy. Každá vývojová fáza má prislúchajúcu testovaciu fázu a až po skončení oboch sa prechádza do

ďalšej fáze. Takto nastavený proces znižuje riziko vynechania nejakej veci pred pokračovaním vo vývoji.

Metóda rýchlej aplikácie je druh inkrementálnej metódy so zameraním sa na čo najrýchlejšie vytvorenie aplikácie. Často sú používané nástroje na tvorbu kódu a vlastný kód je držaný na minime. Zároveň sa zameriava na modularizáciu softvéru a paralelný vývoj nezávislých modulov čo rapídne znižuje dobu vývoja. Táto metóda je vhodná pre väčšie tími kde je dostatok vývojarov na paralelný vývoj softvéru.

Agilná metodológia reprezentuje rodinu iteratívnych a inkrementálnych metód ktoré prioritizujú adaptabilitu a sploprácu so zákazníkom. V základe rozdeľuje projekt na šprinty. Jeden šprint trvá typicky 1 až 4 týždne a predchádza mu scrum, čo je stretnutie na synchronizáciu vykonávanej práce. Agilný prístup zdieľa väčšinu výhod a nevýhod z ostatných iteratívnych metód.

1.1.2 Zber požiadaviek

Softvérové požiadavky predstavujú spôsob akým môžme zachytiť potreby zákazníka. Upresňujú nám rozsah projektu ako do funkcionilty ktorú zákazník očakáva, tak aj do formy akou budú tie funkcionality využité. Zároveň nám môžu upresňovať rozsah aj z opačnej strany v zmysle funkcionalít a vlastností ktoré softvér obsahovať nebude. Dotvára to celkový obraz zákazníkovi a predchádza sa pomocou toho nedorozumeniam.

Dôležitou úlohou na začiatku projeku je získavanie a dokumentácia požiadaviek. Získavanie požiadaviek je proces ktorý prebieha v kooperácií so zákazníkom, kľúčovými členmi a používateľmi. Existuje viacero spôsobom ktorými sme schopní zozbierať tieto požiadavky.

Rozhovory jeden na jednoho
Skupinové rozhovory
Brainstorming
Cieľová skupina
Dotazník
workshop požiadaviek
Sledovanie používateľa
Analýza rozhrania
Analýza dokumentov
Spätné inžinierstvo
Vytváranie prototypov

1.1.2.1 Kategorizácia požiadaviek

Po zozbieraní požiadviek, prichádza na rad rozdelenie požiadaviek podľa rôznych metrík. Hlavnými metrikami zvyknú bývať dôležitosť požiadavky, na ktorú sa používa kategorizácia MoSCoW. Táto kategorizácia delí požiadavky na tie ktoré musia byť zapracované do softvéru, teda must have, tie ktoré by

mali byť zapracované do systému, teda should have, tie ktoré by mohli byť vo výslednom softvéri, teda could have a do poslednej kategorizácie spadajú tie ktoré nebudú implementované do softvéru, teda won't have.

Ďalšou metrikou je FURPS+, ktoré delí požiadavky na funkčné, používateľné, spoľahlivé, výkonostné a

1.2 Webové inžinierstvo

1.2.1 Klient-server architektúra

Klient-server architektúra je sieťový model v ktorom dve hlavné entity, klienti a server, komunikujú medzi sebou za účeľom splenenie špecifickych úloh alebo zdieľania dát. Klient iniciuje požiadavku, čaká na odpoveď serveru a následne zobrazí dáta používateľovi. Server spracúva požiadavky, získa potrebné dáta a pošle ich klientovi.

Výhodou tejto architektúry je centralizovaná správa prostriedkov a dát. Táto výhoda sa zároveň stáva aj nevýhodou v podobe jediného bodu zlyhania. V prípade zlyhania serveru dochádza k nedostupnosti prostriedkov a dát.

Architekúra môže byť rozšírená o ďalšie prvky medzi serverom a klientom v podobe middlewaru.

1.2.2 API

API, aplikačné programovacie rozhranie slúži na komunikácie medzi programami.

1.2.2.1 SOUP API

1.2.2.2 RESTful API

1.2.2.3 Graph API

1.2.2.4 gRPC API

1.2.3 OAuth

1.2.4 Ako služba

1.2.4.1 Typy služieb

1.2.5 Backend ako služba

1.3 Tvorba používateľského rozhrania

1.3.1 Základné prvky používateľského rozhrania

Štvrtou heuristikou Jakoba Nielsna je udržanie konzistencie a štandardy pri návrhu používateľského rozhrania. Z tohto dôvodu ustálili základné prvky, ktoré sa používajú v rámci používateľského rozhrania.

1.3.2 Prototyp

Prototyp slúži na otestovanie použiteľnosti používateľského rozhrania pred tým ako investujeme prostriedky do finálneho riešenia. Podľa vernosti s ktorou sa podobajú na finálny produkt rozlišujeme dve kategórie prototypov. Vernosť môže byť v rôznych oblastiach interaktivita, vizuály alebo obsah a príkazy. Existujú dva druhy prototypov podľa množstva prostriedkov potrebných na ich zostavenie.

Prototyp s nízkou vernosťou umožňuje jednoduché úpravy vďaka jeho jednoduchosti. Jednoduchosť prototypu umožňuje predstavenie základnej štruktúry používateľského rozhrania. Na druhú stranu tento prototyp neposkytuje žiadne takmer žiadnu interaktivitu pre používateľa a všetky prechody sa dejú počas testovania manuálne.

Prototyp s vysokou vernosťou poskytuje reálnejšie odozvy systému. Pri testovaní prototypu s vysokou vernosťou môže nastať nedorozumenie, keď sa môže výsledný systém zdať pripravený na použitie aj keď sa jedná len o prototyp.

1.3.3 Persóny

Persońa je fiktívny, ale realistický popis typického používateľa produktu.

1.4 Testovanie používateľského rozhrania

1.4.1 Testovania mladistvým

https://www.nngroup.com/articles/usability-testing-minors/ https://www.nngroup.com/articles/usab
of-websites-for-teenagers/ https://www.nngroup.com/articles/childrens-websites-
usability-issues/ https://www.nngroup.com/articles/kids-cognition/

Kapitola 2
Analýza

2.1 Stupne Napredovania

Stupne napredovania sú programovou časťou slovenského skautingu. Tvoria základný kameň programu a osobného rastu. Sú určené pre deti a dospievajúcich od 4 rokov až po 24 rokov. Program je rozčlenený do nasledujúcich vekových kategórií:

4 - 6 rokov Janík a Grétka

7 - 10 rokov Vlčia stopa

11 - 14 rokov Skautský chodník

15 - 18 rokov Rangerský horizont

19 - 24 rokov Roverský svet

11+ rokov Nováčik - pre všetkých nových členov v Slovenskom skautingu.

2.1.1 Skautský chodník

Skautský chodník sa delí na tri časti. Prvý skaut, Neznáme cesty a Posledný vrchol. Každá z týchto troch častí by mala zabrať rok.

Skautský chodník sa zameriava na komplexný rozvoj jedinca naprieč rôznymi oblasťami. Pokrýva rozvoj zo svojho vnútra cez duchovný, citový a intelektuálny rozvoj, k rozvoju svojho vonkajšku pomocou telesného rozvoju až ku širšiemu sociálnemu rozvoju a buduje charakter. Pre jednoduchšie uchopenie je tento rozvoj koncipovaný do 4 kategórií.

Charakter (modrá) je kategória zameraná na rozvoj osobnosti a hodnôt. Hlavnou súčasťou tejto kategórie je sociálny aspekt a to najmä v rámci družiny a oddielu.

Služba (červená) je kategória zameraná na ekológiu, rodinu a život v prírode a meste, spoznávanie rozmanitosti iných kultúr, zlepšovanie komunikačných schopností a duchovného dedičstva skautingu.

Zdravie a sila (žltá) sa zaoberá fyzickou kondíciou a životom na tábore.

Schopnosti a zručnosti (zelená) je dedikovaná intelektuálnemu rozvoju a základným schopnostiam ktoré by mal poznať každý ako napríklad varenie.

Každá kategória obsahuje kapitoly ktoré predávajú čitateľovi myšlienku a snažia sa ho niečo nové naučiť. Záverom kapitoly je sada úloh ktoré musí skaut splniť na to aby splnil celú kategóriu. V prípade že by skautovi prišli úlohy moc náročné alebo nemožné splniť, môže požiadať radcu družiny alebo vedúceho o upravenie danej úlohy. Úprava úlohy by mala zachovať obtiažnosť a tému pôvodnej úlohy.

Vyhodnocovanie úloh záleží od jednotlivých úloh. Existujú tri druhy overenia splnenia úlohy. Overenie radcom družiny alebo vedúcim, overenie rodičom skauta a overenie skautom samotným pri čom sa spolieha na jeho skautskú česť. Na úlohách sa cení snaha viac ako samotný výsledok úlohy.

Po splnení určitého množstva úloh sa odomyká skautovi možnosť plniť výzvu z danej kategórie. Každá kategória má vlastnú výzvu.

2.1.1.1 Prvý skaut

Prvý skuat predstavuje prvú časť skautského chodníka. Očakávaný vek skauta ktorý plní Prvého skauta je medzi 11 a 12 rokov. Veku skauta odpovedá aj forma ktorou je vedený Prvý skaut. V rámci Prvého skauta sú všetky úlohy predom dané a skaut nemá žiadnu voľbu čo by plnil. Kapitoly obshaujú veľa rozprávania a témam sa venujú skôr povrchne. Výzvy sa odomykajú po splnení všetkých úloh v danej kategórii.

2.1.1.2 Neznáme cesty

Neznáme cesty predstavujú druhú časť skautského chodníka na ktorý nadväzujú. Oproti Prvému skautovi poskytujú skautovi väčšiu voľnosť vo výbere úloh. Každá úloha má počet bodov ktoré skaut dosiahne za jej splnenie. Cieľom každej kategórie je nazbieranie potrebného počtu bodov z úloh z danej kategórie. Každá kapitola obshuje jednu povinnú úlohu a sadu nepovinných úloh z ktorých si môže skaut vybrať. Výzvy sa odomykajú po splnení povinných úloh z danej kategórie.

2.1.1.3 Posledný vrchol

Posledný vrchol je posledná a záverečná časť skautského chodníka. Predstavuje najväčšiu volňnosť vo výbere úloh z celého skautského chodníka v podobe povinných a voliteľných kapitól. Obsahom kapitol je podobná Neznámym cestám. Na rozdiel od Neznámych ciest v rámci Posledného vrcholu stačí skutovi naz-

bierať celkový počet naprieč všetkými kategóriami. Má takto najväčšiu volňnosť vo výslednej podobe jeho priechodu Posledným vrcholom.

2.1.2 Rangerský horizont

Rangerský horizont je koncipovaný na maximálne 3 roky. Skladá sa z piatich častí. Osobný rozvoj I, osobný rozvoj II, expedícia I, expedícia II a rangerský projekt. Osobný rast I a II predstavuje podobnú sériu úloh s akou sa skaut stretol počas plnenia skautského chodníka. Slúži na ďalšie prekonávanie samého seba. Expedícia sa zameriava zas na spoznávanie okolia. Na splnenie expedície si je treba vybrať druh expedície na ktorý sa skuat vydá, xy pre I a za pre II. Projekt je určený na tímové riešenie danej problematiky ktorú si rangery zvolia a schváli im ju vodca.

2.2 Odborky

Ďalšou programovou časťou sú odborky, ktoré fungujú nezávisle na stupňoch napredovania. Odborky predstavujú súbory úloh, ktorých splnenie vedie k nadobudnutiu odbornosti v danej téme. Delia sa do dvoch stupňov, zelený a červený. Pokiaľ odborka obsahuje zelený stupeň, treba ho získať pred splnením červeného stupňa. Odborky sú podobne ako stupne napredovania rozdelené do viacerých vekových kategórií. Sú prispôsobené charakteristikám a potrebám vekovej kategórie.

2.3 Výzvy

Výzvy sú programovou časťou zameranou na prekonávanie samého seba. Vedú skuatov mimo ich komfortnú zónu a vystavujú ich situáciam ktoré nie sú bežné. Výzvy obsahujú ciele ktoré by mali skauti naplniť. Trvanie výziev je rôzne, pri niektorých sa jedná o jednorázovú aktivitu a pri iných zas o dlhodobejšie venovanie sa danej tématike.

2.4 Voľné programové moduly

Dobrovoľným rozšírením programu sú voľné programové moduly. Formou aj obsahom sa najviac približujú ku výzvam.

2.5 Najvyššie programové ocenenia

Vyvrcholením programovej činnosti pre vekovú kategóriu je najvyššie programové ocenenie, ktoré sa získava po splnení ostatných častí programu pre danú vekovú kategóriu.

2.6 Prípady použitia

UC-1: **Moje napredovanie**
Aktér: Skaut
Po otvorení aplikácie sa zobrazia úlohy ktoré si používateľ zvolil ako sledované. Zo zoznamu úloh si môže pridať nové úlohy medzi sledované. Po dokončení úlohy sa mu úloha odstráni zo sledovaných.

UC-2: **Vlastná úloha**
Aktér: Skaut, Radca
Skaut zvolí úlohu ktorú chce nahradiť, navrhne nové znenie, úplne nové alebo úprava stávajúceho, a pošle ho radcovi na schválenie. V procese schvaľovania môžu ešte znenie novej úlohy pozmeniť. Keď úloha zodpovedá očakávanej náročnosti, tak ju schváli a skautovi sa pridá medzi aktívne úlohy.

UC-3: **Splnenie úlohy**
Aktér: Skaut, Radca
Po splnení úlohy môže nastať schvaľovanie splnenia zo strany radcu. Radca môže zadať hromadné splnenie úlohy členom svojej družiny.

UC-4: **Moja družina**
Aktér: Radca
Používateľovi sa zobrazí postup všetkých členov jeho družiny.

UC-5: **Môj zbor**
Aktér: Zborový
Používateľovi sa zobrazia všetci členovia jeho zboru a môže spravovať ich priradenie do družín. Nastavuje Družinových radcov.

UC-6: **Úprava programu**
Aktér: Programový
Pri zmenách vo vdelávacio výchovnom programe môže programový vedúci zmeniť, pridať alebo odstrániť časti programu. Prípadne môže vytvoriť úplne nový program alebo zrušiť už existujúci.

2.7 Zber požiadaviek

Zber požiadaviek bol realizovaný .

2.7.1 Požiadavky

REQ-1: **Registrácia**
FURPS+: Funkčný
MoSCoW: Musí mať
Aplikácia umožní registráciu nového používateľa.

REQ-2: **Prihlásenie**
FURPS+: Funkčný
MoSCoW: Musí mať
Aplikácia umožní prihlásenie poučívateľa do jeho používateľského účtu.

REQ-3: **Pridanie sa do družiny**
FURPS+: Funkčný
MoSCoW: Musí mať

REQ-4: **Pridanie do družiny pomocou QR kódu**
FURPS+:
MoSCoW:

REQ-5: **Prezeranie programovej ponuky**
FURPS+: Funkčný
MoSCoW: Musí mať

REQ-6: **Pridanie časti programu do wishlistu**
FURPS+: Funkčný
MoSCoW: Musí mať

REQ-7: **Začať program z ponuky/wishlistu**
FURPS+: Funkčný
MoSCoW: Musí mať

REQ-8: **Sledovanie splnených bodov programu**
FURPS+: Funkčný
MoSCoW: Musí mať

REQ-9: **Moja družina - skaut**
FURPS+:
MoSCoW:

REQ-10: **Moja družina - radca**
FURPS+:
MoSCoW:

REQ-11: **Upozornenie na začatie červeného stupňa odborky bez absolvovaného zeleného stupňa**

FURPS+:
MoSCoW:

REQ-12: **Zobrazenie výzvy**
FURPS+:
MoSCoW:

REQ-13: **Poslat progres druziny na mail**
FURPS+:
MoSCoW:

Kapitola 3
Návrh

Táto kapitola sa venuje návrhu frontendu a backendu. Návrh frontendu sa skladá z LoFi a HiFi prototypu. Návrh backandu sa skladá z návrhu API štruktúry.

Kapitola 4

Implementácia

Kapitola 5

Testovanie

Kapitola 6

Záver

6.1 Budúci vývoj

Dodatok A

Nějaká příloha

Sem přijde to, co nepatří do hlavní části.

Obsah příloh