**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**

# Diplomová práce

## Analýza a návrh informačního systému pro sportovní klub s využitím UML

**Bc. Vojtěch Karas**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Vojtěch Karas

Informatika

Název práce

**Analýza a návrh informačního systému pro sportovní klub s využitím UML**

Název anglicky

**Analysis and design of an information system for a sports club using UML**

---

**Cíle práce**

Cílem diplomové práce je navrhnout model informačního systému za použití modelovacího jazyku UML pro vybraný sportovní klub. Informační systém bude zaměřen na správu interních záležitostí v rámci daného sportovního klubu. Dílčí cíle práce jsou:
- Vymezení pojmů v rámci literární rešerše k tématu návrhu informačních systémů
- Návrh modelů, kterými jsou objektový model, stavový model a model interakcí
- Návrh drátěných modelů a následně i funkčního prototypu

Vedlejším cílem práce je analýza aktuálního řešení ve vybraném sportovním klubu a jeho následné porovnání s modelem informačního systému navrhovaným v rámci této diplomové práce.

**Metodika**

Metodika práce vychází ze studia odborných literárních a internetových zdrojů. Na základě nabytých poznatků bude navržen vhodný model systému pro správu interních záležitostí sportovního klubu. Pro návrh modelu informačního systému budou použity metody softwarového inženýrství a modelovací jazyk UML. Pomocí UML bude navržen objektový model, stavový model a model interakcí navrhovaného informačního systému. Následně budou dle navržených UML modelů zkonstruovány drátěné modely pro návrh designu uživatelského rozhraní. Podle nich bude poté vytvořen funkční prototyp informačního systému. V poslední řadě bude provedena analýza aktuálního řešení ve vybraném sportovním klubu a bude provedeno porovnání s navrženým modelem.

**Doporučený rozsah práce**

60-80

**Klíčová slova**

informační systém, sportovní klub, návrh, analýza, prototyp, UML, IS

---

**Doporučené zdroje informací**


BRUCKNER, T. *Tvorba informačních systémů : principy, metodiky, architektury.* Praha: Grada, 2012. ISBN 978-80-247-4153-6.

BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. Příklady modelů analýzy a návrhu aplikace v UML. Praha: Oeconomica, 2013. Vysokoškolská učebnice. ISBN 978-80-245-1922-7.

FOWLER, Martin. Destilované UML. Praha: Grada, 2009. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. Brno: Computer Press, 2004. ISBN 80-251-0231-9.

TILLEY, Scott a Harry J. ROSENBLATT. Systems Analysis and Design. Boston, MA, United States: Cengage Learning, Inc, 2016. ISBN 9781305494602.

VRANA, I. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. PROVOZNĚ EKONOMICKÁ FAKULTA, – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. KATEDRA INFORMAČNÍHO INŽENÝRSTVÍ. *Projektování informačních systémů s UML.* V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta, 2008. ISBN 978-80-213-1817-5.


---

**Předběžný termín obhajoby**

2022/23 LS – PEF

**Vedoucí práce**

Ing. Martin Pelikán, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 26. 11. 2022

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

**doc. Ing. Tomáš Šubrt, Ph.D.**

Děkan

V Praze dne 06. 12. 2022

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Analýza a návrh informačního systému pro sportovní klub s využitím UML" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.3.2023 _____

**Poděkování**

Rád bych touto cestou poděkoval Ing. Martinu Pelikánovi, Ph.D. za vedení mé diplomové práce a trpělivost, cenné rady a čas, který mi věnoval v průběhu psaní této práce. Dále bych rád poděkoval vedení klubu SK Viktoria Sibřina za ochotu zhostit se role zadavatele informačního systému navrhovaného v rámci této diplomové práce.

# Analýza a návrh informačního systému pro sportovní klub s využitím UML

**Abstrakt**

Tato diplomová práce se zabývá analýzou a návrhem informačního systému pro sportovní klub v jazyce UML. Modelovaný informační systém má poskytovat podporu pro správu sportovního klubu se zacílením na evidenci událostí a jejich účastníků.

V teoretické části práce jsou zpracovány teoretická východiska k problematice softwarového inženýrství, návrhu informačních systémů, analýzy požadavků, UML, systémového modelovaní a návrhu uživatelského rozhraní.

V praktické části této diplomové práce je nejdříve představena vize navrhovaného informačního systému. Poté je provedena analýza požadavků zadavatele, na základě které je následně vytvořen návrh systému. Ten tvoří model tříd, stavový model a model interakcí. Detailně jsou pak zpracovány pohledy na zvolenou podmnožinu funkcionalit, která je určena jako klíčová pro vývoj prvotní verze navrhovaného systému.

V závěru práce je návrh systému doplněn o interaktivní prototyp, který implementuje všechny funkcionality z vybrané podmnožiny pro prvotní verzi systému. Na závěr jsou autorem formulovány doporučení pro implementaci.

Výstupem práce je návrh systému za pomoci UML modelů a interaktivní prototyp informačního systému.

**Klíčová slova:** informační systém, analýza, návrh, modelování, prototyp, uživatelské rozhraní, UML, IS, model tříd, stavový model, model interakcí, wireframe, sportovní klub

# Analysis and design proposal of an information system for a sports club using UML

**Abstract**

This thesis deals with the analysis and design of a UML information system for a sports club. The modelled information system is supposed to provide support for the administration of a sports club with a focus on the registration of events and their participants.

In the theoretical part of the thesis, the theoretical background to software engineering, design of information systems, requirements analysis, UML, system modelling and user interface design are presented.

In the practical part of this thesis, the vision of the proposed information system is first presented. Then, an analysis of the client's requirements is performed based on which the system design is subsequently developed. System design consists of a class model, a state model and an interaction model. Detailed views of the selected subset of functionalities, which is identified as crucial for the development of the initial version of the proposed system, are then elaborated.

At the end the system design is complemented by an interactive prototype that implements all the functionalities from the selected subset for the initial version of the system. Finally, recommendations for implementation are formulated by the author.

The output of the thesis is a system design using UML models and an interactive prototype of the information system.

**Keywords**: information system, analysis, design, modeling, prototype, user interface, UML, IS, class model, state model, interaction model, wireframe, sports club

# Obsah

# Seznam obrázků

# Seznam tabulek



# Seznam použitých zkratek

EIS – Enterprise Information Systems
DSS – Decision Support Systems
MIS – Management Information System
TPS – Transaction processing system
BI – Business Intelligence
RAD – Rapid Application Development
OOAD – Object-Oriented Analysis and Design
RUP – Rational Unified Process
EUP – Enterprise Unified Process
ASD – Adaptive Software Development
FDD – Feature-Driven Development
UML – Unified Modeling Language
OMG – Object Management Group
MDA – Model Driven Architecture
UI – User Interface
UX – User Experience
GUI – Graphical User Interace
CLI – Command Line Interface
Lo-Fi – Low Fidelity
Hi-Fi – High Fidelity (Věrný/Reálný model)
GDPR – General Data Protection Regulation (Obecné nařízení o ochraně osobních údajů)

# 1 Úvod

V současně době je již naprosto nemožné realizovat evidenci a správu dat v jakémkoliv odvětví bez pomoci informačních a komunikačních technologií. Žádná firma či jen společnost pro volnočasovou aktivitu se dnes bez této podpory neobejde. Největšími pomocníky v případě práce s daty pak bývají právě informační systémy. Informační systém by v každém podniku měl pomáhat zjednodušovat, zrychlovat a celkově optimalizovat vybrané procesy, na které cílí zaměření informačního systému. Cílem jejich využívání je tedy větší efektivita podniku, pracovníků apod. V případě podniků, které mají za cíl maximalizovat svůj zisk, mohou IS pomáhat se snížením nákladů, optimalizací firemní struktury, vylepšováním péče o zákazníky nebo s plánem na zvýšení tržeb. Informační systémy jsou ale potřeba i v případech, kdy není pro podnik či společnost hlavním cílem generovat zisk, i když ten je nepochybně spojen i s těmito odvětvími. To je ale právě případ, na který se zaměřuje tato diplomová práce.

Tato diplomová práce se zabývá analýzou a návrhem informačního systému pro sportovní klub. Konkrétně pro fotbalový klub SK Viktoria Sibřina. V nedávné době vedení klubu hledalo ideální řešení pro správu a evidenci účasti hráčů na zápasech, trénincích či dalších událostech. Řešení by samozřejmě mělo být mnohem komplexnější než jen zastřešovat evidenci účastí a omluvenek hráčů všech věkových kategorií. Správa dodatečných informací, plateb, dokumentů, profilů jednotlivých uživatelů a rozdělení členů do patřičných týmů v rámci klubu je jen menším výčtem z požadovaných funkcionalit. V práci je promítnuto i autorovo působení v klubu jakožto hráče, a tudíž i jeho zkušenosti s vnitřním fungováním a chodem klubu.

Nutností trenéra, který vede jakýkoliv sportovní trénink určité skupiny lidí, je dopředu jednoznačně vědět s kolika lidmi bude daný den pracovat, dle toho si může dopředu připravit různá cvičení pro konkrétní počet osob. Důraz je kladen hlavně na uživatelskou přívětivost systému a zajištění všech podstatných informací jeho uživatelům. IS by v ideálním případě měla většina uživatelů využívat nanejvýš pár minut denně. Jeho jednoduchost je tedy elementární vlastností, kterou takto zadaný IS musí splňovat.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Hlavním cílem této diplomové práce je analýza a návrh modelu informačního systému pro sportovní klub SK Viktoria Sibřina. Model informačního systému je primárně zaměřen na správu interních záležitostí v rámci klubu. Ve sportovním klubu je potřeba zajistit evidenci osob u různých událostí, typicky se jedná o evidenci účasti na zápasech, trénincích apod. Navrhovaný model by měl poskytnout podporu při správě dat převážně pro vedení klubu.

Cílem teoretické části práce je seznámit se s pojmy z oblastí tvorby informačních systémů, analýzy uživatelských požadavků, využívaných modelů softwarového inženýrství a návrhu uživatelského rozhraní. Na pojmy z těchto oblastí je následně odkazováno v praktické části této práce.

Prvním krokem praktické části práce je analýza požadavků na informační systém, které jsou definovány ze strany sportovního klubu. Následně je za pomoci modelovacího jazyka UML vytvořen návrh informačního systému, který vychází z provedené analýzy požadavků. V rámci návrhu je vytvořen model tříd, stavový model, model interakcí či diagram nasazení IS. V posledním kroku praktické části je dle vytvořených modelů navrženo uživatelské rozhraní modelovaného informačního systému pro zobrazení na mobilních zařízeních. Nejprve jsou vytvořeny drátěné modely a na jejich základě i funkční prototyp uživatelského rozhraní.

Na závěr je zhodnoceno, zda návrh informačního systému proběhl úspěšně a zda by po jeho teoretické budoucí implementaci mohl informační systém zadavateli pomoci s řízením chodu sportovního klubu. V posledním kroku je provedeno porovnání aktuálního řešení s navrženým modelem.

## 2.2 Metodika

Metodika k této diplomové práci vychází z nabytých poznatků ze studia odborných literárních a internetových zdrojů týkajících se problematiky tvorby informačních systémů. Jako první krok jsou shromážděny všechny požadavky, které by měl informační systém dle zadavatele ve výsledném řešení splňovat. Následně je provedena analýza těchto požadavků. Součástí této analýzy je primárně stanovení jejich důležitosti pro implementaci.

Z výsledků analýzy požadavků poté vychází samotný návrh informačního systému. Ten je realizován za pomoci modelovacího jazyka UML. S využitím technik softwarového inženýrství, konkrétně disciplíny datového modelování jsou sestaveny a navrženy jeho hlavní modely. Jako první je vytvořen model tříd a k němu i datový slovník. Následuje stavový model a na závěr model interakcí IS. Pro doplnění je vytvořen i diagram nasazení IS.

Na závěr je práce doplněna o návrh možného uživatelského rozhraní vytvářeného informačního systému. Návrh uživatelského rozhraní vyplývá z navrhovaných modelů. Nejprve jsou vytvořeny drátové modely a na jejich základě i funkční prototyp IS. Je vytvořena varianta drátěných modelů pro verzi zobrazení ve formě mobilní aplikace. Tvorba návrhu uživatelského rozhraní je realizována pomocí programu Axure.

# 3 Teoretická východiska

## 3.1 Softwarové inženýrství

Softwarové inženýrství je technická disciplína, která má za cíl produkci komplexních informačních systémů či jiného softwaru. Obsahem této disciplíny je vše od fáze analýzy a návrhu systému, přes tvorbu až po jeho údržbu a reingeneering. Disciplína softwarového inženýrství je stále velmi nová, i proto se neustále mění a rozšiřuje. Samotný pojem „softwarové inženýrství“ byl úplně poprvé definován na konferenci sponzorovanou NATO v roce 1968. Jejími účastníky bylo několik desítek odborníků z různých oblastí praxe nebo vzdělávání. Kromě samotné definice se zde zformulovaly i směry, kterými se výzkum tohoto oboru bude v budoucnu ubírat. První definice SI definovaná na této konferenci zněla:

*„Softwarové inženýrství je disciplína, která se zabývá zavedením a používáním řádných inženýrských principů do tvorby software tak, abychom dosáhli ekonomické tvorby softwaru, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích.“*

V průběhu 70. let pak vznikaly první aplikace umožňující interakci s uživatelem. Poté v 80. letech započal masivní rozvoj softwarového inženýrství, silný nástup tvorby a využívání metodik, objektově orientovaného přístupu, standardizace, rozvoj produktových řad, komponent, architektur a modelů. Softwarové inženýrství je ale nakonec až v roce 1997 definitivně uznáno jako obor s certifikací v USA.

Dnes můžeme definici softwarového inženýrství zjednodušit na:

*„Softwarové inženýrství je inženýrská disciplína zabývající se praktickými problémy vývoje rozsáhlých softwarových systémů“* (Vondrák, 2002)

Nebo dle oficinálního standardu IEEE 610.12 zní definice takto:

*„Softwarové inženýrství je aplikace systematického, disciplinovaného, kvantifikovatelného přístupu k vývoji, provozu a údržbě softwaru, tj. aplikace inženýrství na software. Také je to studium přístupů dle výše uvedeného.“*

Všechna terminologie využívaná v disciplíně softwarového inženýrství je definovaná jako standard IEEE 610.12. (IEEE Xplore, 2002)

Softwarové inženýrství kloubí dohromady několik disciplín. Inženýrství, vědu, či umění. Využívá rutinní inženýrské postupy, zahrnuje řešení úloh pomocí matematických disciplín nebo využívá aspekty běžně využívané při umění navrhování vzhledu. Základem softwarového inženýrství jsou oblasti jako správa požadavků, softwarový návrh, tvorba

softwaru, testování softwaru a údržba softwaru. Tyto oblasti jsou při tvorbě vymezeny jako jednotlivé fáze v rámci daného softwarového procesu. Každý takový proces musí být jednoznačně definován, vytvořen, revidován a po dobu svého produkčního cyklu i vylepšován. S jednotlivými oblastmi softwarového procesu pak úzce souvisí primárně pojem informační systém a dále pak životní cyklus tvorby softwaru, metodiky vývoje, analýza požadavků, návrh systému apod. Tyto pojmy jsou blíže vysvětleny v následujících kapitolách. (Arlow, 2007)

## 3.2 Informační systém

Informační systémy jsou již neodmyslitelně spjaté s fungováním dnešního světa, tak jak je nám znám. Jedná se o odvětví, které se dynamicky rozvíjí a zasahuje již skoro do všech činností lidského života. Klíčovou roli dnes sehrávají informační systémy primárně ve fungování organizací všech druhů a nejvýznamněji pak v byznysovém světě. (Bruckner, 2012)

Pojem informační systém nebo systém je možné vysvětlit mnoha definicemi. Svou definici systému vyslovil kdysi i Aristoteles. „*Mnohé složité věci jsou jako celek více než jen souhrn částí, ze kterých se skládají.“* (Bruckner, 2012)

Pojem infomační systém je pak odvozen z definice systému. Informační systém může být definován například takto: „*Informační systém lze chápat jako systém vzájemně propojených prostředků a procesů, které slouží k ukládání, zpracovávání a poskytování informací.*“ (Bruckner, 2012)

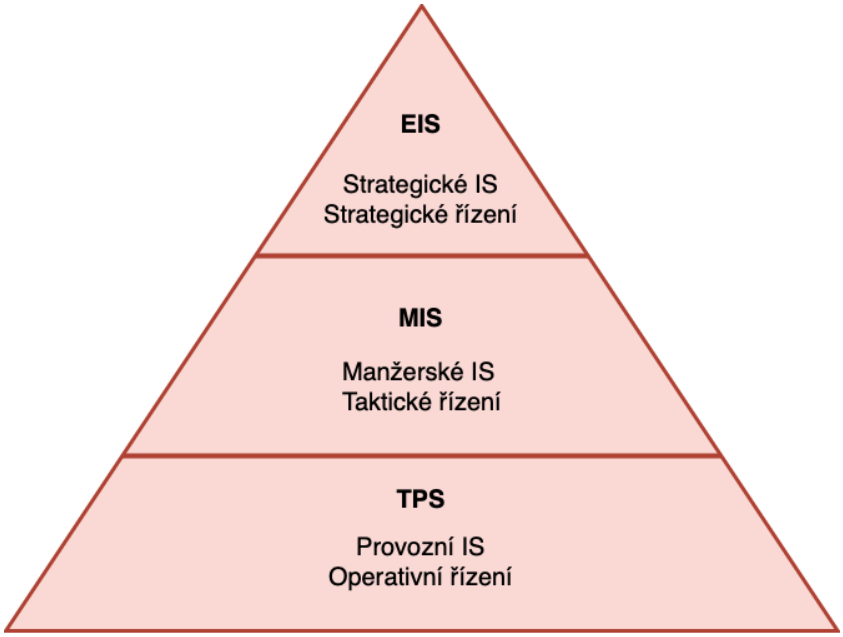
Odlišnou definicí informačního systému pak může být například tato: *„Informační systém organizace je systém informačních technologií, dat a lidí, jehož cílem je efektivní podpora informačních a rozhodovacích procesů na všech úrovních řízení organizace (firmy).“* (Procházka, 2009)

Všechny výše zmíněné definice jsou ale správné. Informační systém je nutné chápat v kontextu toho za jakým účelem je využíván. Informační systém je totiž nezbytnou součástí systému byznysového. Rozdílem těchto dvou systémů je jejich primární cíl, protože cílem informačního systému je zajištění správných informací na správném místě ve správný čas. Naopak cílem byznysového systému je realizace definovaných byznysových cílů a záměrů či zlepšení konkurenceschopnosti podniku na trhu. K jejich dosažení dnes podnikům napomáhají primárně právě informační systémy. (Bruckner, 2012)

Hlavním cílem informačního systému je tedy podpora jistých činností. Informační systém nemusí napomáhat pouze u automatizovaných činností, naopak je žádoucí, aby podporoval i činnosti neautomatizované, tedy ty činnosti, které mohou provádět i lidé. Volba optimálního poměru mezi těmito dvěma typy činností pak bývá nejzásadnějším problémem při návrhu informačních systémů. (Král, 1998)

### 3.2.1 Klasifikace informačních systémů dle úrovně řízení

Organizace jsou vždy rozděleny do několika řídících úrovní. Zpravidla se jedná o tři vrstvy řízení. Ty můžeme definovat jako vrstvy – strategickou, taktickou a operativní. Každá z těchto vrstev vyžaduje jiný druh dat. Navíc každá vrstva tyto data transformuje na informace podle svého zacílení. Na každé úrovni je potřebný jiný druh informací. (Vymětal, 2009)




**Obrázek č. 1 - Klasifikace IS podle úrovní řízení (Vymětal, 2009)**

a) **Strategické IS** – nejvyšší úroveň hierarchického modelu. Infomační systémy na této úrovni slouží k nalezení dlouhodobých trendů organizace za pomocí různých ukazatelů. Hlavním úkolem těchto IS je predikce změn. Tyto IS by také měli nabízet nejvyššímu vedení organizace informaci, zda může na tyto změny nějakým způsobem reagovat. Typické funkce EIS systémů jsou dlouhodobé plánování, ekonomická analýza hospodaření firmy, manažerské výkaznictví, rozbor na trhu apod. EIS se často využívají jako prezentační vrstva pro výstupy Business Intelligence. Data, která EIS využívají jsou brána z nižších vrstev TPS a MIS. EIS pro své správné fungování vyžadují podporu pro Business Intelligence (BI). BI zastřešuje termíny jako datový sklad, datová tržiště, systémy pro podporu rozhodování (DSS) a další. V těchto případech jsou využívány data z interních i externích datových zdrojů. (Vymětal, 2009)

b) **Manažerské IS** – prostřední vrstva taktického řízení podniku obsahuje informace, které jsou nutné k plnění administrativních úkolů a podpoře rozhodování na úrovni manažerů v organizaci. Nejedná se o dlouhodobé plánování, ale spíše krátkodobé či střednědobé. Převažují zde evidenční a analytické práce. Využívají se zde pravidelně plánované reporty založené na datech získaných z transakčních systémů, které jsou prostřednictvím MIS transformovány do podoby výstupních sestav obsahujících souhrny výsledků z požadované oblasti (obecně tzv. reporting). (Vymětal, 2009)
c) **Provozní IS** – jedná se o operativní vrstvu, která zajišťuje zpracování informací pro jednoduché a opakující se činnosti či úkoly. Tou mohou být například činnosti týkající se výroby, prodeje či logistiky. Má tedy za úkol podporu hlavních činností podniku. Důležitou vlastností informačních systémů na této úrovni je dodávat přesné, aktuální a jednoduše dostupné informace. (Vymětal, 2009)

### 3.2.2 Modely životního cyklu informačního systému

Tvorba informačních systémů se zpravidla odvíjí od tzv. životního cyklu informačního systému. Existuje několik modelů pro cyklus tvorby informačních sytémů. Tyto modely vznikly z praxí ověřených metodik, které si vývojové týmy s přibývající praxí osvojily.

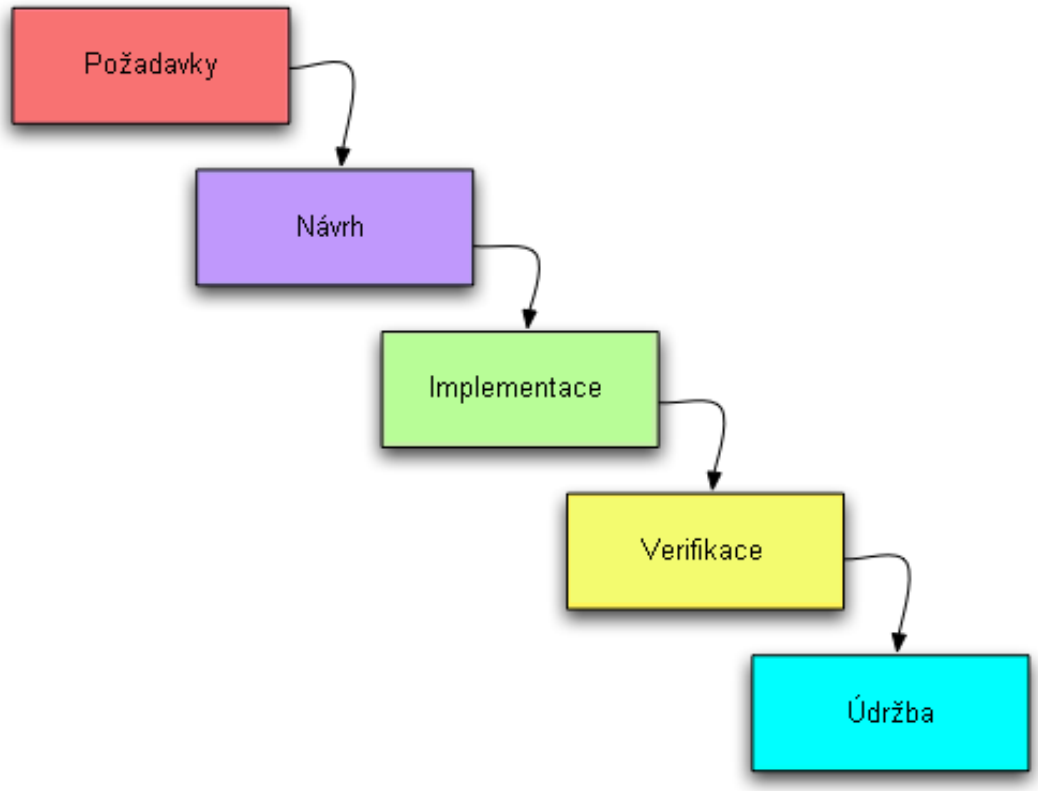
Tyto modely se vždy skládají z několika jednotlivých fází, které se mohou ale nemusí překrývat, opakovat nebo probíhat lineárně za sebou. To kdy jednotlivé fáze v průběhu tvorby nastávají je definováno právě modelem, který je pro vývoj softwaru zvolen. Často je v průběhu vývoje využito více typů modelů, to z toho důvodu, že každý model může být méně či více vhodný pro určité fáze. Stejně tak má každý model své klady i zápory, které je nutné respektovat. Jednotlivé fáze vývoje softwaru se dělí následovně: (Bruckner, 2012)

- Prvotní analýza, určení cílů, sběr požadavků.
- Analýza systému a sjednocení požadavků.
- Návrh systému.
- Implementace.
- Testování.
- Nasazení do produkčního prostředí.
- Zkušební provoz.
- Produkční období.
- Údržba, podpora a reengineering.

### a) Vodopádový model

Jedním z nejstarších a zároveň nejpoužívanějších modelů je model vodopádový. Vodopádový model vychází z myšlenky, že jednotlivé fáze navazují a nastávají lineárně po sobě. Po skončení jedné fáze nastává následující fáze a k předcházející se již vůbec nevrací. Nejdůležitějšími fázemi vývoje, na které je kladen velký důraz jsou prvotní fáze vývoje, protože co není ošetřeno či zachyceno v prvotních fázích v systému zůstává až do konce. Jedná se o primárně o fáze, požadavků, analýzy a návrhu.

Jednotlivé fáze v obecné definici tohoto modelu, jak následují po sobě, jsou: definování požadavků, návrh systému, tvorba systému a jeho následná implementace, testování spojené s verifikací a v neposlední řadě údržba systému. Dnes se již tento model nevyužívá tolik jako dříve. (Bruckner, 2012)




**Obrázek č. 2 - Vodopádový model schéma (Hoadley, 2008)**

Výhody:

+ Je jednoduchý a srozumitelný.
+ Jasně vymezené jednotlivé fáze.
+ Postup a výsledky jsou dobře zdokumentovány.
+ Vhodný pro malé projekty

Nevýhody:

- Nefunkční funkcionality jsou objeveny až na konci vývoje.
- Není vhodný pro již probíhající projekty.
- Nemůže vyhovět měnícím se požadavkům v čase.
- Dodání první verze je až po ukončení všech fází (Nelze odhalovat nedostatky ještě před dodáním celého systému).

### b) Iterativní model

Iterativní model přichází s naprosto odlišným přístupem k vývoji oproti vodopádovému modelu. V první fázi vývoje, která se nazývá iterací, se implementuje určitá malá podmnožinu všech požadavků. Tento proces – iterace se následně opakuje až do té doby, dokud nejsou implementovány všechny funkcionality a požadavky. Po první implementaci se ve fázi testování funkcionality otestují. Následuje vyhodnocení implementovaných funkcionalit v dané iteraci. Pokud klient nemá žádné připomínky k dané iteraci a vše funguje, jak bylo předem stanoveno, pokračuje se v další iteraci znovu se všemi jejími fázemi. Je možné říct, že je prováděna série malých vodopádů. (Bruckner, 2012)


**Obrázek č. 3 - Iterativní model schéma (TOPS Infosolutions Pvt Ltd, 2020)**

Jednotlivé fáze iterativního modelu začínají celkovým úvodním plánováním, poté se opakují v každé iteraci fáze:

- Plánování.
- Výběr požadavků.
- Analýza a návrh.
- Implementace.
- Testování.
- Vyhodnocení.

Pokud je daná iterace vyhodnocena jako úspěšná jsou funkcionality zakomponovány do výsledného softwaru či produkčního prostředí. Tím jsou změny nasazeny a pokračuje se v další iteraci, dokud není projekt dokončen. Dnes se jedná o hojně využívaný model, který je značně využíván v agilních metodikách. (Bruckner, 2012)

Výhody:

+ Vyhodnocení je získáno včas a pravidelně – snazší sledování postupu projektu.
+ Klient vidí výsledky s koncem každé iterace – může poskytovat zpětnou vazbu.
+ Vhodné pro velké a kritické projekty, protože častá iterace umožňuje snadné testování a ladění.
+ Je méně nákladné upravovat požadavky.

Nevýhody:

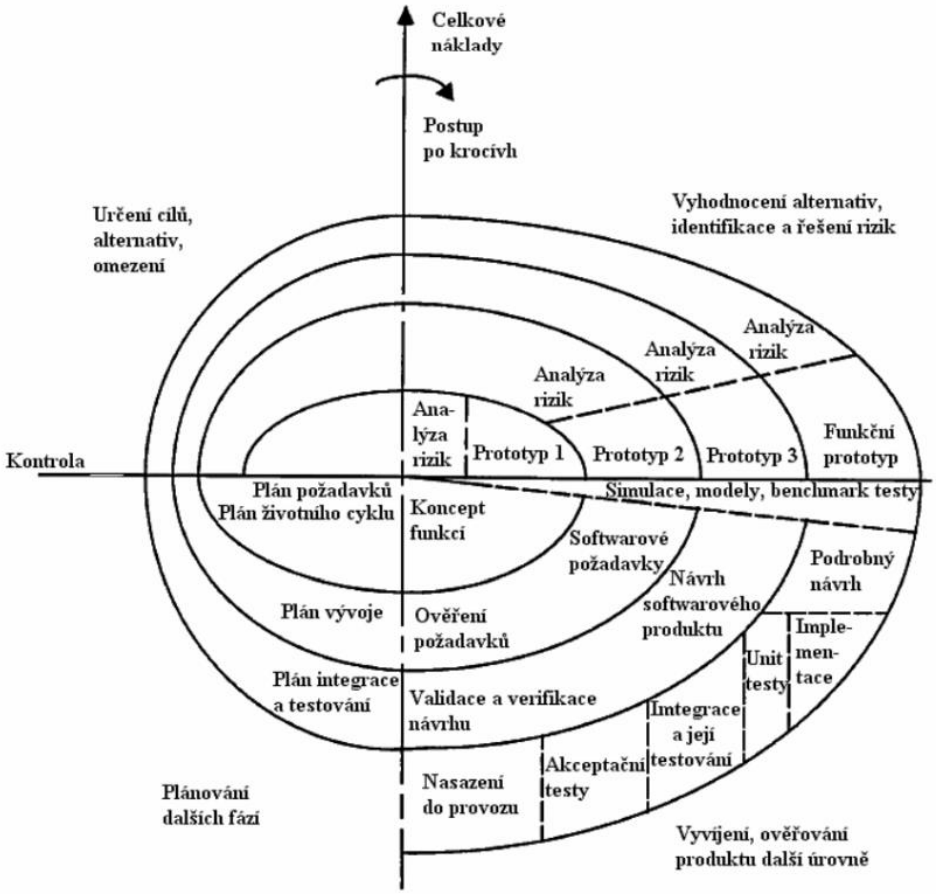
- Mohou nastat problémy s architekturou systému nebo návrhem, protože nejsou všechny požadavky shromážděny na začátku celého životního cyklu.
- Časově náročné.

### c) Spirálový model

Spirálový model je také jedním ze starších modelů. Byl navržen B.W. Boehmem v roce 1988. Kombinuje prototypový model s opakovanou analýzou rizik. Jednotlivé vývojové fáze se periodicky opakují. Nejdříve je implementováno samotné jádro systému. Následně se přidávají částí na vyšších úrovních. S opakujícími se iteracemi narůstá časová i nákladová náročnost. Vývoj je v jednotlivých krocích shodný s vodopádovým modelem. Každá iterace se pak skládá z následujících částí:

- Analýza – specifikace cílů.
- Vyhodnocení – vyhodnocení, identifikace a řešení rizik.
- Vývoj.
- Revize požadavků, validace (testování, zda prototyp pracuje, jak má).
- Plánování – plán pro příští iteraci.

Na konci každé iterace se provádí revize a předání dané iterace. (Molnár, 1992) (Doucek, 2004)




**Obrázek č. 4 - Spirálový model schéma (Boehm, 2007)**

Výhody:

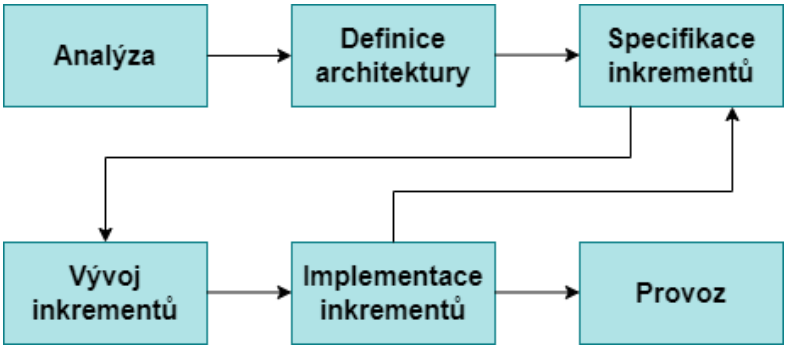
+ Model využívá ověřené kroky vývoje a analýzou rizik předchází chybám.
+ První verze systému je možné sledovat a hodnotit při jejich postupném vzniku.
+ Umožňuje konzultovat požadavky zákazníků již v jednotlivých krocích.

Nevýhody:

- Součinnost klienta je nutná po celou dobu projektu.
- Vyžaduje větší pracnost, proto je vhodný zejména pro vývoj velkých systémů.
- Vyžaduje odborné znalosti v oblasti posuzování rizik.

### d) Inkrementální model

Inkrementální model je velmi podobný modelu iterativnímu. Liší se především vymezením jednotlivých inkrementů neboli přírůstků. Je založen na principu, že je celý systém nejdříve specifikován na velmi nízké úrovni a následně je rozdělen na malé samostatně realizovatelné části, kterým se říká inkrementy. Tyto inkrementy pak prochází všemi fázemi vývoje. Jedná se v podstatě opět o malý vodopádový model, který je ale jasně vymezen pouze na určitou část systému. Nejdůležitější otázkou je definování jasných hranic jednotlivých inkrementů, od kterých se vše následně odvíjí. (Bruckner, 2012)




**Obrázek č. 5 - Inkrementální model schéma (Požár, 2010)**

Výhody:

+ Rychlá zpětná vazba klienta po realizaci každého inkrementu.
+ Snadná identifikace chyb.

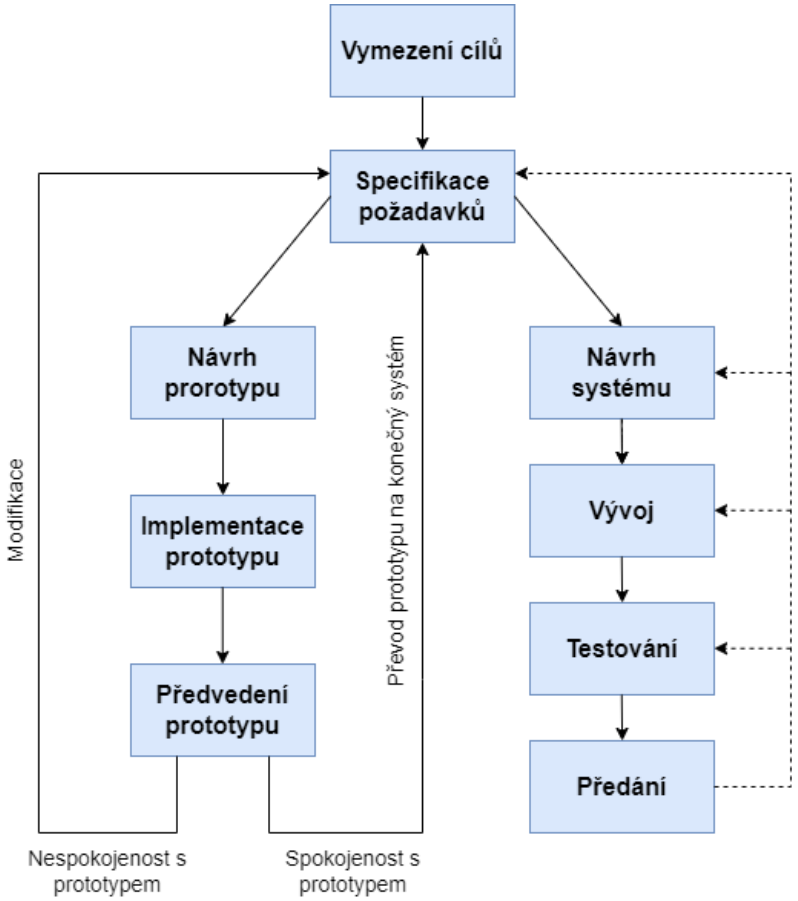
Nevýhody:

- Obtížnější sledování a kontrola postupu vývoje.
- Vyžaduje dobře vytvořený návrh plánu.
- Náprava chyby v jednom z inkrementů, vyžaduje opravu ve všech dalších inkrementech, což bývá velmi časově náročné.

### e) Prototypový model

V prototypovém modelu dochází k vývoji tzv. prototypů. Jedná se o ukázku softwaru, jak bude software vypadat po zapracování všech požadavků klienta. Cílem prototypového modelu je seznámit klienta, v co nejkratší době s prototypem (prvními verzemi) systému. To umožňuje rychlou reakci na výsledky, které mohou být jednoduše upraveny.

Poté co je klient seznámen s prototypem, může své předchozí požadavky upravovat či specifikovat. Klientovi nové požadavky jsou opět naimplementovány do prototypu a klient dostává novou verzi. Tyto iterace probíhají až do té doby, než je klient s řešením spokojen. Dochází tedy k opakovaným iteracím s uživateli pro úpravu finální vize. Po schválení následuje již realizace návrhu a následně implementace celého systému. (Bruckner, 2012)




**Obrázek č. 6 - Prototypový model schéma (Molnár, 1992)**

Výhody:

+ Snížení nebezpečí projektových rizik.
+ Zjednodušení možnosti změn v průběhu procesu vývoje.

+ Uživatel je zapojen v celém procesu vývoje – zvyšuje se tak kladné přijetí konečné implementace.

Nevýhody:

- Málokdy se podaří otestovat vlastnosti systému, které se projevují až při plném provozu.
- Z důvodu opakovaného schvalování prototypu s klientem může projekt začít stagnovat.

### f) Další modely

Mezi další modely může být zařazen například model RAD (Rapid Application Development), kde je snaha upozadit fázi plánování, kvůli rychlejšímu vývoji. Reaguje tak na rigidnost vodopádového modelu. Ke korekci požadavků a návrhu systému se využívají prototypy. Ovšem tento postup může být využit pouze při vývoji takového softwaru, který umožňuje rychlý vývoj. Výhodou je ale snazší řízení rizik či vyšší kvalita. Nevýhodou pak třeba nižší míra škálovatelnosti, proto se využívá spíše u menších či středně velkých projektů.

Dále je pak možné zmínit OOAD model (Objektově Orientovaná Analýza a Design), který se realizuje pomocí diagramů. Zde se jedná o popis systému za pomoci skupiny samostatných objektů, které jsou vzájemně propojeny. Zahrnuje doporučené postupy pro většinu aspektů tvorby softwaru. Počítá s neustálými změnami požadované klientem a zároveň jsou snadno udržovatelné a rozšiřitelné.

Modelů pro vývoj softwaru je dnes již mnoho. Každý má své výhody i nevýhody. Model je tak nutné vybírat až po úplném seznámení se s typem projektu, na který má být uplatněn. Každý model je tedy vhodný pro jiný typ tvorby softwaru za jasně daných okolností. (Bruckner, 2012)

## 3.2.3 Metodiky vývoje informačních systémů

S modely vývoje softwaru úzce souvisí pojem metodika vývoje. Ta rozšiřuje vybrané modely vývoje o postupy a přístupy, jak se budou realizovat jednotlivé fáze vývoje. Metodika je tedy souhrnem postupů, k jednotlivým fázím, vedoucích k dodání funkčního

softwaru. Zároveň by metodika měla odpovídat na otázky Proč? Kdo? Kdy? a Co? (Bruckner, 2012)

Hlavním cílem využívání metodik je snížení výskytu problémů při vývoji softwaru stanovením určitých pravidel. Metodiky se prvotně začali využívat v rámci projektového řízení a převážně až v 80. letech minulého století. Do dnešní doby se nejvíce prosadily dva hlavní směry – rigorózní a agilní metodiky. (Middleware.cz, 2015)

### a) Rigorózní metodiky

Rigorózní metodiky někdy nazývané jako klasické metodiky jsou historicky největší skupinou metodik. Rigorózní metodiky jsou tzv. "těžké" metodiky, to z toho důvodu že obsahují velké množství formalit, jsou podrobné a specifické direktivním řízením. Předpokládá se definování všech požadavků předem. Souvisí tedy především s vodopádovým modelem. Tyto metodiky se také velmi těžko dokáží vypořádat s neočekávanými změnami. Pracuje se zde s předpokladem neměnného zadání od zákazníka. Další charakteristikou může být také jejich objemná dokumentace. (Middleware.cz, 2015)

Z toho ovšem vyplývají i nedostatky těchto metodik. Do rigorózních metodik řadíme vodopádový model, který ale často bývá označován jako příklad toho, jak by se software vyvíjet naopak neměl. I přesto je však hojně používán. Hlavně díky jeho jednoduchosti. Dále s těmito metodikami spojujeme ještě model spirálový, Unified Process, Rational Unified Process (RUP) nebo Enterprise Unified Process (EUP). (Kadlec, 2004)

V důsledku uvedených nedostatků těchto metodik se hledaly nové možnosti přístupu k vývoji softwaru. Poptávající strana softwaru byla a je stále více nucena k častým změnám ve svých obchodních modelech již v průběhu vývoje softwaru. Na přelomu 20. a 21. století se tak z těchto důvodů začalo mluvit o agilních metodikách. (Middleware.cz, 2015)

### b) Agilní metodiky

V roce 2001 byl v Utahu sepsán *Manifest agilního programování*, který definoval přístup k vývoji softwaru dnes známý jako agilní programování. Autory manifestu jsou odborníci z oblasti softwarového inženýrství jako například Kent Beck, Ward Cunningham, či Martin Fowler. Od této chvíle se tedy začali dříve nazývané "odlehčené metodiky" nazývat agilními. (Principy stojící za Agilním Manifestem., 2001)

Agilní metodiky jsou metody, které jsou založené na iterativním a inkrementálním přístupu. Agilní znamená, že dáváme přednost reálným výsledkům před striktně zavedenými procesy. Agilní metody umožňují vývojářům rychlý a dynamický vývoj softwaru. Počítá se zde se změnou požadavků na funkcionality již v průběhu řešení. Naopak čas a zdroje jsou zde oproti klasickým metodikám neměnné. S klientem se aktivně komunikuje a konzultuje aktuální stav řešení v průběhu celého vývoje. Následně se přizpůsobují priority do dalšího vývoje dle zpětné vazby klienta. (Šochová, 2014)

Dle agilního manifestu jsou upřednostňovány tyto 4 priority vývoje informačních systémů: (Principy stojící za Agilním Manifestem., 2001)

- Lidé a jejich spolupráce, komunikace před procesy a nástroji.
- Funkční software před obsáhlou dokumentací.
- Spolupráce se zákazníkem před sjednáváním kontraktu.
- Rychlá reakce na změnu před přesným dodržováním plánu.

V manifestu je dále také definováno 12 základních principů agilního programování. (Chlapek, 2011)

Mezi známější příklady agilních metodik lze zařadit SCRUM, extrémní programování nebo metody Crystal Clear.

Tou nejznámější a zároveň nejpoužívanější je pravděpodobně metoda SCRUM, která je vhodná pro menší projekty s předpokladem často se měnících požadavků klienta. Vývoj je zde rozdělen do jednotlivých iterací, které jsou nazývány sprinty. (Kadlec, 2004)

U metod Crystal Clear je pak hlavní myšlenkou to, že je lepším řešením metodiku vytvořit a přizpůsobit potřebám konkrétního projektu jako zcela novou a unikátní. Podle této filozofie žádná metodika nemůže vyhovovat každému projektu.

Dalšími méně známými příklady pak jsou třeba Adaptive Software Development (ASD), Feature-Driven Development (FDD) nebo Lean development. (Middleware.cz, 2015)

Výčet typů metodik není úplný, což ale není ani žádoucí, jelikož se metodiky navzájem doplňují a každá z nich obsahuje mnoho společných prvků. Navíc jsou agilní metodiky stále velmi nové, ale za to velmi rychle se rozvíjející odvětví softwarového inženýrství. Jelikož se potřeby a požadavky klientů stále mění a rozvíjejí, tak je potřeba vyvíjet i samotné metodiky k jejich realizaci. To je cílem dnešních agilních metodik. (Šochová, 2014)

### c) Porovnání agilních a rigorózních metodik

Porovnání těchto dvou přístupů k vývoji softwaru, které vycházejí z naprosto odlišných předpokladů a odlišného pohledu, lze vidět na jednoduchém shrnutí hlavních rozdílů v následující tabulce.

| | **Porovnání rigorózních a agilních metodik** | |
|---|---|---|
| | Rigorózní metodiky | Agilní metodiky |
| **Předpoklady** | SW procesy lze popsat<br>Požadavky je možné definovat předem | SW procesy nelze popsat<br>Předem jen hrubé požadavky |
| **Obsah** | Přesně definované procesy, činnosti, artefakty | Jen generativní pravidla, praktiky a principy |
| **Použití** | Standardní projekty<br>Velké projekty | Výzkumné projekty<br>Time-to-market<br>Menší týmy |

**Tabulka č. 1 - Porovnání rigorózních a agilních metodik (Buchalcevová, 2005)**

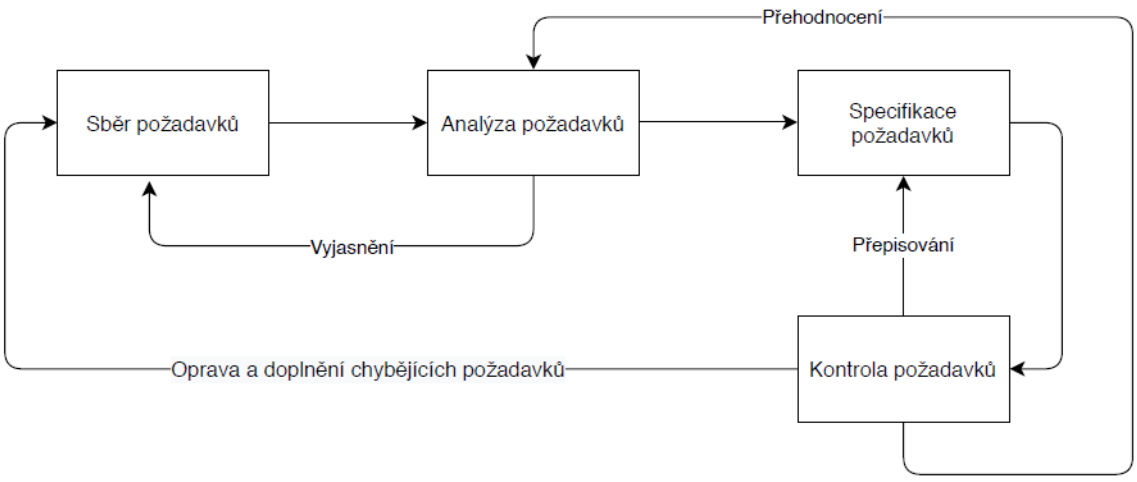
Oba přístupy mají své odpůrce i zastánce. Každý se navíc hodí k jinému typu projektu. Dnes již převažuje spíše agilní přístup k vývoji softwaru. Řešením problému, který z těchto přístupů by se měl k vývoji aplikovat, ale může být kombinace obou přístupů. Rigorózní metodiky je možné zjednodušit a aplikovat v rámci nich některé části z agilních přístupů a na druhé straně pak, pokud se agilní přístup využívá při vývoji větších projektů, je možné zde více formalizovat a dokumentovat. Ani jeden z přístupů tedy není špatný, ale každý přináší svá pozitiva a negativa. Ve většině případů se tak setkáme s vývojem, ve kterém se tyto přístupy prolínají. (Buchalcevová, 2005)

## 3.3 Analýza požadavků na informační systém

### 3.3.1 Zadavatel a jeho požadavky

Požadavky definují budoucí chování a vlastnosti vyvíjeného softwaru. Tyto požadavky vychází z představy zadavatele. Zadavatelem rozumíme osobu či společnost, která poptává zakázku vývoje či změny softwaru pro zefektivnění své činnosti. Zadavatel dodavateli musí sdělit svou představu budoucího řešení. Na základě těchto nároku jsou sestaveny požadavky na informační systém. V další fázi procesu analýzy požadavků se zadavatel stává schvalovatelem řešení, které dodavatel navrhne. Zadavatel je ve většině případů i zdrojem financí pro vývoj požadovaného softwaru. Vše by se tedy mělo odvíjet dle představy zadavatele.

Požadavky by vždy měly procházet transformačním procesem. V tomto procesu jsou všechny požadavky upřesněny pro vývojáře, kteří budou pracovat na vývoji softwaru, tak aby bylo vše jednoznačné. Analyzují se omezení systému, proveditelnost jednotlivých požadavků anebo se případně doplňují jejich detaily. Tento proces je popsán následujícím schématem.

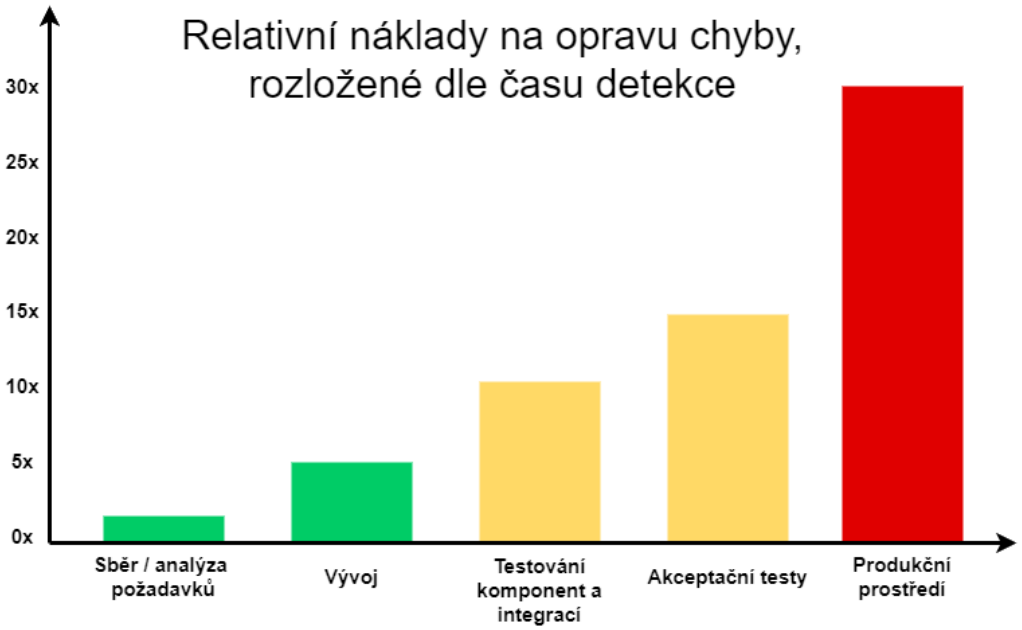



**Obrázek č. 7 - Schéma tvorby požadavků**

Tento proces sběru, kompletace a smysluplné interpretace požadavků na nový systém je klíčovou fází pro úspěšné dokončení projektu bez většího časového či finančního nárůstu. Na realizaci nového informačního systému ve většině případů pracuje více týmů či členů. V případě rozsáhlých systémů je tak nutná kooperace až několika týmů. Mezi ně můžeme zařadit například vývojový, analytický, testovací či manažerský tým. Z tohoto důvodu se na

tvorbě zadání podílí všechny zmíněné týmy. Všechny týmy musí mít přesně definované zadání již na začátku projektu. (Řepa, 2012)

Ucelené a správně navržené zadání je kritickou částí pro úspěšnost projektu. Pokud je zadání navrženo špatně, mohou se budoucí náklady na projekt až mnohonásobně zvýšit. S každou další fází vývoje projektu se cena opravy chyb exponenciálně zvyšuje. Je tedy žádoucí eliminovat maximum chyb a nejasností v co nejdřívější fázi projektu. (Deep Source, 2019)




**Obrázek č. 8 - Náklady na opravu chyby dle fáze projektu (Deep Source, 2019)**

### 3.3.2 Sběr požadavků

První fází vývoje softwaru a zároveň prvním krokem při tvorbě požadavků je tzv. sběr požadavků. Při sběru požadavků zadavatel popisuje svou představu budoucího systému. Sběr požadavků se často provádí v takzvaných iteracích. Ze začátku zadavatel definuje své prvotní vize, dodavatel s těmito požadavky projde celý cyklus tvorby požadavků a vrací se zpět k zadavateli, ale tentokrát se jde s požadavky více do hloubky. Zadavatel je obeznámen s aktuálním stavem a následně se tvoří více konkrétní požadavky. Takových iterací je možné provést několik do té doby, než jsou obě strany s výsledkem spokojeny a dokud nejsou vytvořeny přesně definované systémové a business požadavky. V praxi je důležité nepřehnat detail požadavků hned ze začátku. Často dokud neprojdou požadavky procesem analýzy,

není vůbec jisté, zda bude vůbec možné technicky požadavek splnit. Není žádoucí zadavatele zklamat něčím, co by později nebylo možné technicky do systému implementovat.

Každá iterace může být provedena za použití více různých technik sběru požadavků. Často je to dokonce i vhodné. Typicky používanými metodami pro sběr informací jsou: (Řepa, 2012)

- Dotazníkový průzkum.
- Interview.
- Focus groups.
- Uživatelské scénáře.
- Rozbor stávající dokumentace.
- Pozorování zavedených procesů.
- Aktivní zapojení do pracovního procesu.
- Studium legislativních omezení.
- Prototypování.
- Brainstorming.

Mezi dva nejvíce využívané postupy se řadí dotazníkový průzkum a interview.

**a) Dotazníkový průzkum**

Jedná se o kvantitativní metodu. Výhodou je, že dotazník může být prezentován širokému okruhu účastníků a jeho vyhodnocení není příliš časově náročné, i díky tomu že může být částečně automatizováno. Kvalitu výstupu dotazníkového průzkumu určují použité otázky. V praxi se nejčastěji využívají uzavřené otázky, které jsou doplněny o sadu otevřených otázek. Je také možné použití škály, která umožňuje respondentovi vyjádřit hodnocením dotazovaný předmět zájmu. Problémem u této metody může být špatná definice otázek. (Zvesela.cz, 2010)

**b) Interview**

Při tomto postupu se jedná o moderovaný strukturovaný rozhovor, který je veden s předem vybranou osobou. Otázky by měly být připravené tak, aby odpovídaly zaměření oblasti či role dotazované osoby ve vztahu k problematice. V praxi se interview hojně využívá v kombinaci právě s dotazníkovým průzkumem, který může být detailněji diskutován právě v rámci interview. Tímto způsobem lze odhalit nedorozumění

v nepochopení zadání a případné formální nedostatky otázek dotazníku. Interview je ale časově a administrativně náročnější metodou. Vyhodnocení je pak vždy individuální, ale poskytuje nejvíce interaktivní přístup k získávání informací. (Zvesela.cz, 2010)

### 3.3.3 Analýza požadavků

Analýza požadavků je proces, který slouží k detailnímu popisu požadavků zadavatele, jejich následné analýze a zdokumentování. Také je zřejmě nejdůležitější fází procesu tvorby požadavků. Hlavními cíli analýzy požadavků jsou: (Chlapek, 2011)

- Vymezení funkčnosti systému.
- Upřesnění požadavků tak, aby jim rozuměli všichni zúčastnění.
- Hledáni chyb a mezer v požadavcích.
- Analýza proveditelnosti.
- Prioritizace požadavků.
- Kategorizace požadavků.
- Odhad množství práce.
- Vyjasnění zadaní.
- Zachycení omezení.

V rámci analýzy požadavků je žádoucí stanovení priorit požadavků, to se většinou provádí na základě případů užití (UseCase). Užitečnou pomůckou při analýze je rozdělení požadavků do subsystémů. Požadavky v jednotlivých subsystémech jsou více přehledné a dodavatel se může věnovat nejdříve jedné části a až poté se přesunout k další. Díky tomuto rozdělení se lépe odhalují i ještě nepopsané požadavky. Problém mohou tvořit požadavky s nepřesným popisem cíle či nesdělení zásadní informace pro potřebu tvorby systému.

Výsledkem analýzy požadavků by měly být jasně definované business cíle, sestavení vzájemných souvislostí mezi požadavky a ucelená představa o vyvíjeném informačním systému. Pro finální přehled a schválení zadavatelem může být využit například interaktivní prototyp systému. Výstupem analýzy požadavků je pak tzv. katalog požadavků, který je velmi důležitou součástí, jelikož je využíván ve všech dalších etapách vývoje IS. (Chlapek, 2011)

### 3.3.4 Typy požadavků

Požadavky jsou východiskem pro tvorbu systémů a tvoří podklad pro jejich vývoj. Reprezentují to, co by měl daný systém dělat, avšak nikoliv to, jak by to měl systém dělat. Účelem požadavku je tedy popsat to, co by mělo být implementováno v rámci vývoje systému. Může se jednat například o popis chování systému, vlastnosti systému či omezení na systém. (Arlow, 2007)

Rozlišuje se několik typů požadavků. Tím základním rozdělením je rozdělení na požadavky:

a) Funkční
b) Nefunkční
c) Uživatelské
d) Podnikatelské

Je časté, že každý typ požadavků pochází z jiné fáze tvorby projektu. (Arlow, 2007)

a) **Funkční požadavky** jsou požadavky, které nám říkají, jak by se měl systém chovat a co by měl dělat. Příkladem funkčního požadavku může být třeba následující: (Arlow, 2007)
   - Kalkulačka sečte dvě zadaná čísla a výsledek vypíše na obrazovku.
   - Po stisknutí tlačítka On/Off se zapne/vypne počítač.

b) **Nefunkční požadavky** mají v českém jazyce poněkud nešťastný překlad z anglického non-functional. Nejedná se ale o požadavky, které by nefungovaly, ale o ty, které nezajišťují funkcionalitu. Do kategorie nefunkčních požadavků patří požadavky na kvalitu, použitelnost, efektivitu, spolehlivost, dodržování standardů, způsob dodání nebo jak budou řešena různá omezení systému. Mohou se zde i vyskytovat požadavky, které řeší, jak budou v systému implementovány funkční požadavky. Jinými slovy jsou to kvalitativní požadavky. Příkladem těchto požadavků může být třeba:
   - Platební brána systému bude akceptovat karty typu VISA/Mastercard.

c) **Uživatelské požadavky** jsou požadavky, které chce implementovat sám zadavatel. Na druhou stranu **podnikatelské požadavky** zahrnují právě všechny uživatelské požadavky a přidávají něco navíc. Tedy něco, co je potřeba implementovat vzhledem k přihlédnutí k podnikatelské činnosti zadavatele. (Arlow, 2007)

Požadavky se ještě mohou rozlišovat podle priorit. Priority požadavků určují, jak moc je nutné, aby byl požadavek implementován. Rozdělujeme je na tři kategorie:

a) **Normální** (Normal) někdy také kritické – požadavky, které jsou „samozřejmé“ neboli nutné.
b) **Očekávané** (Expected) – důležité požadavky, které by systém měl obsahovat, ovšem v krajním případě se bez nich dokážeme obejít. (např. snadnost používání).
c) **Užitečné** (Exciting) – požadavky, které jsou pro uživatele něčím navíc (v budoucnu se mohou změnit v kategorii Normal), nepovinné požadavky.

Některé další literatury nazývají jednotlivé úrovně priority požadavků trochu odlišněji, význam je ovšem stejný. Například u Arlowa a Neustadta lze nalézt názvy: (Arlow, 2007)

a) **Nezbytný** (Must have)
b) **Možný** (Should have)
c) **Eventuální** (Could have)

a navíc přidávají ještě čtvrtou úroveň, kterou lze označit požadavky, které mohou být zahrnuty až v dalších verzí systému při budoucím vývoji.

d) **Chceme mít** (Want to have)

Dle Wiegerse by pak měl každý požadavek splňovat tyto vlastnosti – úplnost, správnost, proveditelnost, jednoznačnost, prioritu a ověřitelnost. (Wiegers, 2008)

Pro ohodnocení priority požadavků v praktické části této práce byly využity úrovně priority definované Arlowem a Neustadtem. (Arlow, 2007)

## 3.4 UML (Unified Modeling Language)

Zkratka UML skrývá celý název Unified Modeling Language, česky pak unifikovaný modelovací jazyk. Důvodem k vytvoření UML bylo sjednocení dříve používaných metodik v jednu, která by je všechny sjednotila. UML notace je dnes již otevřeným standardem.

*„Sjednocený modelovací jazyk (UML) je druh grafické notace podporovaný nezávislým meta-modelem, který umožňuje popisovat a navrhovat softwarové systémy, konkrétně systémy budované využitím objektově orientované (OO) metodiky.“* (Fowler, 2009)

Jedná se o grafický jazyk či notaci pro vizualizaci, specifikaci, navrhování a dokumentaci systémů. UML využívá objektově orientovaný přístup. Naopak v rámci UML není definována žádná konkrétní metodika, jak by se měla notace používat. Výhodou UML je, že poskytuje dostatečnou úroveň abstrakce, a navíc je srozumitelné pro každého. UML je možné definovat jako nositele syntaxe pro vizuální tvorbu modelů. Standard UML je řízen sdružením Object Management Group (OMG). To vzniklo za účelem vytváření standardů, které mají podporovat vzájemnou slučitelnost objektově orientovaných systémů. (Fowler, 2009) (OMG, 2017)

### 3.4.1 Historie UML

Vývoj první verze UML se začal objevovat v roce 1994. UML začalo vznikat z několika soupeřících jazyků a metodik, které se řadily v té době k nejúspěšnějším. Těmi byly Booch (autorem Grady Booch), OMT (autorem James Rambaugh) a metodika Objectory (autorem Ivar Jacobson). Mezi roky 1994 – 1996 tak začal ve firmě Rational Software vznikat na základě těchto metodik i jazyk UML. Všichni tři autoři v té době významných metodik spolupracovali na vývoji UML. Jejich výsledkem byl první návrh UML a metodika RUP (Rational Unified Process). Jelikož veřejné společnosti viděli v UML potenciál, bylo vytvořeno konsorcium jazyka UML. V roce 1997 pak toto konsorcium zaslalo standardizační komisi OMG první návrh standardu UML.

V roce 1997 byl pak jazyk UML přijat sdružením OMG (Object Management Group) a jednalo se tak o první standard objektově orientovaného jazyka. Od prosince 2017 se používá zatím poslední publikovaná verze 2.5.1. (Arlow, 2007) (OMG, 2017)

Dnes je UML možno použít jednak jako notaci pro grafické modelování, tak i využívat jej i jako programovací jazyk. UML poskytuje možnost vývojáři nakreslit UML diagramy, ze kterých lze následně vygenerovat přímo spustitelný kód. Pro tuto funkcionalitu jsou ale

nutné specializované nástroje a velmi přesné a jasné vyjadřování v UML diagramech. Pojem Model Driven Architecture (MDA), což je jedním z dalších standardů skupiny OMG, se často pojí právě s možností převádění UML digramů na spustitelný kód. MDA pojem se snaží standardizovat použití UML jako použití programovacího jazyka.

### 3.4.2 Základní charakteristika a struktura UML

Cílem modelovacího jazyku UML je definování grafické notace k vyjádření analytických a návrhových modelů. Důležitým faktem je, že žádný UML diagram nemůže zachytit navrhovaný systém jako celek. Cílem je série různých pohledů na navrhovaný systém. To je realizováno pomocí několika spolu souvisejících diagramů a modelů, kdy každý zachycuje jiný pohled.

Strukturu UML je možné rozdělit mezi 3 hlavní části. Těmi jsou: (Fowler, 2009)

- Základní struktura (stavební bloky).
- Obecné mechanismy.
- Architektura.

### 3.4.3 Základní struktura

Do základní struktury UML někdy nazývané jako stavební bloky jsou zahrnuty 3 typy. Předměty (things), relace (relationships) a diagramy (diagrams).

**a) Předměty**

Předměty nebo abstrakce jsou základními prvky modelu. Předměty jsou tedy strukturní abstrakce, které jsou vyjádřeny podstatnými jmény v UML modelu. Jsou jimi například třídy, rozhraní, případ užití, komponenty atd. Chování předmětů se vyjadřuje slovesy. Sem patří stavy nebo interakce. Dále je možné v rámci předmětů využít poznámky, které se používají jako komentáře k popisu konkrétního prvku modelu. Poslední částí, která se týká předmětů je seskupení. Těmi jsou zpravidla například balíčky. Ty mají za cíl seskupit více významově souvisejících prvků do jednoho balíčku. (Ambler, 2005)

**b) Relace**

Relace definují vztah mezi dvěma či více objekty. Určují jejich vzájemnou souvislost. Existují 4 základní typy vztahů. Těmi jsou závislost, asociace, generalizace a realizace.

I. Závislost značí, že jsou dva prvky na sobě závislé. Fakticky se ale nejedná o asociaci. Používá se při modelování vztahu mezi třídami, balíčky apod. Grafickým značením prvního typu závislost je tečkovaná šipka. (Arlow, 2007) (Vrana, 2008)

II. Asociace popisuje vztah mezi jednou nebo více třídami. Tento vztah je abstrakcí množiny spojení mezi instancemi spojovaných tříd. Asociace mají definovaný název, role a násobnost. Dalšími typy jsou agregace a kompozice. (Arlow, 2007) (Vrana, 2008)

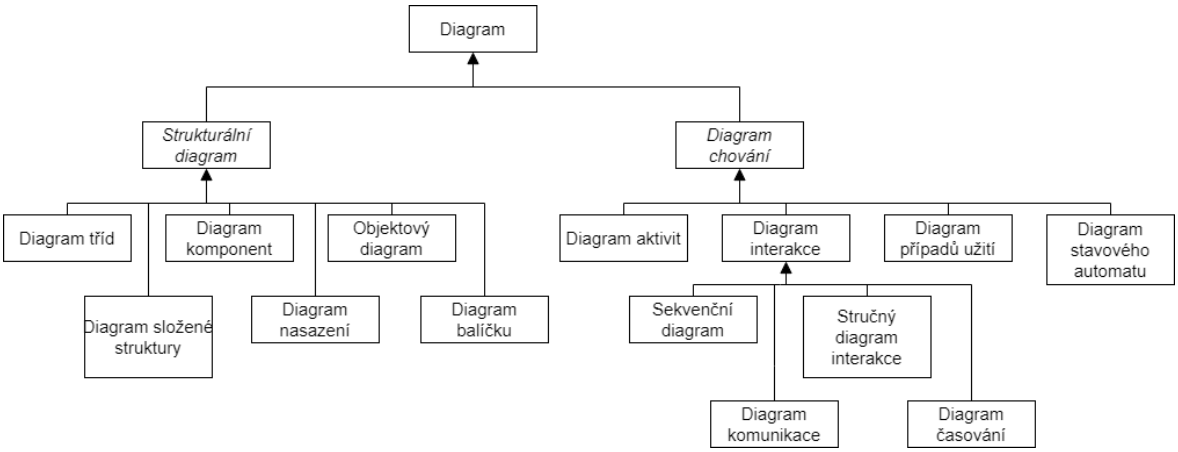
III. Generalizace neboli zobecnění je zásadním konceptem pro objektově orientovaný návrh či samotné objektové programování. Smyslem je, že podřízené třídy dědí ze svého předka jeho vlastnosti, a navíc mohou ke zděděným vlastnostem přidat své vlastní. Grafickým značením vztahu generalizace je plná čára s dutou šipkou směřující k nadřazené třídě. (Arlow, 2007) (Vrana, 2008)

IV. Realizace je vztah mezi dvěma klasifikátory, kdy první klasifikátor určí, co uskutečňuje druhý klasifikátor. Nejčastěji se realizace používá k definování vztahu mezi rozhraním a třídou. Realizace se graficky znázorňuje jako tečkovaná čára s šipkou směřující ke klasifikátoru. (Booch, 1999)

Detailněji jsou všechny typy vztahů vysvětleny v kapitole 3.5.1 - Model tříd.

**c) Diagramy**

Diagramy jsou grafickým znázorněním prvků modelu, jejich vztahů nebo chování. Graficky reprezentují systém z různých úhlů pohledu. Žádný z diagramů ale není modelem samotným, poskytují pouze neúplné pohledy na model. V UML je definováno několik různých typů diagramů. Ty lze rozdělit na 3 hlavní skupiny. Diagramy struktury (tzv. statický model), diagramy chování a diagramy interakce (tzv. dynamický model). Statický model zachycuje předměty a vztahy mezi nimi. Dynamický model pak zachycuje chování a vzájemné ovlivňování jednotlivých předmětů. Diagramy a jejich rozdělení je graficky znázorněno na následují stránce. (Arlow, 2007)




**Obrázek č. 9 - UML diagramy a jejich dělení**

I. Diagramy struktury:

- Diagram tříd.
- Objektový diagram.
- Diagram balíčků.
- Diagram komponent.
- Diagram nasazení.
- Diagram složené struktury.

II. Diagramy chování:

- Diagram aktivit.
- Diagram případů užití.
- Diagram stavového automatu.

III. Diagramy interakce:

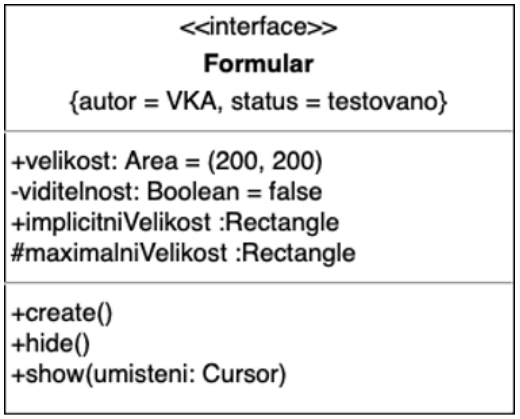
- Diagram komunikace.
- Diagram časování.
- Sekvenční diagram
- Stručný diagram interakce

### 3.4.4 Obecné mechanismy

V jazyce UML platí pro modelování 4 konzistentní mechanismy. Jedná se o specifikace, ornamenty, podskupiny a mechanismy rozšiřitelnosti.

a) **Specifikace** jsou textovým popisem prvků diagramu. Jsou základem modelu UML, který dává celému modelu smysl. (Arlow, 2007)

b) **Ornamenty** představují sadu jednoduchých symbolů, které vyjadřují vlastnosti prvků. Jejich použití přispívá ke zjednodušení komplikovaných struktur. Počet použitých ornamentů není nijak omezen a je možné je tedy přidávat do té doby, dokud nebude prvek obsahovat dostatečnou úroveň podrobností. Příkladem může být následující obrázek. Jsou zde použity symboly pro vlastnosti atributů (private (-), public (+), protected (#)) a další. (Ambler, 2005)




**Obrázek č. 10 - Třída obohacená o ornamenty**

c) **Podskupiny** reprezentují možnost dvou různých pohledů na řešenou problematiku. UML rozlišuje 2 typy podskupin: (Arlow, 2007)

I. Klasifikátory a instance – klasifikátorem je označeno abstraktní vyjádření typu předmětu, instance je pak konkrétním výskytem abstraktní představy. Příkladem může být obecný pojem auto. Instance poté vyjadřuje nějaké konkrétní auto.

II. Rozhraní a implementace – rozhraní určuje, čím se budou implementace řídit a implementace poté udává jednu konkrétní realizaci. Jinak řečeno implementace uvádí, jakým způsobem bylo rozhraní fyzicky vyřešeno. Příkladem rozhraní může být dálkový ovladač od televize. Implementace by

v tomto případě uváděla, jaký byl použit materiál, rozložení tlačítek anebo jednotlivých funkce tlačítek

d) **Mechanismy rozšiřitelnosti** představují možnosti rozšíření UML modelů. Tyto mechanismy se v UML nachází kvůli potenciálnímu rozšíření v budoucnosti. Do UML tak byly zahrnuty 3 jednoduché mechanismy, které umožňují budoucí rozšiřitelnost modelů. Omezení, stereotypy a označené hodnoty. (Arlow, 2007)

I. Omezení je podmínka nebo pravidlo, které se týká pouze jednoho prvku a musí být vyhodnoceno jako pravdivé. Omezující podmínka je v modelu graficky zobrazena jako text ve složených závorkách. (Arlow, 2007)

II. Stereotyp je nástroj, který umožňuje vytvářet nové prvky v modelu, které vycházejí z již existujících. Nové prvky mají stejné vlastnosti jako existující, ale používají se s jiným záměrem. Jejich grafickou reprezentací jsou dvojité lomené závorky umístěné nad názvem prvku. (Arlow, 2007)

III. Označené hodnoty umožňují stávající prvky modelu rozšířit o nové informace ve specifikaci prvku. Označené hodnoty jsou v modelu zobrazeny pod názvem prvku jako text ve složených závorkách. (Arlow, 2007)

### 3.4.5 Architektura

Architektury se využívají nejčastěji v případech, kdy se jedná o vývojově komplexnější systémy, tak aby bylo možné zachytit všechny jeho podstatné aspekty pomocí modelů či diagramů. Architekturu lze definovat jako několik různých úhlů pohledu na systém. Pokud se několik pohledů zkombinuje získáme architekturu.

## 3.5 Systémové modelování

Ve fázi návrhu informačního systému se využívají nástroje systémového modelování. Těmi jsou abstraktní modely navrhovaných systémů. Tvorba a použití těchto modelů se označuje za systémové modelování. V současné době se při tvorbě modelů v systémovém modelování téměř vždy využívá UML notace. Cílem systémového modelování je vytvořit návrh vytvářeného systému pomocí vybraných modelů. Modely dělíme na 3 základní typy: model tříd, stavový model a model interakcí. Tyto 3 modely popisují architekturu a chování vytvářeného systému. Všechny modely se doplňují a jsou vzájemně propojeny. Díky těmto modelům jsou zodpovězeny otázky, co má systém dělat, jak to má dělat a kdy to má dělat. (Vrana, 2008)

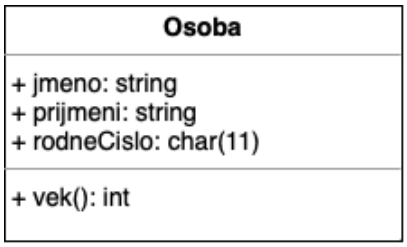
### 3.5.1 Model tříd

Pro zachycení statické struktury systému se využívá model tříd. Model tříd vyjadřuje strukturu systému pomocí objektů a jejich vztahů. Zobrazuje pouze statickou strukturu systému, jelikož neobsahuje časovou složku. Model tříd je zpravidla tím nejdůležitějším modelem, jelikož poskytuje intuitivní grafickou reprezentaci systému. Model tříd nám poskytuje zobrazení systému tak, aby bylo jednoduché nejdříve pochopit modelovanou realitu. Model ignoruje nepodstatné detaily, to je zásadní cestou ke zvládnutí složitosti systémů. Kromě objektů a jejich vztahů jsou v systému definovány i atributy a operace, které charakterizují každou třídu. Nejčastěji se pro modelování využívá diagram tříd. (Vrana, 2008)

**Diagram tříd**

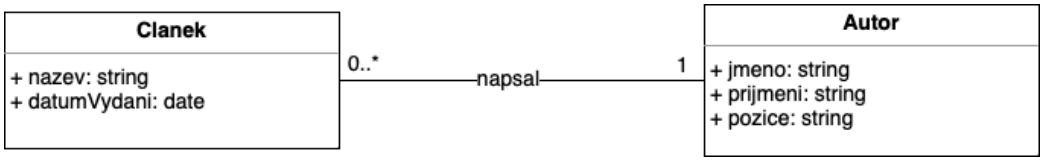
Reprezentací diagramu tříd je neorientovaný graf, který obsahuje uzly (třídy objektů) a hrany (vazby). Vychází z objektově orientovaného modelování, lze v diagramu tedy využít například abstraktní třídy, dědičnost či polymorfismus.

Obdélník je grafickou reprezentací jedné třídy objektů. Název je jménem třídy. Pod názvem jsou definovány všechny atributy se svým datovým typem, případně se zde uvádí další omezení u atributů. Jako je třeba maximální počet znaků či zda je atribut povinný apod. Pod atributy jsou uvedeny operace, které jsou definovány v dané třídě. Operace samozřejmě nejsou povinnou složkou třídy. U operací se definuje jako datový typ její návratový typ.




**Obrázek č. 11 - Příklad třídy Osoba v diagramu tříd**

Pro vytvoření vztahu mezi dvěma třídami se využívají vazby. Vazby mají vždy definovanou multiplicitu. Multiplicita se často vyjadřuje jako násobnost. Udává, kolik výskytů jedné třídy může mít vztah k jednomu výskytu přiřazené třídy. Multiplicita je tedy omezení rozsahu (uzavřený interval obohacený o 0). Nezaměňovat násobnost za kardinalitu, ta označuje počet instancí objektů jedné třídy v souboru. V notaci UML se multiplicita značí vždy na konci vazby (čáry). Pro označení nepovinného anebo žádného výskytu ve vazbě se využívá nula. Symbolem hvězdičky se poté označuje libovolný počet výskytů. Poslední možností je vypsání konkrétní hodnoty číslem. Nejčastěji se využívá jednička. V následujícím příkladu vazby jsou vazby definovány takto: Každý autor napsal žádný nebo libovolný počet článků, a naopak článek je vždy napsán právě jedním autorem. (Vrana, 2008)

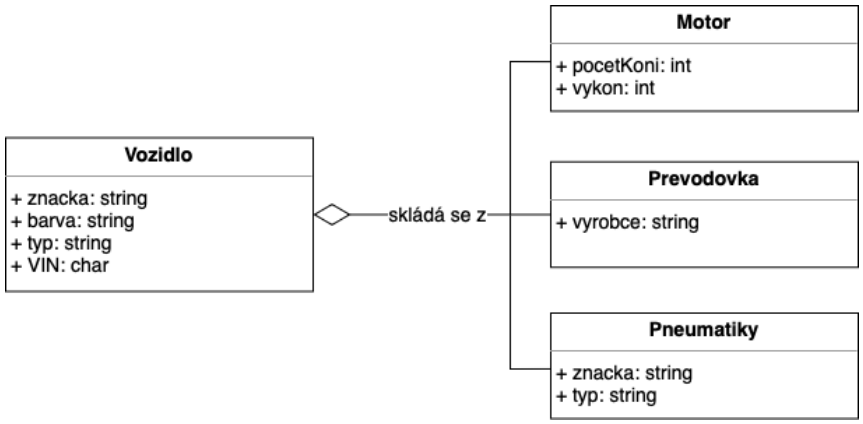



**Obrázek č. 12 - Příklad vazby Článek, Autor diagram tříd**

Dalším parametrem, který definuje vazbu mezi objekty, je její druh. Rozlišujeme druhy asociace, agregace, kompozice a generalizace (zobecnění). Druh vazby se v diagramu vždy rozlišuje graficky symbolem v místě spojení uzlu a hrany.

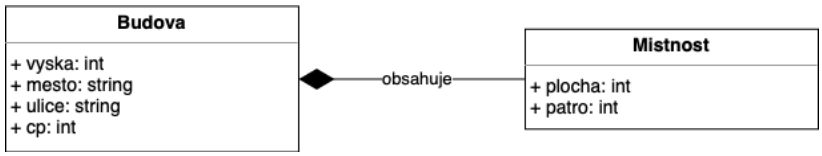
Základním vztahem, který je graficky značen pouze jednoduchou čárou, je asociace. Příkladem asociace je i výše zobrazený vztah mezi článkem a jeho autorem. Instance tříd mají v tomto případě mezi sebou přímou vazbu, která je definována výše vysvětlenou násobností. Vazbu je možné rozšířit o její název nebo pojmenování rolí. (Vrana, 2008)

Dalším typem vazby je agregace. Ta reprezentuje vztah součást – celek. Definuje vztah celku a jeho součástí. Může být typu „skládá se z“ či obráceně „je součástí“. Jeden objekt tak může být vytvořen z více částí. Součást celku ale může existovat i bez existence celku. Graficky je reprezentována pomocí prázdného diamantu. Šíření vlastností platí pouze z celku na součást, nikoliv obráceným směrem. Příkladem může být následující vztah. (Vrana, 2008)



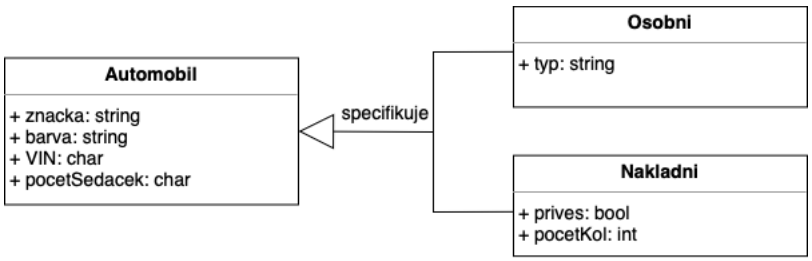

**Obrázek č. 13 - Příklad agregace diagram tříd**

Dalším typem vazby je kompozice, ta představuje silnější formu agregace. Kompozice je na rozdíl od agregace rozšířena o dvě další omezení. První z nich je, že součást může patřit pouze jednomu celku a druhé omezení zní, že zánikem celku zaniká i součást. Graficky ji pak reprezentuje plný diamant. Příkladem pak může být například vztah mezi budovou a jejími místnostmi. Budova obsahuje místnosti, ale místnost nemůže samostatně existovat bez budovy. (Vrana, 2008)




**Obrázek č. 14 - Příklad kompozice diagram tříd**

Posledním typem vazby je generalizace. Ta reprezentuje dědičnost mezi třídami (zobecnění). Jedná se o vztah rodič (nadtřída) - potomek (podtřída). Jedná se o hierarchický vztah, kdy podtřída dědí atributy, operace a vazby ze své nadřazené třídy. Podtřída nejenom dědí, ale navíc i přidává své vlastní specifické atributy a operace. Graficky se znázorňuje pomocí prázdného trojúhelníku. Příkladem generalizace může být například vztah mezi automobilem a jeho specifickými typy. (Vrana, 2008)




**Obrázek č. 15 - Příklad generalizace diagram tříd**

### 3.5.2 Model stavů

Model stavů je tzv. dynamickým modelem. To z toho důvodu, že popisuje chování systému v čase. Cílem modelů stavů je popsání reakcí systému na určité události. Ať už se jedná o vstupy z okolního prostředí či reakce na chování objektů uvnitř systému. V rámci modelů stavů definujeme také reakce systému na určité konkrétní vstupy. Pro grafickou reprezentaci se využívají stavové diagramy, které jsou vytvářeny pro každou třídu zvlášť. Není nutné vytvářet stavové diagramy u tříd, které nemají žádné dynamické chování. Pro takové třídy to není smysluplné. (LANO, 2009)

#### Stavový diagram

Stavový diagram popisuje vnitřní chování jedné konkrétní třídy objektů nebo části systému. Pro každou třídu může vždy existovat jeden nebo více stavových diagramů. Jsou ale i třídy, ke kterým stavový diagram vytvořen být nemusí, jelikož nemusí nabývat žádných stavů nebo není potřebné jej k pochopení modelu vůbec vytvářet.

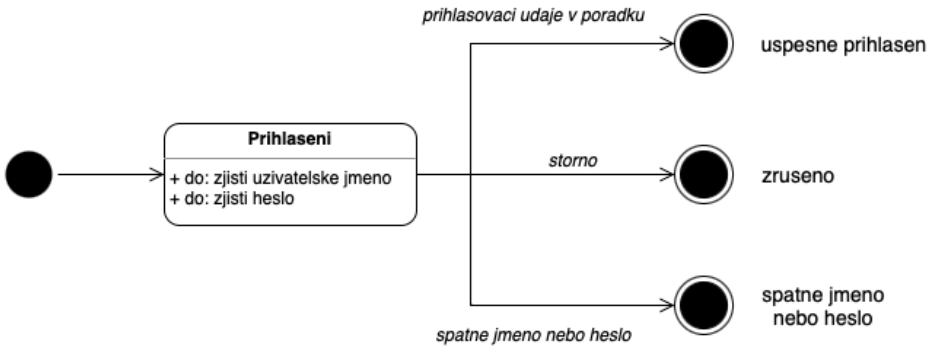
Každý stavový diagram má vždy jasně definovaný začátek a jeden nebo více konců. Začátkem lze rozumět výchozí stav objektu. Vlivem podnětů se následně stav objektu přesouvá do jiných definovaných stavů. Všechny stavy, do kterých se může objekt dostat

musí být definovány v rámci jeho diagramu stavů. Reakce objektu na určitou událost se odvíjí od stavu, ve kterém se zrovna daný objekt nachází. (Vrana, 2008)

Mezi základní prvky stavového diagramu patří stavy, události a přechody mezi stavy. Stav reprezentuje daný objekt a jeho soubor hodnot atributů v daném okamžiku. Událost je podnět od jednoho objektu k druhému, vyvolává tedy změnu stavu objektu. Přechodem mezi stavy se rozumí změna stavu objektu a jeho atributů. Tedy cesta mezi aktuálním a následujícím stavem.

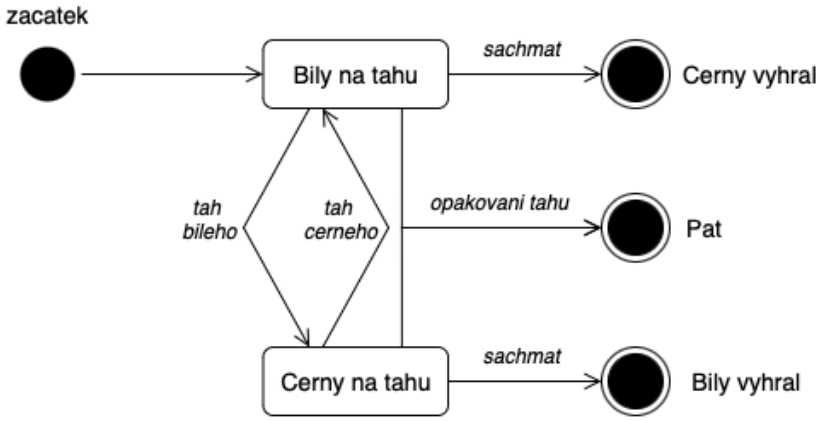
Doplňujícím prvkem stavového diagramu mohou být aktivity, které jsou odezvou objektu na odpovídající stav a událost. Aktivity jsou prováděny na přechodu nebo na začátku, konci či uvnitř stavu. Aktivity popisují chování a reakci objektu jako odezvu na změnu stavu.

Jednoduchý příklad s aktivitami typu *do* je například jednostavový diagram přihlášení do systému, který může skončit 3 konci – úspěšným přihlášením, zrušením nebo zadáním špatných údajů. (LANO, 2009)




**Obrázek č. 16 - Příklad přihlášení stavový diagramu**

Dalším jednoduchým příkladem stavového diagramu může být například průchod šachové hry. V tomto případě se hra může nacházet pouze ve stavu, že je bílý na tahu nebo že.je černý na tahu. Událostmi pro přechod mezi těmito stavy pak je tažení jednoho či druhého. Pokud již hráč nemůže táhnout, dostává šachmat a hra končí. Koncovými stavy pak je buď výhra jednoho nebo druhého či případně pat. (Vrana, 2008)




**Obrázek č. 17 - Příklad šachové hry stavový diagram (Vrana, 2008)**

### 3.5.3 Model interakcí

Model interakcí zachycuje jak interakce systémové, tak interakce s uživatelem. Model interakcí uvádí, jak na sebe jednotlivé objekty navzájem působí. Je to pohled na mnoho objektů najednou. Oproti modelu stavů, který popisuje chování objektů pouze jednotlivě. K úplnému popisu chování systému je potřeba jak stavový model, tak model interakcí.

Model interakcí popisuje systém na různých úrovních abstrakce. Na nejvyšší úrovni popisuje interakce systému s okolím pomocí UseCase diagramu (diagram případů užití). Podrobnější pohled je poté zobrazen sekvenčním diagramem, který popisuje výměnu zpráv mezi určitým množstvím objektů v čase. Záleží, o jaký UseCase se právě jedná, jelikož nemusí být zapojeny všechny objekty systému. Nejdetailněji pak popisuje chování systému diagram aktivit. Ten zobrazuje tok řízení mezi jednotlivými výpočetními kroky.

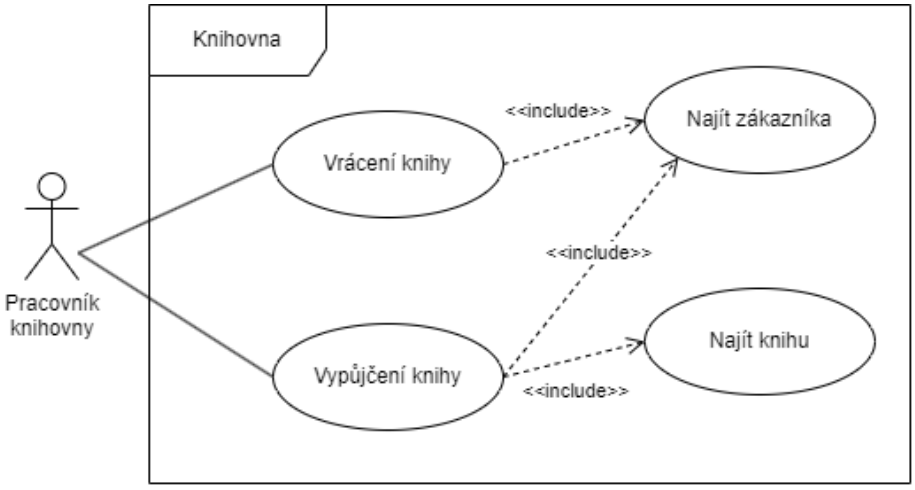
#### a) Diagram případů užití

Pro zobrazení chování systému z pohledu uživatele se využívá diagram případů užití (UseCase diagram). V rámci digramu jsou zachyceni všichni uživatelé a jejich případy užití systému, které jsou s každým uživatelem spojeny při jejich využití systému. Uživatelé, kteří jakkoliv vstupují do procesu jsou nazývání aktéři (actors). Mezi aktérem a jeho UseCasem existuje vztah, který je graficky znázorněn jako čára. Tento vztah reprezentuje tok informací. Vztahy mohou mezi sebou mít i dílčí UseCasy. (Arlow, 2007)

V diagramu případů užití lze rozlišit 3 typy vztahů – zahrnutí, rozšíření a generalizaci. První typ zahrnutí, označován anglickým slovem *include,* reprezentuje vztah povinnosti. V případě užití prvního případu je nutně využit i případ, který je označen klíčovým slovem

*include*. Zahrnutí se graficky znázorňuje přerušovanou čarou s textovým označením *include* a směřováním k případu podřízenému. Typ rozšíření je označován anglickým slovem *extend* a představuje vztah rozšiřitelnosti, kdy jeden případ užití může, ale nemusí zahrnovat i případ užití druhý. Rozšíření je reprezentováno přerušovanou čarou s označením *extend* a šipkou mířící k nadřazenému případu užití. Poslední generalizace se dělí na dva typy. První se využívá ke znázornění dědičnosti mezi aktéry a druhý typ označuje speciální vazby mezi jednotlivými případy užití. (Arlow, 2007)

Příklad níže zobrazuje jednoduchý UseCase digram systému v knihovně. Pracovník knihovny provádí pouze 2 činnost (2 případy užití) - půjčení a vrácení knihy. Vypůjčení knihy zahrnuje UseCasy vyhledání zákazníka v systému a vyhledání půjčované knihy v systému. Druhým případem užití je pak vrácení knihy, kde je nutné pouze vyhledat zákazníka a jeho aktuální výpůjčky.

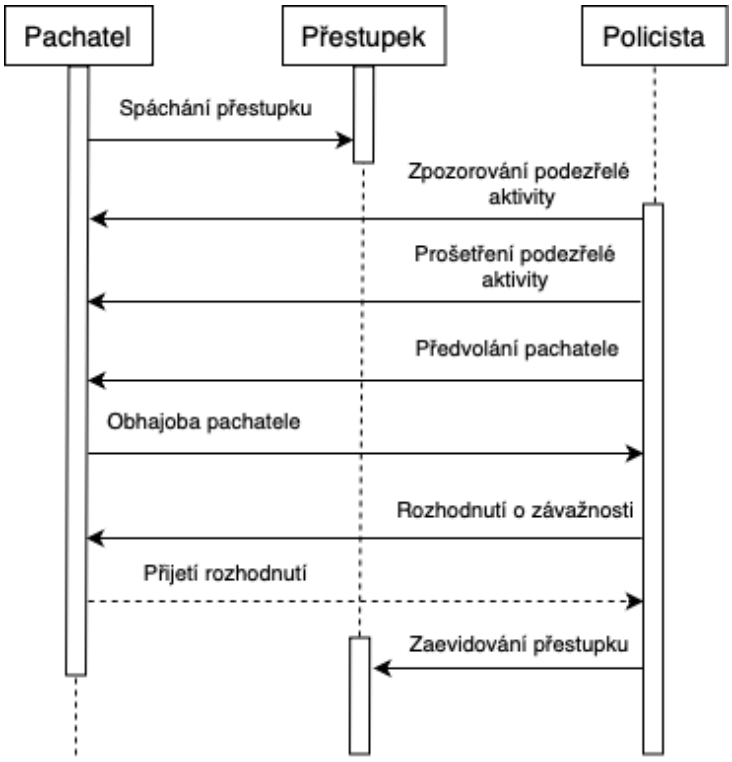



**Obrázek č. 18 - Příklad knihovna UseCase diagram**

**b) Sekvenční diagram**

Sekvenční model je složen ze scénářů a sekvenčních diagramů. Je vhodný pro popis posloupnosti chování z pohledu uživatele systému. V rámci sekvenčního diagramu se definuje posloupnost činností mezi jednotlivými objekty či aktéry. Diagram zobrazuje jednotlivé kroky, které systém provede v reakci na akci uživatele. Zobrazuje veškerou komunikaci mezi aktéry či objekty v rámci daného procesu. Sekvenční diagramy jsou spojeny se scénáři jednotlivých případů užití. Scénářem je myšlena strukturovaná posloupnost událostí, která má jasnou definici rolí, alternativních scénářů a vstupních a výstupních podmínek. Scénář je ve většině případů zadán textově. Ke každému UseCase je přidružen vždy minimálně jeden scénář. Pořadí zpráv mezi aktéry v sekvenčním diagramu postupuje vždy od shora směrem dolů. Tím vyjadřujeme pořadí zpráv.

Příkladem sekvenčního diagramu je například následující diagram, který popisuje posloupnost dílčích činností při spáchání přestupku.

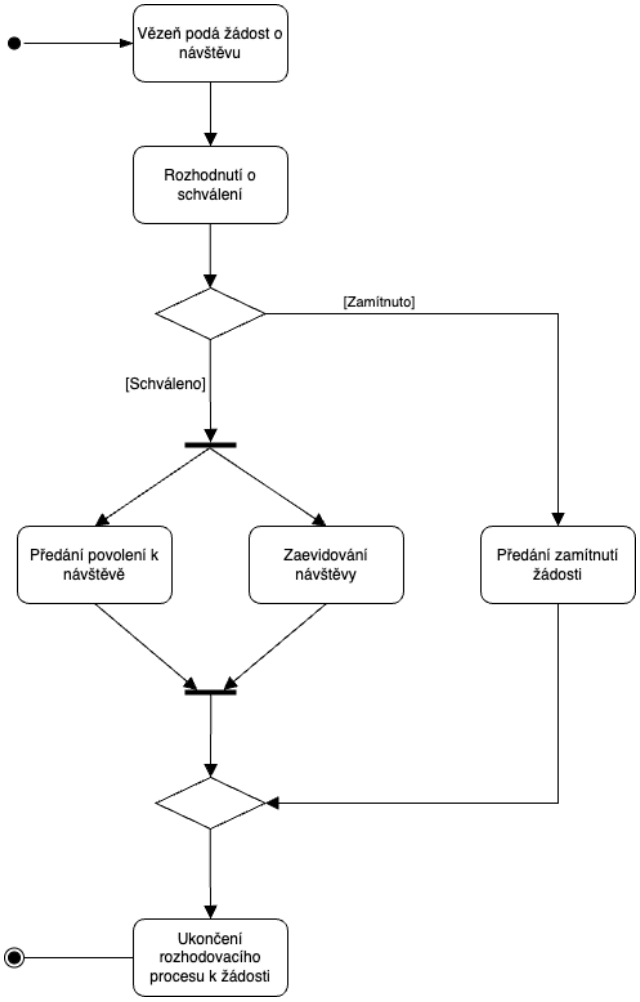



**Obrázek č. 19 - Příklad spáchání přestupku sekvenční diagram**

### c) Diagram aktivit

Nejnižší (nejdetailnější) úroveň zobrazení aktivit v modelu interakcí je zobrazována pomocí diagramu aktivit. Jeho účelem je zejména zachycení lineárních a paralelních toků řízení. Diagram aktivit vždy začíná černým kruhem a končí kruhem s opsanou kružnicí (stejně jako v případě stavového diagramu). Diamant vždy značí rozhodovací podmínku a větví běh procesu. Tlustá čára pak značí paralelizaci nebo sjednocení procesů. Notace je velmi podobná té u vývojových diagramů. Často se diagramy aktivit používají pro zobrazení složitějších procesů například algoritmů. (LANO, 2009)

Výhodou diagramů aktivit je, že mohou zahrnovat i zobrazení pro více jednotlivých případů užití současně. Mohou zachytit detailně posloupnost všech dílčích kroků i pro konkurenční toky a také pro případné alternativní scénáře. Příklad diagramu aktivit na následujícím obrázku zobrazuje rozhodovací proces žádosti vězně o schválení návštěvy ve věznici. V příkladu je využita podmínka i paralelně běžící procesy.

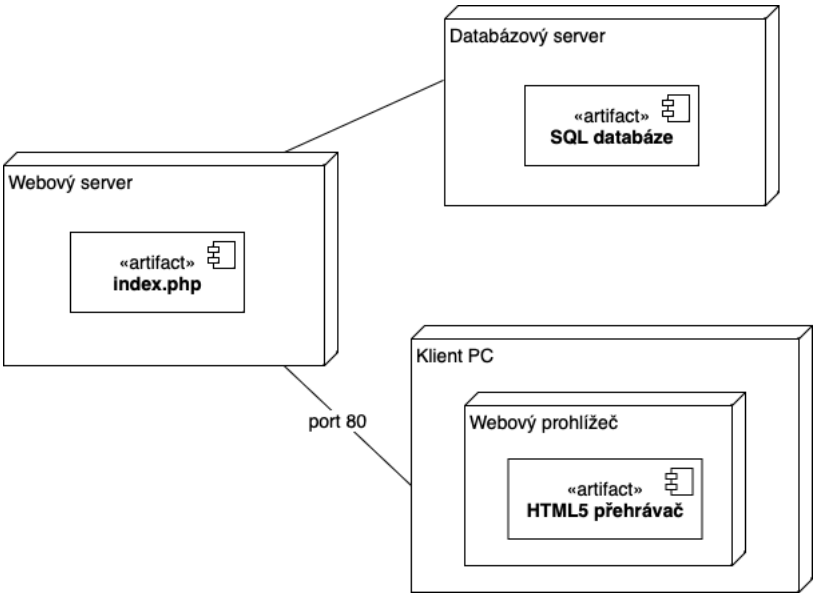



**Obrázek č. 20 - Příklad vězeňská návštěva diagram aktivit**

### 3.5.4 Diagram nasazení

Jedná se o typ statického diagramu. Diagram nasazení zobrazuje softwarovou architekturu na fyzické architektuře, na níž bude v budoucnu systém implementován. Ve fázi návrhu se vytváří diagram nasazení pro ucelenou představu, jak bude systém realizován na konkrétních částech hardwaru.

Diagram nasazení zobrazuje uzly a jejich komponenty a vztahy mezi nimi. Uzly reprezentují typ hardwaru, na kterých bude systém nasazen a komponenty reprezentují typ softwaru, který je v daných uzlech využit. Vztahy pak mohou popisovat protokoly či porty, na kterých probíhá vzájemná komunikace. (Arlow, 2007) (Microsoft, 2021)




**Obrázek č. 21 - Příklad video přehrávač v prohlížeči diagram nasazení**

## 3.6 Uživatelské rozhraní (UI)

Uživatelské rozhraní v překladu User Interface, ze kterého vychází používaná zkratka UI, je souhrnem způsobů, jakým uživatel ovlivňuje chování systému, strojů či softwaru. Jeho cílem je umožnit efektivní provoz systému, zatímco systém současně poskytuje zpětné informace uživateli, které mu napomáhají v procesu rozhodování. Hlavním cílem UI tedy je vytvořit uživatelské rozhraní, které umožňuje snadné, efektivní a příjemné neboli uživatelsky přívětivé obsluhování způsobem, který poskytuje požadovaný výsledek (tj. maximální použitelnost).

Nejběžnějším typem uživatelského rozhraní je dnes u softwaru obecně GUI – Grafické uživatelské rozhraní. To umožňuje uživateli ovládat software pomocí interaktivních grafických ovládacích prvků. Grafické prvky jsou zobrazovány na displeji v podobě menu, ikon či dalších typů grafických prvků. K interakci s jednotlivými grafickými prvky se využívá klávesnice a myš nebo dotykový displej. GUI vzniklo jako potřeba pro nahrazení rozhraní příkazového řádku (CLI). CLI bylo nutné nahradit něčím čemu by všichni uživatelé rozuměli jednoduše a intuitivně bez zdlouhavého a náročného učení se příkazům.

Samotná fáze návrhu uživatelského rozhraní je velmi důležitá primárně právě pro grafické znázornění zapracovaných změn v systému. Využívá se tedy jako prezentační nástroj pro zadavatele, který tak může velmi jednoduše posoudit, zda mu aktuální verze systému vyhovuje či nikoliv bez komplexnějších znalostí. Následně tak může řešení schválit či pozměnit jednotlivé komponenty o svou představu. Tato fáze bezprostředně předchází implementaci změn do systému.

### 3.6.1 Základní pravidla pro tvorbu UI

Při tvorbě uživatelských rozhraní je důležité držet se několika pravidel, která jsou základními pilíři pro jejich tvorbu. Jedná se o tzv. vzory (patterns), které jsou již odzkoušené a spolehlivě fungují v interakci člověka se softwarem. Jedná se například o principy rozvržení grafických prvků, se kterými dokáže uživatel jednoduše a intuitivně pracovat v rámci softwaru. Většina z těchto vzorů pro tvorbu UI je společností natolik zažita, že jakákoli jejich změna či úprava udělá uživateli práci se softwarem spíše méně komfortní. Některé z nich jsou důležitější, a proto by měly být dodržovány ve většině návrhů UI. Ve výsledku, ale samozřejmě vždy záleží na samotném designérovi nebo zaměření a typu vytvářeného softwaru či v poslední řadě požadavcích zadavatele. (Lidwell, 2011)

### a) Návrhový vzor – diagonální rozvržení prvků

Mezi nejdůležitější používané vzory při tvorbě UI patří rozvržení prvků. Zde se téměř vždy dodržuje pravidlo diagonálního vyvážení. Zde platí, že mozek typického uživatele vnímá obsah na stránce od levého horního rohu směrem ke spodnímu pravému rohu, a proto by nejdůležitější či počáteční prvky měly být umístěny v horním levém rohu, a naopak prvky koncové (například tlačítko potvrzení) by měly být v pravém spodním rohu. Výjimku tvoří arabské země, kde z důvodu jejich psaní písma zprava doleva, je vše převrácené s vertikální osou. (Lidwell, 2011)

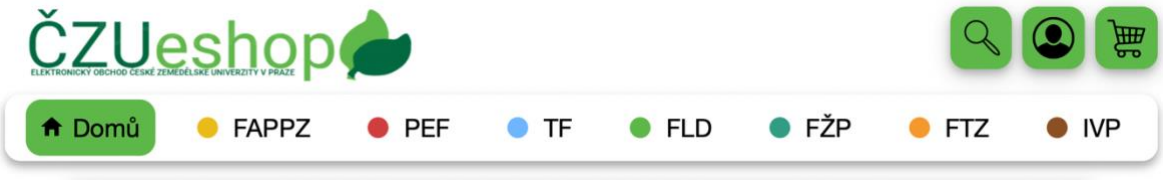

**Obrázek č. 22 - UI návrhový vzor diagonálního vyvážení prvků**

### b) Návrhový vzor – prvky v navigaci

Dalším důležitým a používaným vzorem je organizace a správné pojmenovávání položek v navigaci. Zde vždy platí, že by měl být obsah rozdělen podle příslušných kategorií a jejich pojmenovávání by mělo být výstižné a hlavně krátké. Dnes často nalezneme u jednotlivých položek v navigaci i příslušné ikony pro jednoduší grafické znázornění. (Lidwell, 2011)

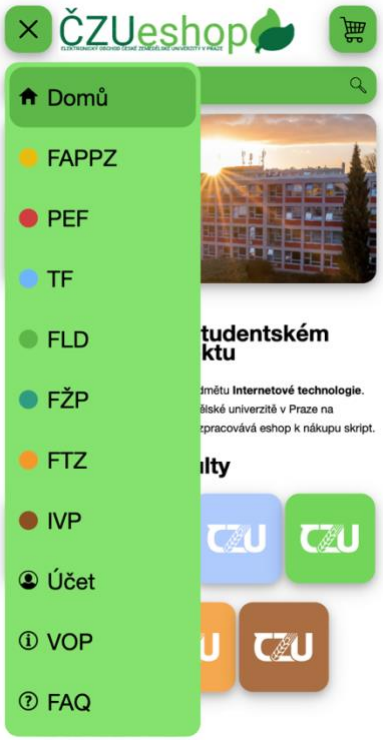
Seřazení jednotlivých položek v navigaci se pak odvíjí nejvíce od toho, pro jaké zobrazení návrh vytváříme. Jinak se dnes zobrazují navigace na desktopových zařízeních a jinak například na mobilních telefonech. Pro mobilní telefony se dnes používají tzv. hamburger menu, kdy se kompletní navigace rozbalí až po kliknutí na ikonu menu. Položky jsou pak kvůli rozlišení mobilních zařízení seřazeny vertikálně oproti horizontálnímu desktopovému zobrazení. Příklad rozdílného zobrazení těchto dvou variant je vyobrazen na následujících obrázcích v návrhu nové webové stránky pro ČZUeshop se skripty.

Na obrázcích je vidět menu stejné webové stránky ve dvou zobrazeních. Návrhy jsou vlastní prací a jejich vyhotovení bylo dílčím cílem v rámci jednoho z předmětů při studiu.




**Obrázek č. 23 - UI návrhový vzor navigace – desktopové zobrazení**


**Obrázek č. 24 - UI návrhový vzor navigace – mobilní zobrazení**




**Obrázek č. 25 - UI návrhový vzor navigace – hamburger menu po otevření**

### c) Návrhový vzor – střídání barev řádků v tabulce

Dalším návrhovým vzorem, který se běžně využívá v případě zobrazování dat v tabulkách je střídání barev řádků v tabulce. To se využívá kvůli lepší čitelnosti a jednoznačné odlišitelnosti řádku a jejich hodnot v tabulkách.

| | A | B | C |
|---|---|---|---|
| 1 | **Test** | **Date** | **Product** |
| 2 | KTE12.0-Test1 | 12/1/2015 | KTE |
| 3 | KTE12.0-Test2 | 12/2/2015 | KTE |
| 4 | KTE12.0-Test3 | 12/3/2015 | KTE |
| 5 | KTE12.0-Test4 | 12/4/2015 | KTE |
| 6 | KTE12.0-Test5 | 12/5/2015 | KTE |
| 7 | KTO7.9.0-Test1 | 12/6/2015 | KTO |
| 8 | KTO7.9.0-Test2 | 12/7/2015 | KTO |
| 9 | KTO7.9.0-Test3 | 12/8/2015 | KTO |
| 10 | KTO7.9.0-Test4 | 12/9/2015 | KTO |
| 11 | KTW10.0-Test1 | 12/10/2015 | KTW |
| 12 | KTW10.0-Test2 | 12/11/2015 | KTW |
| 13 | KTW10.0-Test3 | 12/12/2015 | KTW |

**Obrázek č. 26 - UI návrhový vzor střídání barev řádků (ExtendOffice.com, 2021)**

Jednou z nejdůležitějších vlastností všech prvků umístěných v uživatelském rozhraní je jejich barva. S použitím barev je spjata spousta pravidel, které by se neměly porušovat, pokud chce návrh dosáhnout dobrého výsledku. Volbou barev můžeme velice ovlivnit uživatelovo vnímání softwaru.

Základním pravidlem je, že by se měla jako pozadí hlavní barvy volit její barva komplementární. Díky tomu je zajištěna dobrá čitelnost textů, jelikož je zde největší barevný kontrast. Toto pravidlo by se mělo aplikovat například pro odlišení důležitých prvků v systému jako jsou třeba tlačítka. Pokud hlavní používanou barvou v systému bude například modrá, tak její komplementární barvou, která by se měla pro tlačítko použít, je barva oranžová. Volba odstínu či sytosti je pak už na uvážení designéra. (Lidwell, 2011)

**Obrázek č. 27 - Hlavní a komplementární barvy (Sibera-servis.cz, 2016)**

Pro mobilní aplikace jsou využívány odlišné návrhové vzory, než je tomu u desktopových aplikací. Takže je důležité znát cíl a využití vytvářeného softwaru, pro který je rozhraní navrhováno a dle toho poté volit, jaké vzory je správné použít.

Návrhových vzorů v oblasti návrhů UI, kterými se UI designér řídí, je opravdu hodně a každý se hodí a využívá v jiných případech či v kombinaci s ostatními. UI/UX designéři se těmto vzorům učí někdy i celý život, protože dobře navržené uživatelské rozhraní nelze tak jednoduše navrhnout. Zda je navrženo dobře se pozná až po samotném otestování uživateli, kteří často daný software vnímají úplně odlišně než samotný designér, a i proto je tento obor tak komplexní a složitý.

Ve výsledku je důležité držet se hlavních zásad a pravidel pro vytvoření dobře navrženého uživatelského rozhraní, kterými jsou následující:

- Orientování na uživatele (uživatel vždy na prvním místě).
- Zobrazení pouze užitečných informací.
- Zpětná vazba uživateli při interakci.
- Tolerance chyb uživatele (lze se vracet při chybě, tlačítko zpět apod.).
- Předvídatelné chování uživatele.
- Pokud je UI komplikované k používání, tak je pravděpodobně navržené špatně.

Dalšími užitečnými zásadami, kterými je dobré se při návrhu uživatelského rozhraní držet, pak mohou být například tyto: (Lidwell, 2011)

- Minimalismus – odstranit vše, co lze odstranit.
- Responzivita – poskytnout uživateli správnou zpětnou vazbu.
- Známost – použití známých metafor.
- Konzistentnost – použití konzistentního vizuálního a interakčního jazyku.
- Ne vždy to, co vypadá dobře, bude i dobře použitelné.
- Méně je více.

### 3.6.2 UseCase

UseCase (případ užití) je popis určité interakce mezi uživatelem a systémem. Jedná se o definici požadavků zadavatele na určitou funkci systému. Pomocí případů užití lze popsat všechny požadavky, které zadavatel po systému vyžaduje. Jde o způsob, díky kterému je snadné zkontrolovat, zda vývojář dokázal do návrhu či funkčního prototypu zahrnout

všechny požadované funkcionality, které má systém splňovat. UseCase je vždy psán z pohledu uživatele a definuje co uživatel od systému očekává a co požaduje.

Nevýhodou případů užití je, že nedefinují, jak by měla být daná funkcionalita do systému implementována. Neinformuje o podrobnostech chování ani vzhledu systému. Další jeho nevýhodou je, že UseCase obecně nejsou ohodnoceny prioritami. Všechny jsou tedy na stejné úrovni. Při vývoji UI tedy není zřejmé, které požadované UseCasy jsou těmi nejdůležitějšími. (Lidwell, 2011)

### 3.6.3 Scénář

Scénář je oproti UseCase psán z pohledu systému. Scénář podrobně popisuje chování systému, a jaké akce provádí v reakci na podněty uživatele. Scénáře jsou často psány velice detailně, až na úroveň jednotlivých prvků. Scénář je využíván v kombinaci s logickými či grafickými návrhy, ze kterých není patrné chování systému. (Lidwell, 2011)

Scénář se váže na předem definovaný UseCase. Ke každému UseCase musí být vytvořen nejméně jeden scénář. Je důležité, aby byly zachyceny všechny možné interakce uživatele popsané konkrétním UseCase.

Scénáře by měly být definovány velmi podrobně. Pokud jsou totiž definovány detailně, mohou se využít i v testovací fázi jako scénáře testovací. Vývojový cyklus systému se tímto může značně zkrátit.

### 3.6.4 Logický návrh / wireframe (Lo-Fi model)

Logický návrh někdy také nazýván jako wireframe (drátový model) či jako Lo-Fi (Low Fidelity) model je formou grafického znázornění uživatelského rozhraní. Zobrazuje zvolené komponenty a prvky a jejich rozvržení v prostoru rozhraní systému, které budou zprostředkovávat požadované funkce. Tvoří se tedy pro definování základní představy o návrhu UI. Zpravidla se v něm nepoužívají barvy ani další grafické úpravy. Je snaha ho udržet na co nejjednodušší úrovni. Často pouze jako černobílé rozvržení prvků v podobě obdélníků či obrysů jednotlivých komponent. Slouží jako základ pro vytvoření následného grafického modelu či interaktivního prototypu uživatelského rozhraní.

Logický návrh může být zpracován i pouze za pomoci tužky a papíru nebo dnes již častěji za pomoci grafických programů. Počet návrhů pro jedno uživatelské rozhraní se odvíjí od konkrétního projektu, ale minimem je vždy alespoň jeden návrh ke každé obrazovce. (Lidwell, 2011)

### 3.6.5 Grafický návrh (H-Fi model)

Grafický návrh vychází vždy z logického návrhu. Popisuje finální podobu uživatelského rozhraní. Grafický návrh se využívá i k finálnímu schválení podoby vyvíjeného systému. Oproti logickému návrhu je grafický návrh obohacen o finální podobu barev systému, zatímco logický návrh využívá většinou pouze barvy černé, šedé nebo bílé. V grafickém návrhu by již měly být zohledněny všechny požadavky zadavatele.

Grafický návrh může být vytvořen pouze jeden pro představu zadavatele, jak bude výsledný systém vypadat. Pokud to, ale zadavatel vyžaduje, je možné vytvořit grafický návrh ke každé obrazovce. Grafický návrh však stále postrádá zobrazení interakce mezi uživatelem a systémem. (Lidwell, 2011)

### 3.6.6 Prototyp

Prototyp je interaktivním návrhem systému. Na rozdíl od grafického návrhu je obohacen o možnost s prvky a komponentami interagovat a zobrazovat tak reakce systému na uživatelovo chování, tak jako to bude možné u finálního funkčního systému. U návrhů, které vyžadují pro jejich lepší pochopení vizualizaci interakce se systémem, je dobré prototyp vytvořit. Prototyp by měl zohledňovat všechny požadované funkční požadavky. Prototyp může být použit pro prezentaci finálního výsledku návrhu systému.

# 4 Vlastní práce

V praktické části této práce bude provedena analýza požadavků zadavatele a návrh informačního systému pro podporu správy sportovního klubu. Budou popsány fáze analýzy požadavků, návrhu systému a vývoje prototypu systému.

První částí je analýza požadavků zadavatele, která čerpá ze sběru požadavků, které byly sbírány za pomoci rozhovorů s potenciálními uživateli. Požadavky jsou podrobeny analýze požadavků a následně jsou dále upřesňovány autorem práce dle nabytých zkušeností z této oblasti. Požadavky ve své finální podobě slouží jako podklad pro vytvoření modelu tříd, stavového modelu a modelu interakcí. Druhá část se zabývá návrhem zmíněných UML modelů. Třetí části pak odpovídá návrh uživatelského rozhraní a tvorba jeho prototypu pro vytvoření představy o výsledné podobě systému.

Systém je zamýšlen na míru pro konkrétního zadavatele. Není zde tedy snaha o vytvoření univerzálního řešení pro celý trh, proto není provedena analýza konkurenčních řešení v této oblasti informačních systémů.

V závěru jsou formulovány doporučení pro budoucí vývoj a implementaci. V neposlední řade je zhodnocena kvalita návrhu informačního systému.

## 4.1 Obecný popis systému

Navrhovaný informační systém by měl efektivně podporovat všechny činnosti spojené se správnou sportovního klubu. Nejstěžejnější funkcionalitou je evidence členů a jejich docházek na jednotlivých událostech. Informační systém je zamýšlený v první řadě jako mobilní aplikace. Jeho cílem je snížit administrativní zátěž, která je kladena na trenéry a vedoucí. Ať už se jedná o správu dokumentů, potvrzení plateb za členství či evidenci údajů pro komunikaci s jednotlivými členy. Systém zajišťuje také bezpečnost uchovávaných dat, jelikož jsou chráněna autentizací uživatele či moderním šifrováním.

Systém bude poskytovat další řadu funkcionalit, které jsou požadovány ze strany zadavatele. Výčtem se jedná například o zmíněnou evidenci událostí a docházky, zadávání plateb a evidenci dokumentů, přehledné a rychlé zobrazení důležitých informací formou příspěvků, evidenci údajů konkrétních hráčů dle věkových kategorií či zadávaní omluvenek k daným událostem. Všechny požadavky jsou detailně vypsány v kapitole 4.2 Analýza požadavků.

Zásadním benefitem zamýšleného systému je tedy poskytnutí rychlé komunikace mezi členy a trenéry, snížení administrativních činností na minimum, a ve výsledku díky využívání systému, ušetření drahocenného času všech uživatelů.

## 4.2 Analýza požadavků

V této kapitole je provedena analýza požadavků na systém. Systémové požadavky byly sbírány metodou rozhovorů s budoucími uživateli, kterými jsou v tomto případě členové vedení klubu. Analýza požadavků je výchozím bodem pro návrh celého systému, proto je nutné, aby byla provedena, co nejlépe. Níže jsou nejdříve vypsány požadavky v bodech, které byly získány z jednotlivých rozhovorů. Nachází se tedy ve velmi hrubém formátu. Tyto požadavky byly podrobeny analýze a jejím výstupem jsou dvě přehledné tabulky – funkční a nefunkční požadavky.

Z rozhovorů se zadavatelem byl vytvořen následující seznam hrubých požadavků, které by měl systém umožňovat:

1. Mobilní aplikace – iOS/Android.
2. Rozdělení dle věkových kategorií.
3. Hlavní stránka s přehledem – důležité informace apod.
4. Přidávání příspěvků na zeď. (Pouze správce).
5. Profil pro každého uživatele.
6. Více uživatelských rolí – člen, správce, trenér, hráč.
7. Přihlašování jméno, heslo.
8. Registrace člena do klubu.
9. Dokumentový server každého člena.
10. Evidence plateb člena.
11. Realizace plateb přes platební bránu třetích stran.
12. Evidence událostí – tréninky, schůze apod.
13. Možnost nominace členů na konkrétní události.
14. Možnost komentování událostí a příspěvků.
15. Evidence docházky členů na událostech.
16. Notifikace.
17. Správa členů a událostí v jednotlivých kategoriích.
18. Filtrování událostí na historické a nadcházející.

19. Možnost zobrazení profilu člena.
20. Vygenerování seznamu členů u konkrétní události. (Pouze správce).

### 4.2.1 Funkční požadavky

| ID | Požadavek | Priorita |
|---|---|---|
| **F01** | Systém bude umožňovat správu uživatelů. | 1. |
| **F02** | Systém bude umožňovat registraci nového uživatele. | 2. |
| **F03** | Systém bude umožňovat přihlášení do systému. | 1. |
| **F04** | Systém bude uživatele členit dle věkových kategorií. | 2. |
| **F05** | Systém bude obsahovat hlavní stránku s přehledem a důležitými informacemi. | 3. |
| **F06** | Systém bude umožňovat správu příspěvků na hlavní stránce. | 3. |
| **F07** | Systém bude obsahovat profil každého uživatele. | 2. |
| **F08** | Systém bude umožňovat spravovat dokumenty. | 2. |
| **F09** | Systém bude umožňovat spravovat platby. | 2. |
| **F10** | Systém bude umožňovat správu událostí. | 1. |
| **F11** | Systém bude umožňovat nominaci členů na konkrétní události. | 1. |
| **F12** | Systém bude umožňovat správu komentářů u události. | 3. |
| **F13** | Systém bude umožňovat evidovat docházku u událostí. | 1. |
| **F14** | Systém bude umožňovat rozdělení událostí na historické a nadcházející. | 1. |
| **F15** | Systém bude umožnovat zobrazení profilů všech členů. | 2. |
| **F16** | Systém bude moci generovat seznam členů dle požadavku správce. | 3. |

**Tabulka č. 2 - Funkční požadavky**

F01 – Systém bude umožňovat správu uživatelů

Jelikož se systémem budou pracovat různé kategorie uživatelů, je nutné zajistit pomocí správy uživatelů přidělení práv pro konkrétní kategorie uživatelů. To bude zajištěno pomocí uživatelských rolí. Správu uživatelů a jejich rolí bude v systému realizovat administrátor (správce). Požadované funkce jsou – přidání, editace a odebrání uživatele, nastavení jednotlivých oprávnění u uživatelských rolí a přiřazování a odebírání rolí uživatelům.

F02 – Systém bude umožňovat registraci nového uživatele

Systém umožní komukoliv, kdo chce aplikaci využívat samostatnou registraci a odeslání její žádosti ke schválení administrátorovi. Bez schválení se ale nebude moci do systému uživatel přihlásit. Registrace uživatele tak bude muset být nejdříve schválena ze strany administrátora a uživateli musí být přidělena odpovídající role.

F03 – Systém bude umožňovat přihlášení do systému

Každý uživatel se ke svému účtu bude přihlašovat pomocí uživatelského jména a zvoleného hesla.

F04 – Systém bude uživatele členit dle věkových kategorií

Každý uživatel na úrovni člena bude mít přidělenou věkovou kategorii, do které spadá. Tomu bude odpovídat také přidělení práv pro jednotlivé kategorie. Vedoucí pak mohou mít přiděleno více věkových kategorií pod svým účtem.

F05 – Systém bude obsahovat hlavní stránku s přehledem a důležitými informacemi

Systém bude obsahovat hlavní stránku s přehledem důležitých informací, na které se člověk ocitne vždy po přihlášení. Hlavní stránka bude pro všechny uživatele stejná.

F06 – Systém bude umožňovat správu příspěvků na hlavní stránce

Uživatelé s přidělenými právy budou moci přidávat, editovat a mazat zobrazený obsah na hlavní stránce.

F07 – Systém bude obsahovat profil každého uživatele

Každý uživatel bude muset pro registraci vyplnit osobní údaje, aby bylo možné je následně využívat napříč systémem. Například pro komunikaci mezi členy. Mezi základní údaje patří jméno, příjmení, email či telefonní číslo.

F08 – Systém bude umožňovat spravovat dokumenty

Do systému bude možné nahrávat dokumenty, ke kterým bude mít přístup pouze zvolený počet uživatelů. Je možné využít role a věkové kategorie k přidělení přístupu k souborům či alternativním řešením je realizace pomocí adresářů, u kterých jsou práva již předem nastavena. Nahranými dokumenty mohou být například účtenky za platby apod.

F09 – Systém bude umožňovat spravovat platby

Do systému bude možné konkrétnímu uživateli přidělit platbu. Ta bude nabývat stavů zaplaceno/nezaplaceno. Platba bude realizována pomocí platební brány třetích stran. Tedy pouze pomocí přesměrování na bránu.

F10 – Systém bude umožňovat správu událostí

V systému bude možné vytvářet události s volitelnými vlastnostmi. Každá události bude přidělena ke konkrétní věkové kategorii, které se týká. Bude možné je následně upravovat, případně mazat.

F11 – Systém bude umožňovat nominaci členů na konkrétní události

Každá událost bude přidělena ke konkrétní kategorii členů, tím pádem bude zajištěna jejich nominace na událost. O donominování jednotlivých členů, kteří nejsou v kategorii, které se událost týká, se v aktuálním stavu neuvažuje.

F12 – Systém bude umožňovat správu komentářů u události

V systému bude umožněno každému uživateli přidávat komentáře u jednotlivých událostí. Bude umožněna jejich správa – tedy přidání, editování a smazání.

F13 – Systém bude umožňovat evidovat docházku u událostí

V systému bude přehledně viditelné, kdo u dané události z nominovaných členu přijmul či odmítnul účast.

F14 – Systém bude umožňovat rozdělení událostí na historické a nadcházející

Události, které již proběhly budou označeny jako již historické a nebude u nich možné provádět úpravy. Bude na volbě správce systému, zda chce, aby byly historické události viditelné pro členy, kterých se události týkali.

F15 – Systém bude umožnovat zobrazení profilů všech členů

Uživatelé s vyššími oprávněním např. trenér si budou moci zobrazit základní informace o každém členovi. Příkladem je telefonní číslo či email určený pro komunikaci. Každý člen musí mít tedy vlastní profil s informacemi.

F16 – Systém bude moci generovat seznam členů dle požadavku správce

Funkce generování seznamu členů u konkrétní události bude sloužit k vytvoření přehledného exportu členů a jejich reakcí u požadované události z dat k určitému času. Systém bude umožňovat export např. pro tisk ve formátu PDF.

### 4.2.2 Nefunkční požadavky

| ID | Požadavek | Priorita |
|---|---|---|
| **N01** | Systém bude dostupný jako mobilní aplikace. | 1. |
| **N02** | Systém bude provádět autentizaci uživatele. | 1. |
| **N03** | Systém bude umožňovat spravovat oprávnění. | 1. |
| **N04** | Systém bude respektovat zásady použitelnosti. | 2. |
| **N05** | Systém bude rozšiřitelný. | 3. |
| **N06** | Systém bude splňovat nařízení GDPR. | 1. |
| **N07** | Systém bude možné zálohovat. | 1. |
| **N08** | Systém bude umožňovat platby pomocí platební brány třetích stran. | 2. |
| **N09** | Systém bude schopen notifikovat uživatele. | 2. |
| **N10** | Systém bude v budoucnu dostupný přes webový prohlížeč. | 4. |
| **N11** | Systém bude pro bezpečné připojení využívat protokol HTTPS. | 4. |
| **N12** | Systém bude dostupný minimálně 99 % času. | 1. |

**Tabulka č. 3 - Nefunkční požadavky**

N01 – Systém bude dostupný jako mobilní aplikace pro iOS/Android

Systém bude vytvořen jako mobilní aplikace pro nejrozšířenější operační systémy, kterými jsou iOS a Android. Systém tedy bude dostupný pro tyto operační systémy ve formě aplikace, která bude zdarma k dispozici ke stažení v příslušném online obchodě s aplikacemi.

N02 – Systém bude provádět autentizaci uživatele

Pro přihlášení do systému bude provedena autentizace uživatele na základě zadání uživatelského jména a hesla. Uživatelské heslo bude muset být tvořeno z minimálně osmi znaků, které budou kombinací písmen a čísel. Heslo bude zároveň ukládáno ve formě hashe, což zaručí vyšší bezpečnost systému. Pokud bude uživatel v aplikaci nečinný déle než 15 minut, bude ze systému automaticky odhlášen.

N03 – Systém bude umožňovat spravovat oprávnění

V systému bude možné přiřadit každému uživateli různá oprávnění dle definovaných rolí k využívání funkcionalit systému. Vše bude řešeno pomocí uživatelských rolí, kterými by měly být pro začátek role člen (hráč), vedoucí (trenér) a správce. Jedním z důvodů pro udělování oprávnění je i ochrana citlivých dat v systému. Správně definovanými rolemi

uživatelů je možné zamezit přístupu nebo v horším případě dokonce přepisu dat neoprávněnými uživateli.

N04 – Systém bude respektovat zásady použitelnosti

Zásadní vlastností systému je jeho intuitivní ovládání. Proto je nutné při návrhu dodržovat návrhové vzory a zažité konvence. Jelikož je v prvotní fázi systém připravován pouze jako mobilní aplikace, je nutné, aby dodržoval zásady pro tento způsob zobrazení.

N05 – Systém bude rozšiřitelný

Systém by měl být připravený na jakákoliv budoucí rozšíření funkcionalit systému. Lze tedy vyvinout modul s novými funkcionalitami, který bude následně pouze implementován do stávajícího řešení bez nutnosti většího přepracovávání systému.

N06 – Systém bude splňovat nařízení GDPR

Systém bude splňovat pravidla definovaná nařízením o ochraně osobních údajů (GDPR). Osobními údaji, u kterých je nutné dodržet ochranu osobních údajů evidovaných v systému mohou být např. jméno a příjmení, datum narození, pohlaví, adresa, fotografie atd. Ochrana osobních údajů se vztahuje na všechny údaje týkající se evidovaných fyzických osob v systému. (MVČR, 2023)

N07 – Systém bude možné zálohovat

Předpokládá se, že systém bude uchovávat důležitá data, která je nezbytné mít zálohované, v případech ztráty dat z jakýchkoliv důvodů. Je tedy nezbytné nastavit optimální frekvenci zálohování systému, a to zároveň s ohledem na finanční náklady na uchování záloh.

N08 – Systém bude umožňovat platby pomocí platební brány třetích stran

Platby, které bude uživatel moci realizovat z aplikace budou fungovat za pomoc přesměrování uživatele na platební bránu třetích stran. Za uskutečnění transakce tak ponese odpovědnost uživatel a provozovatel platební brány.

N09 – Systém bude schopen notifikovat uživatele

Systém bude umožňovat posílat uživateli notifikace o důležitých informacích, nově vytvořených událostí apod.

N10 – Systém bude v budoucnu dostupný přes webový prohlížeč

Informační systém bude v budoucnu dostupný pomocí internetového prohlížeče. Systém by měl podporovat přístupnost ze všech základních webových prohlížečů jakými jsou např. Chrome, Opera, Edge, Firefox, Safari a další běžně používané prohlížeče. Ve všech prohlížečích by měl zůstat vzhled i funkcionality zachovány.

N11 – Systém bude pro bezpečné připojení využívat protokol HTTPS

Všechna komunikace systému by měla být realizována za pomoci zabezpečeného protokolu HTTPS využívajícím např. kryptografický protokol TLS 1.3. Díky tomu by měla být zajištěna bezpečnost a důvěryhodnost komunikace.

N12 – Systém bude dostupný minimálně 99 % času

Systém bude zaručovat svou dostupnost minimálně 99 % času. Do této doby se nezapočítává nezbytná údržba systému. Dostupnost systému ve 100 % případů nelze ani s využitím moderních technologií nikdy zaručit.

## 4.3 Návrh systému

Na základě analýzy požadavků a teoretických východisek, popsaných v první časti práce, je navržen model tříd, stavový model a model interakcí. Všechny modely jsou vytvořeny ve standardech jazyka UML za pomoci nástroje Draw.io. Model stavů a interakcí je nejdetailněji zaměřen na princip evidence docházky u událostí, která je stěžejní funkcionalitou systému. Dále jsou vytvořeny wireframy s návrhem uživatelského rozhraní pro mobilní aplikaci systému. Následně je vytvořen interaktivní prototyp, který je obsažen v příloze této práce. V poslední řadě je doporučen možný způsob implementace systému za pomoci diagramu nasazení.

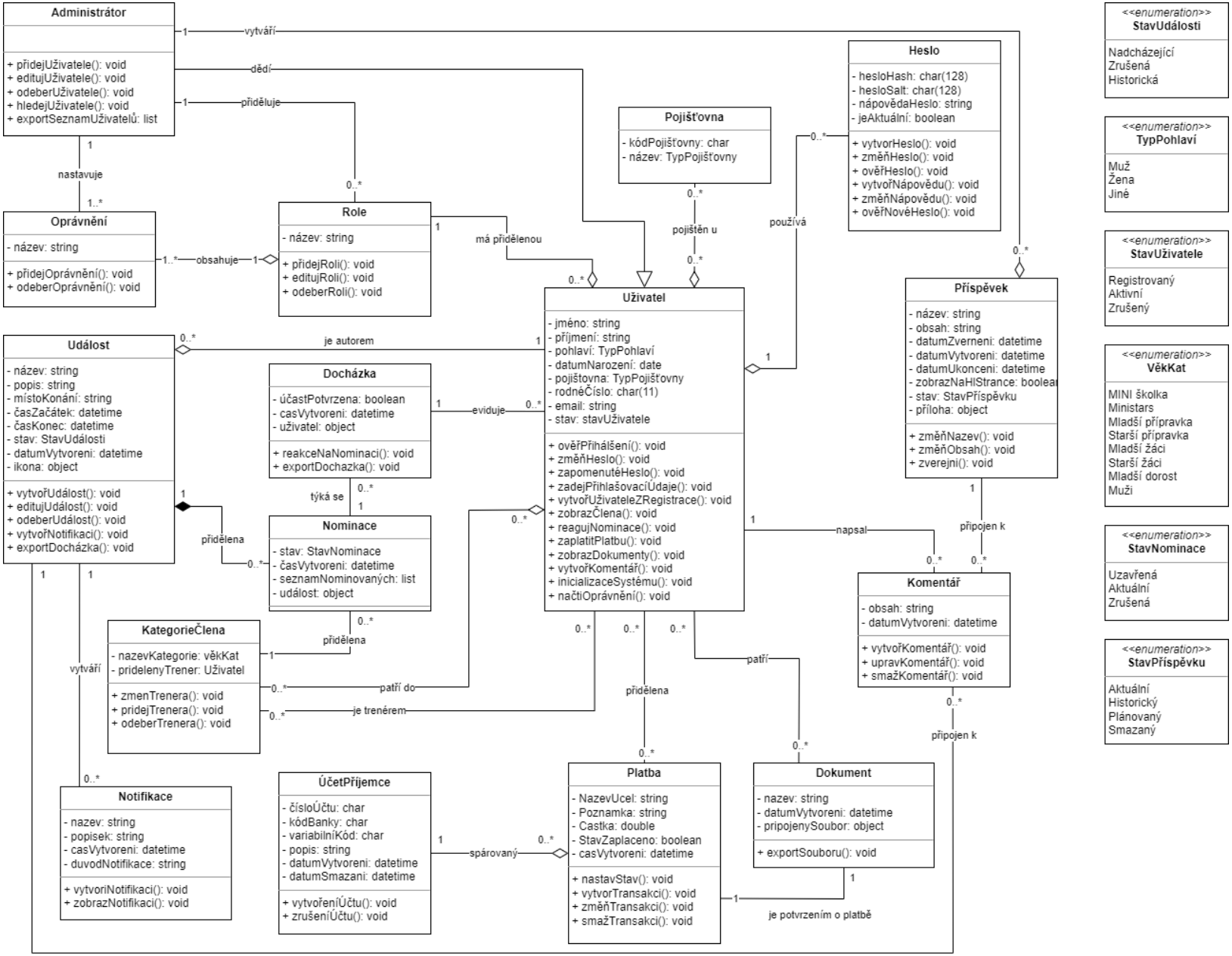
### 4.3.1 Model tříd

Navržený model tříd je reprezentován diagramem tříd, který je vyobrazen na obrázku Obrázek č. 28. Jeho cílem je zobrazit statickou strukturu systému, za pomoci zobrazení jednotlivých tříd a vazeb mezi nimi.

Dle metodiky pro tvorbu diagramu tříd byly nejdříve identifikovány hlavní třídy, které systém pro splnění všech požadavků vyžaduje. Názvy tříd byly zvoleny tak, aby byly jednoznačně identifikovatelné a odpovídaly objektům zájmu, které reprezentují. Dále byly přidány atributy jednotlivých tříd a následně jsou identifikovány vazby mezi třídami. Až poté byly upřesněny typy jednotlivých vazeb. V návrhu jsou použity všechny 4 typy vazeb. V neposlední řadě byla doplněna násobnost vazeb a jejich popis pro jednoduší interpretaci. Dále byly přidány operace, o kterých se předpokládá, že budou v systému muset být implementovány. Pro atributy byly doplněny datové typy, v nutných případech byly vytvořeny číselníky pomocí enumerace pro konkrétní účely systému. V průběhu tvorby dalších modelů byl diagram tříd nadále zpřesňován.

Diagram tříd je doplněn o datový slovník, pro jednoznačné pochopení samotného návrhu diagramu tříd. Datový slovník obsahuje detailnější popis jednotlivých tříd pro jejich jednoduší interpretaci.

V diagramu tříd je pokryta pouze nezbytně nutná část systému (tříd) k zajištění jeho fungování pro účely této práce. V následné reálné implementaci by pro realizaci systému bylo zapotřebí pravděpodobně větší množství tříd. Není tedy cílem zachytit úplný detail popisu všech tříd, atributů a operací systému, které by byly obsaženy ve finální implementaci.




**Obrázek č. 28 - Diagram tříd**

### 4.3.2 Datový slovník

Třída Uživatel

Reprezentuje osobu, která je registrována případně přihlášena v systému. Podle přidělených Oprávnění získává uživatel možnost interagovat se systémem.

Třída Administrátor

Je podtřídou třídy Uživatel. Administrátor dědí od třídy Uživatel všechny jeho atributy a operace, a navíc přidává své vlastní. Na rozdíl od třídy Uživatel má tedy přiřazená nejvyšší možná oprávnění v systému, díky kterým má možnost spravovat všechny ostatní uživatele systému.

Třída Role

Obsahuje oprávnění pro konkrétní uživatelskou roli (kategorii). Mezi třídami Oprávnění a Role je použita vazba typu agregace. Za přiřazení rolí zodpovídá Administrátor systému.

Třída Oprávnění

Omezuje používání funkcí systému pro různé uživatele pomocí uživatelských rolí. Přidělené oprávnění rolím definují, jaké funkce systému může uživatel používat. Nastavení oprávnění daným rolím a role uživatelům zajišťuje a přiděluje Administrátor.

Třída Heslo

Třída heslo je propojena s třídou Uživatel, tak aby bylo možné ke každému uživateli asociovat jeho otisk hesla dle cizího klíče. Kvůli zvýšení zabezpečení je navrženo ukládat pouze hash reálného hesla. Při vložení hesla uživatele je dle šifrovací funkce např. SHA-128, pro kterou je vhodný datový typ pro uložení hashe char(128), heslo zakódováno. K tomu je použit řetězec uložený jako tzv. sůl, tím získáme číslo, ze kterého nelze původní heslo snadno zpětně odvodit. Další používanou jednosměrnou hashovací funkcí je např. MD5. Když je tedy potřeba uživatele autentizovat, je jím zadané heslo zakódováno stejným algoritmem (stejnou hashovací funkcí a uloženým řetězcem, který byl použit jako "sůl") a výsledek je porovnán s uloženým tvarem. V případě shody je zadané heslo správné.

Třída Pojišťovna

Obsahuje informace o pojišťovně, u které je zaregistrován daný uživatel. Jedná se o výběr z nabídky evidovaných pojišťoven. Jejich data spravuje Administrátor.

Třída KategorieČlena

Uchovává informace o jednotlivých věkových kategoriích, do kterých může daný uživatel spadat. Uživatel může mít i více přidělených kategorií např. v případě trenérů. Data opět spravuje pouze Administrátor a kategorii členům přiděluje také Administrátor.

Třída Nominace

Poskytuje informaci o docházce člena na konkrétní Události. Přidělení členové k vytvořené události jsou přebráni z jedné konkrétní věkové kategorie, pro kterou je událost vytvořena. Eviduje se stav každého člena, který je z nominované věkové kategorii u události přidělen. Aktuálně lze nominovat pouze celou věkovou kategorii, nikoliv samostatné členy. Docházka lze měnit ze strany nominovaných členů pouze dokud Událost neproběhla.

Třída Událost

Obsahuje informace o dané Události. Jedná se o různé typy akcí, zápasů či tréninků apod. Jejich data spravuje opět pouze Administrátor. Třída podporuje historizaci všech událostí.

Třída Docházka

Docházka obsahuje informaci, zda uživatel potvrdil či zrušil svou účast na události, ke které je nominován. Nominace uživatelů k událostem probíhá pomocí zahrnutí uživatele ve věkové kategorii. Docházka tak propojuje třídy Nominace a Uživatel. U nominace je pak evidovaná konkrétní událost, ke které se Nominace váže. Lze také zobrazit Docházka jednoho Uživatele dle potvrzení účasti a provázanosti na konkrétní Událost přes její Nominaci.

Třída Notifikace

Udržuje informaci o vygenerovaných notifikacích, které se vztahují k jednotlivým událostem. Jsou vygenerovány při vytvoření Události, změně jakéhokoliv detailu události

apod. Zdrojová data pro zobrazení upozornění, lze tak získat jednoduše přímo z této třídy, která obsahuje všechny potřebné údaje k zobrazení.

Třída Příspěvek

Představuje příspěvky, které jsou vytvořeny Administrátorem pro sdělení důležitých informací na hlavní stránce. Lze je umisťovat na hlavní stránku a na stránku všech příspěvků. Po uplynutí lhůty pro zobrazení je příspěvek schován. Spravuje je Administrátor.

Třída Komentář

Uchovává informace o všech komentářích napsaných uživateli pod konkrétní Příspěvek či Událost.

Třída Dokument

Obsahuje informace o Dokumentech, ke kterým mají uživatelé přístup a mohou si je například stáhnout na lokální uložiště. Ať už se jedná o potvrzení platby či jiné dokumenty.

Třída Platba

Udržuje informace o nově vytvořené platbě přidělené konkrétnímu uživateli či o historii již realizovaných platbách v rámci systému. Vztahuje se nejčastěji k zaplacení členského poplatku apod. Spravuje je Administrátor.

Třída ÚčetPříjemce

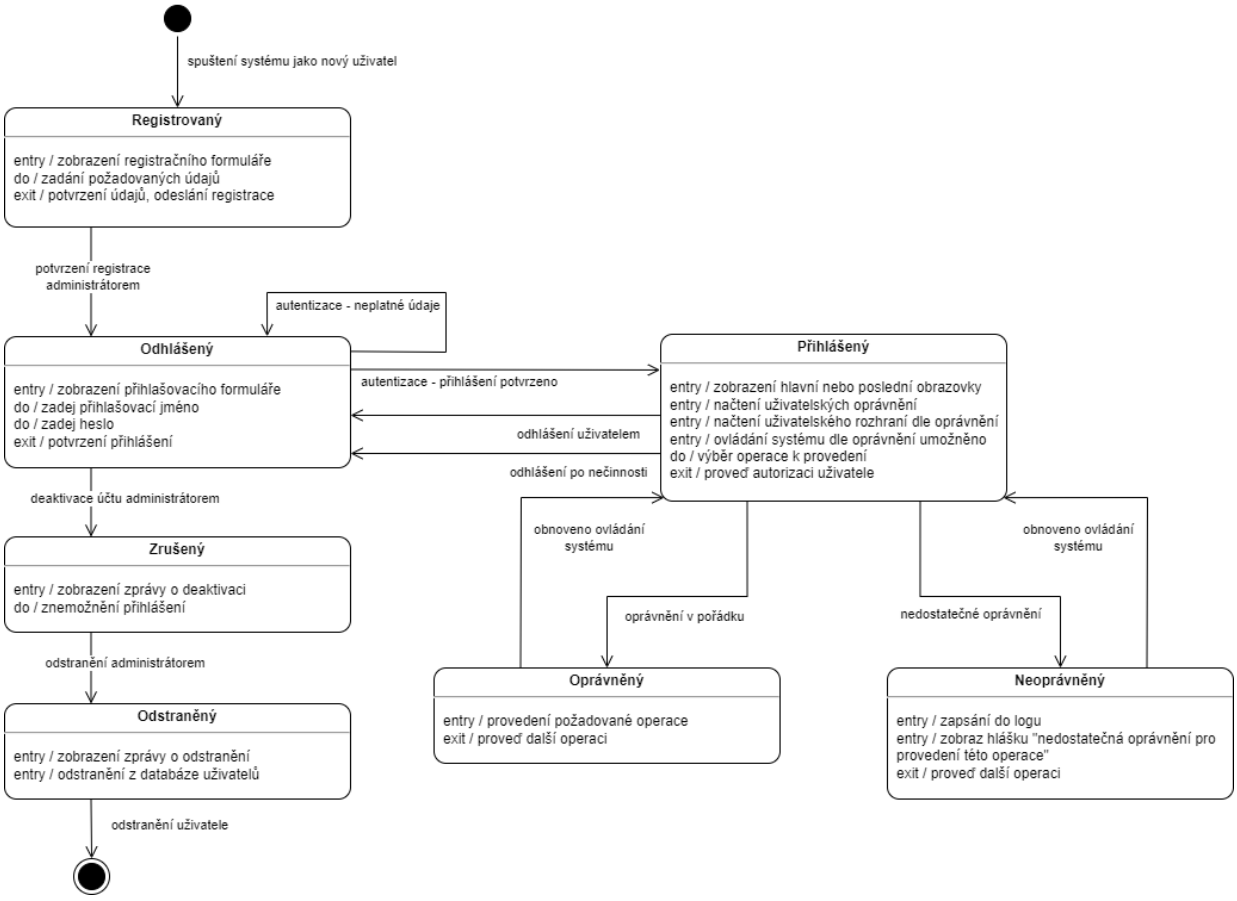
Obsahuje informace o účtech klubu, na které si chce klub nechat zasílat peníze ze zaplacených plateb. Jedná se o jakousi šablonu pro vyplnění účtu u vytváření nové platby v systému. Jinak by bylo nutné jednotlivé položky účtu vždy ručně vyplňovat. Opět spravuje pouze Administrátor.

### 4.3.3 Stavový model

U stavových diagramů je vždy nutné zvážit, zda bude tvorba stavového diagramu pro model užitečná. V rámci této práce byly k vytvoření zvoleny stavové diagramy pro třídy *Uživatel, Událost a Příspěvek.* V modelu systému se nacházejí i další třídy které nabývají více stavů. Pro účely této práce je ale není potřeba detailněji vysvětlovat.

#### a) Stavový diagram třídy Uživatel

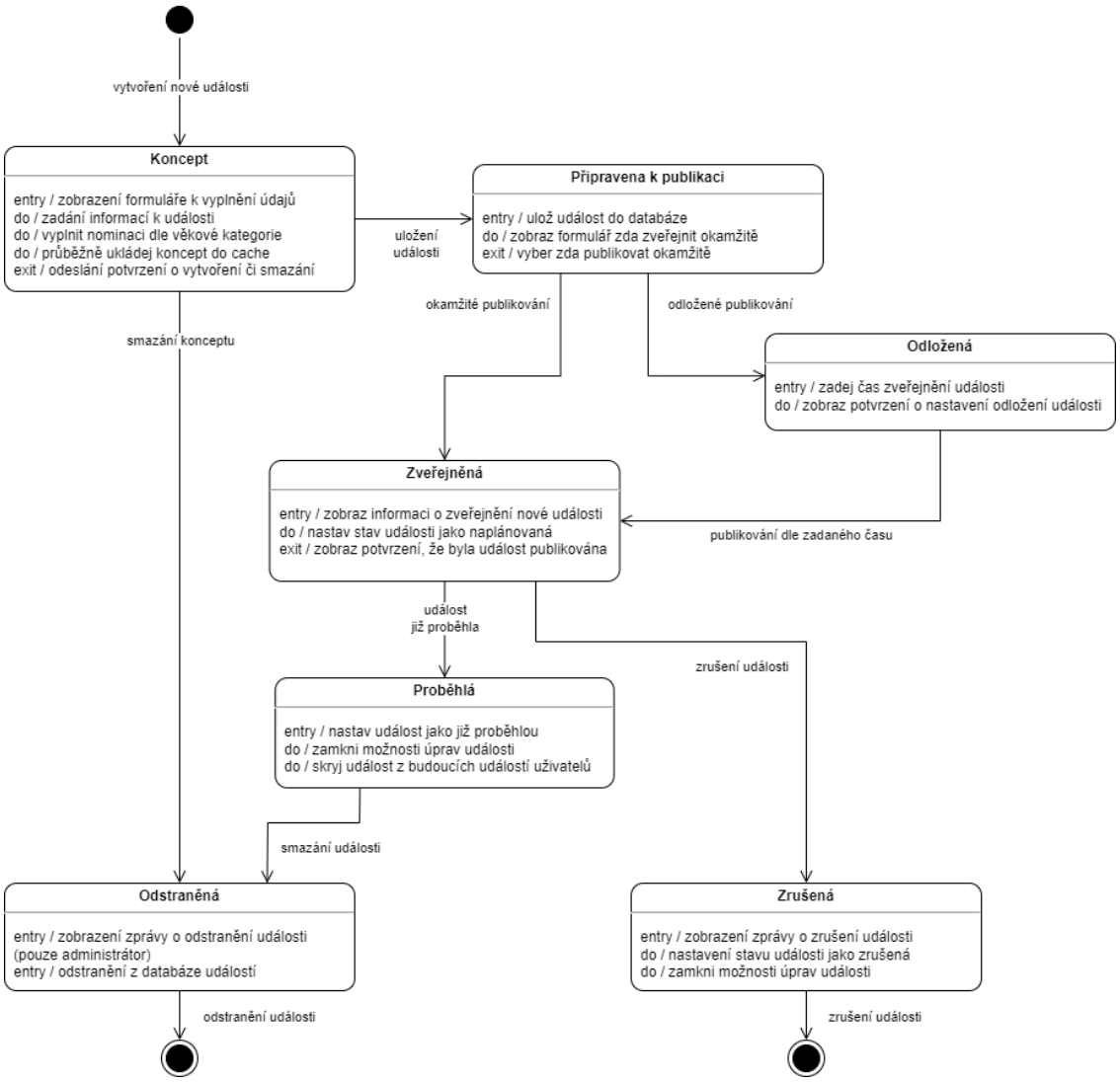
Stavový diagram zobrazuje stavy třídy Uživatel. Po registraci je ve stavu Registrovaný, po schválení umožňuje uživateli přihlašování. Výchozí pro aktivovaného uživatele je stav Odhlášený. Dále v systému může procházet dalšími pěti stavy – Přihlášený, Oprávněný, Neoprávněný, Zrušený a Odstraněný.




**Obrázek č. 29 - Stavový diagram třídy Uživatel**

### b) Stavový diagram třídy Událost

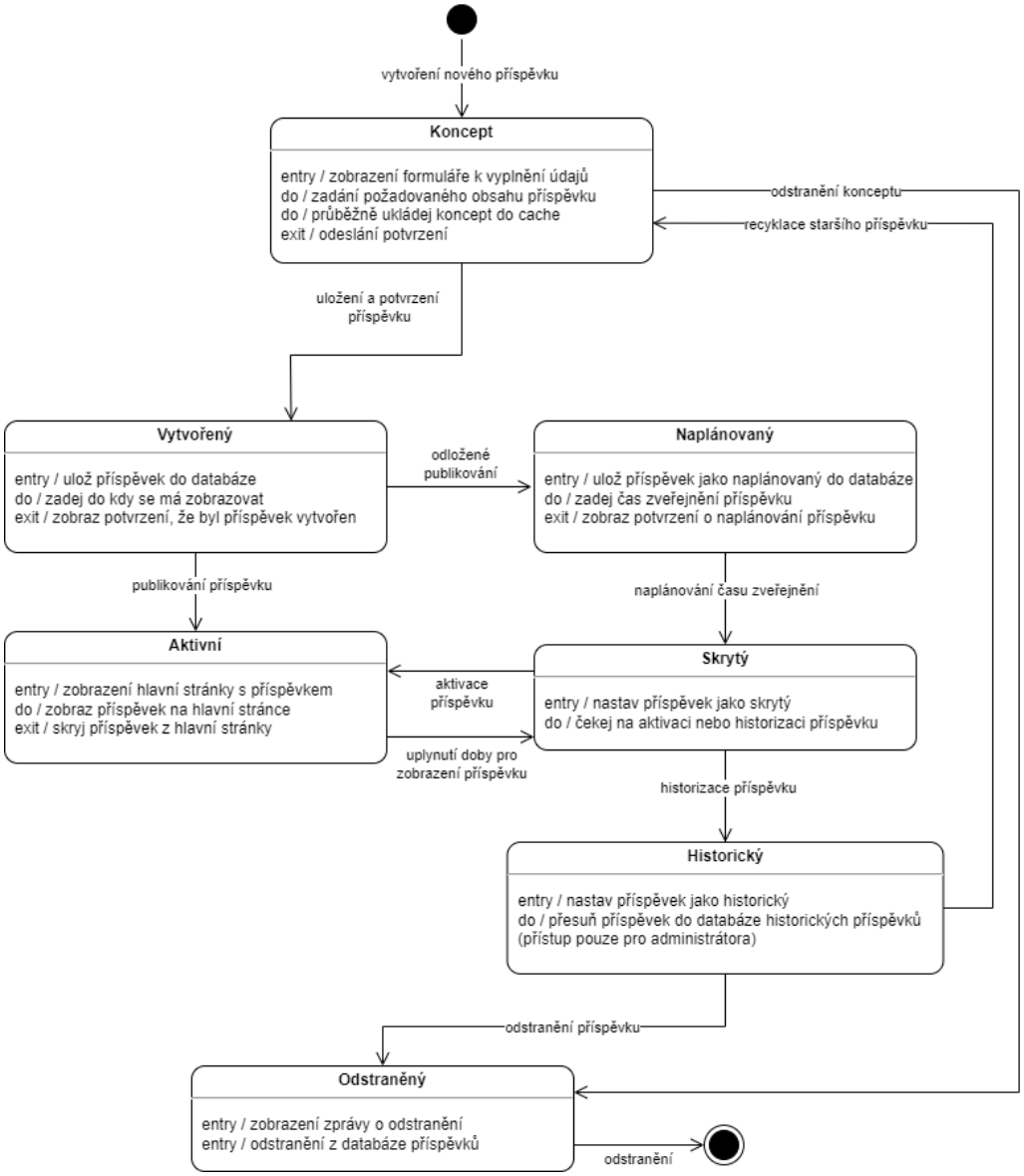
Stavový diagram pro třídu Událost zobrazuje proces životního cyklu Události v systému. Po vytvoření nové instance třídy se nachází událost ve výchozím stavu Koncept. Poté je možné uložit koncept a tím přejit do stavu Připravena k publikaci nebo koncept smazat a tím stav události změnit na Odstraněná a tím proces ukončit. Pokud je událost uložena, je možné ji okamžitě zveřejnit anebo ji zveřejnit odloženě. Pokud událost již proběhla nastaví se automaticky do stavu Proběhlá. Pokud událost administrátor ručně zruší, ještě před uplynutím doby, po kterou je zveřejněna, je přesunuta do stavu Zrušená.




**Obrázek č. 30 - Stavový diagram třídy Událost**

### c) Stavový diagram třídy Příspěvek

Poslední stavový diagram zobrazuje stavy třídy Příspěvek. Při vyžádání vytvoření příspěvku se zobrazí formulář s údaji, které má příspěvek obsahovat – název, obsah apod. Jako výchozí stav je tedy stav Koncept. Poté je možnost příspěvek zveřejnit okamžitě či ho naplánovat. Po zvolení volby a následném zadání času zveřejnění, případně pouze zadání času do kdy se má příspěvek na hlavní stránce zobrazovat se přesune do stavu Skrytý, respektive Aktivní. Po uplynutí doby zobrazení je příspěvek skryt z hlavní stránky a je možné ho historizovat či znovu zaktivovat. Historizované příspěvky lze ze systému úplně smazat či je recyklovat a upravit jejich obsah pro nově vytvářený příspěvek.




**Obrázek č. 31 - Stavový diagram třídy Příspěvek**

### 4.3.4 Model interakcí

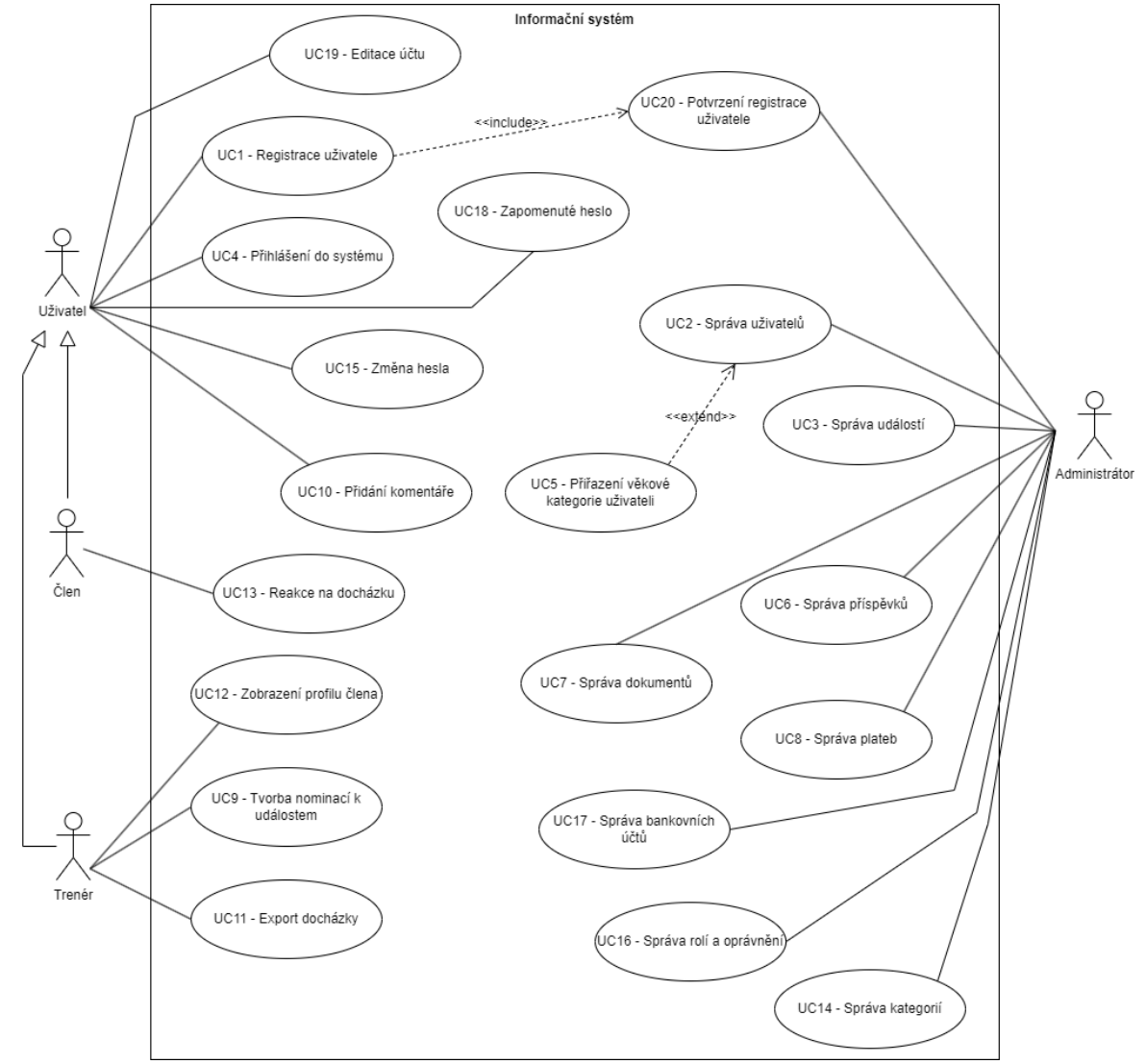
**a) Diagram případů užití**

Tento diagram zachycuje celkový pohled na systém a jeho základní složkou jsou aktéři, kteří interagují se systémem. Případy užití jsou definovány dle funkčních požadavků a měli by je všechny pokrývat. S vytvářeným systémem budou pracovat tři možné typy aktérů – uživatel s rolí člen, uživatel s rolí trenér a administrátor.

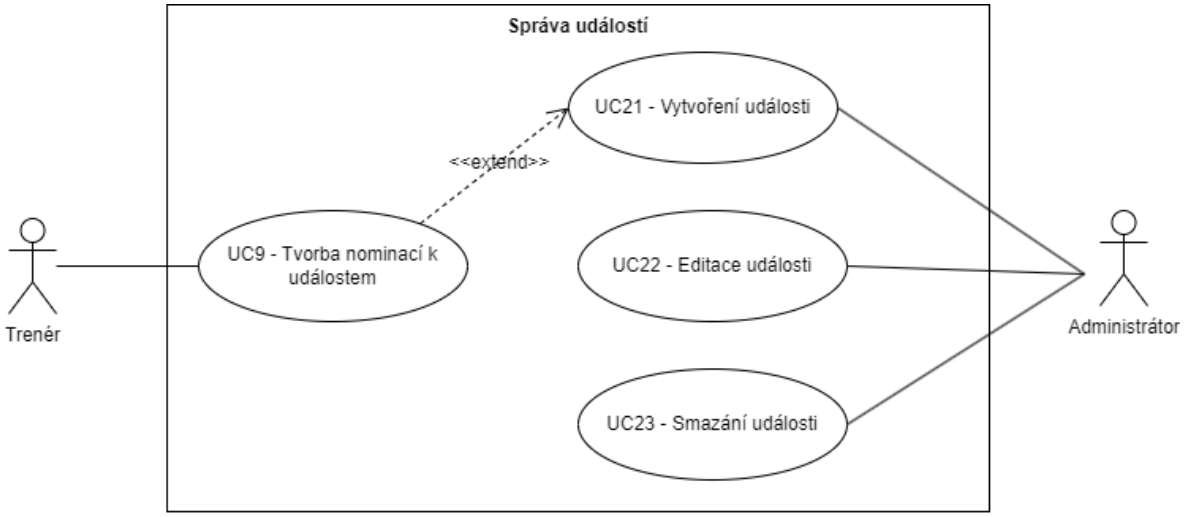
Aktéry jsou v diagramu uživatelé s rolemi – administrátor s nejvyšším oprávněními, obecná role uživatele, od které následně dědí vše jeho potomci člen a trenér a přidávají se u nich navíc specifické případy užití. Uživatelem je v systému každá fyzická osoba, která má v systému založen uživatelský účet a pracuje s funkcionalitami systému na základě přidělených oprávnění. Administrátorem je zvolená osoba, která má na starost správu všech uživatelů v systému a dohlíží na bezproblémový chod celého systému.

**b) Dekompozice případu užití – Správa událostí**

Pro představu byla provedena dekompozice jednoho ze složitějších případů užití. Byl zvolen případ užití pro správu událostí, který v sobě zahrnuje více dílčích případů užití, které mohou nastat. Vše je zachyceno na Obrázek č. 33.




**Obrázek č. 32 - Celkový diagram případů užití**




**Obrázek č. 33 - Dekompozice případu užití Správa událostí**

### c) Mapování případů užití na funkční požadavky

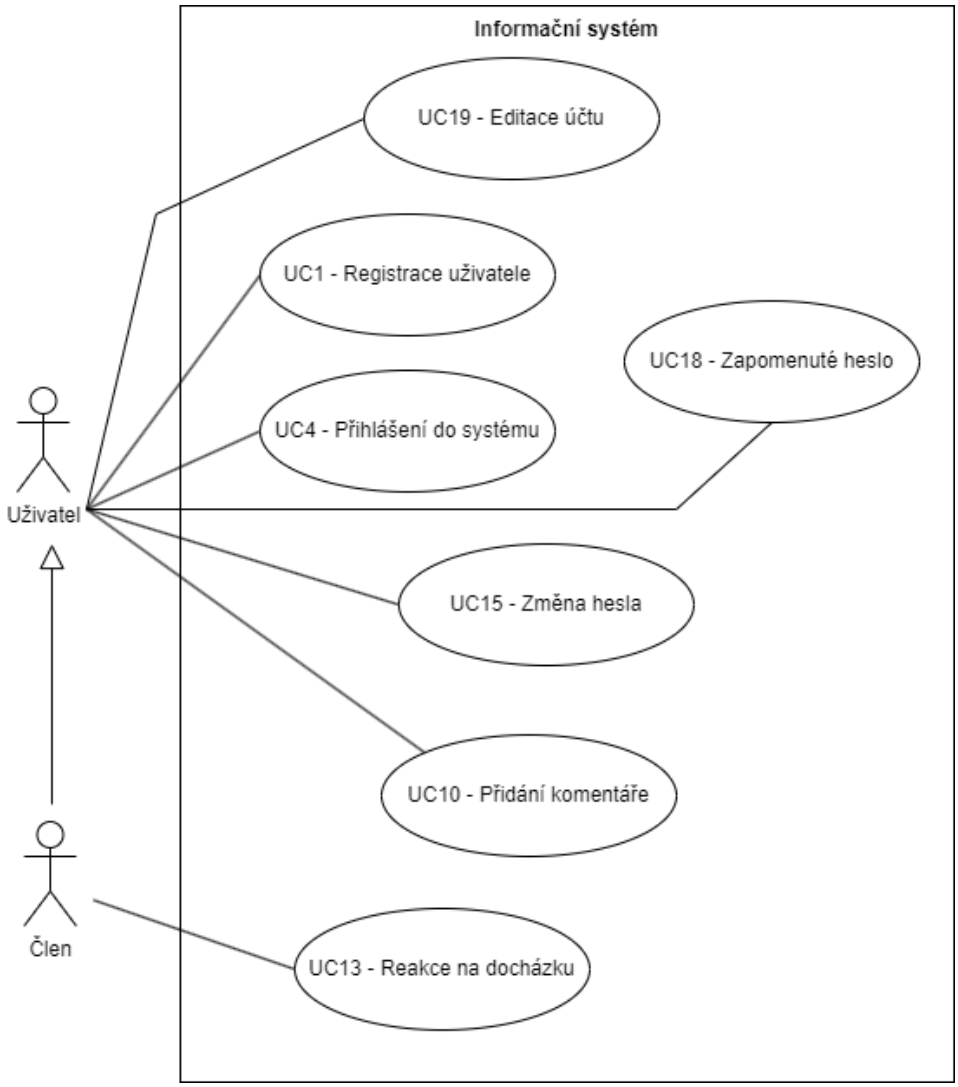
V následující tabulce je zachycena realizace jednotlivých funkčních požadavků příhodnými případy užití. Tuto tabulku lze využít ke kontrole splnění všech funkčních požadavků. Pokud by se v tabulce vyskytl požadavek, který není pokryt případem užití, může být tento funkční požadavek definován nesprávně a tedy zbytečně, případně nejsou identifikovány všechny potřebné případy užití. Z výsledné tabulky lze interpretovat, že všechny identifikované funkční požadavky jsou pokryty alespoň jedním případem užití.

| | F 01 | F 02 | F 03 | F 04 | F 05 | F 06 | F 07 | F 08 | F 09 | F 10 | F 11 | F 12 | F 13 | F 14 | F 15 | F 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **UC1** | | X | | | | | X | | | | | | | | X | |
| **UC2** | X | | | | | | | | | | | | | | | |
| **UC3** | | | | | | | | | | X | | | X | X | | |
| **UC4** | | | X | | | | | | | | | | | | | |
| **UC5** | X | | | X | | | | | | | | | | | | |
| **UC6** | | | | | X | X | | | | | | X | | | | |
| **UC7** | | | | | | | | X | | | | | | | | |
| **UC8** | | | | | | | | | X | | | | | | | |
| **UC9** | | | | | | | | | | | X | | | | | |
| **UC10** | | | | | | | | | | | | X | | | | |
| **UC11** | | | | | | | | | | | | | | | | X |
| **UC12** | | | | | | | X | | | | | | | | X | |
| **UC13** | | | | | | | | | | | | | X | | | |
| **UC14** | | | | X | | | | | | | | | | | | |
| **UC15** | X | | X | | | | | | | | | | | | | |
| **UC16** | X | | | | | | | | | | | | | | | |
| **UC17** | | | | | | | | | X | | | | | | | |
| **UC18** | | | X | | | | | | | | | | | | | |
| **UC19** | X | | | | | | | | | | | | | | | |
| **UC20** | | X | | | | | | | | | | | | | | |

**Tabulka č. 4 - Realizace funkčních požadavků jednotlivými případy užití**

### d) Sekvenční modely

V rámci této práce jsou modelovány pouze sekvenční modely, které představují hlavní neboli ideální scénář daného případu užití. Pro potřeby této práce byla vybrána podmnožina případů užití, ke kterým byly v další části práce vytvořeny scénáře. Vybraná podmnožina případů užití zahrnuje všechnu požadovanou funkcionalitu z pohledu běžného uživatele (role člen) systému. Tato podmnožina byla vybrána i z toho důvodu, že se jedná o případy užití, které jsou požadovány implementovat do první verze systému. Vybraná podmnožina je následně implementována i v interaktivním prototypu.




**Obrázek č. 34 - Podmnožina z diagramu případů užití zahrnutá v této práci**

Stěžejním scénářem je scénář k případu užití *přihlášení uživatele.* Ten popisuje všechny kroky pro přihlášení existujícího uživatele do systému.

| **Případ užití** | **Přihlášení uživatele** |
|---|---|
| **ID** | UC4 |
| **Stručný popis** | Po otevření systému se uživatel přihlašuje do systému. |
| **Hlavní aktéři** | Uživatel |
| **Vstupní podmínky** | Uživatel spustil systém v mobilní aplikaci nebo webovém prohlížeči. |
| **Hlavní scénář** | 1. Systém je spuštěn a řádně inicializován.<br>2. Systém zobrazí uživateli obrazovku s formulářem pro přihlášení.<br>3. Uživatel vyplní přihlašovací jméno a heslo a odešle údaje k autentizaci uživatele.<br>4. Systém ověří správnost vyplněných údajů a po úspěšném ověření přihlásí uživatele do příslušného účtu.<br>5. Systém načte z databáze uživatelova oprávnění.<br>6. Systém zobrazí uživateli jednotlivé ovládací prvky a moduly systému dle načtených oprávnění uživatele.<br>7. Systém zobrazí uživateli úvodní obrazovku po přihlášení. |
| **Výstupní podmínky** | Uživatel byl úspěšně přihlášen do systému a byla zobrazena úvodní obrazovka. |
| **Alternativní scénáře** | Neexistující uživatel v systému<br>Špatně zadané heslo uživatele<br>Zapomenuté heslo uživatele |

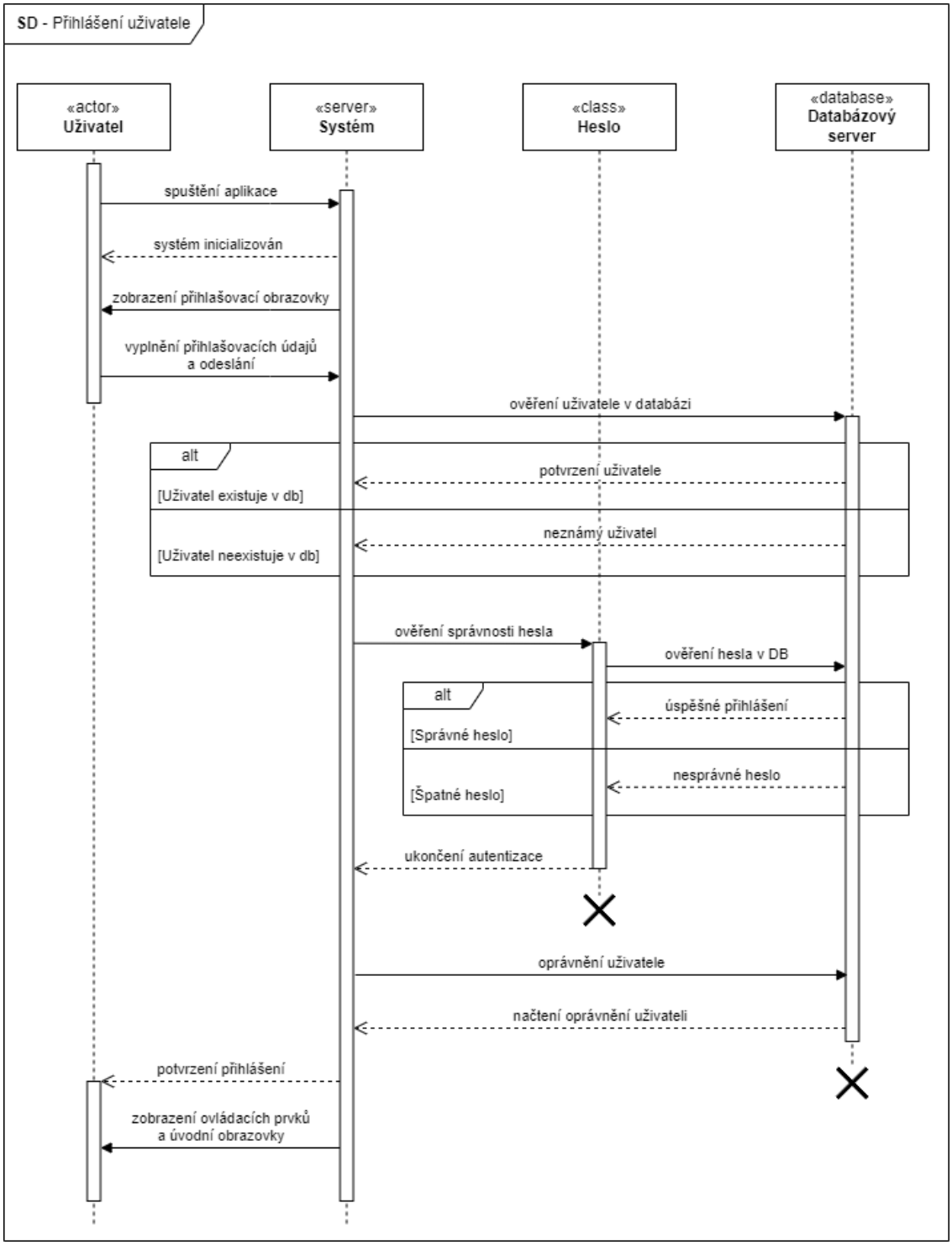
**Tabulka č. 5 - Scénář k případu užití Přihlášení uživatele**

Sekvenční diagram popisuje posloupnost činností, které se dějí při přihlášení uživatele do systému. Detailněji jsou popsány ještě v následném diagramu aktivit.

Nejdříve musí uživatel aplikaci pro informační systém spustit, tím se zavolá metoda *inicializaceSystému(),* ta spustí načítání základních nastavení systému pro inicializaci. Po inicializaci systému se uživateli na obrazovce objeví úvodní obrazovka s přihlášením do systému. Zde musí uživatel vyplnit své přihlašovací údaje, kterými jsou jeho uživatelské jméno a heslo. Po odeslání přihlašovacího formuláře musí třída Uživatel nejdříve ověřit, zda vůbec zadaný uživatel existuje v systému. To provede metoda *ověřPřihlášení()* dotazem do databáze. Po ověření uživatele se stejným způsobem ověřuje i heslo uživatele. Zda bylo zadáno správně se ověřuje pomocí metody *ověřHeslo()* ve třídě Heslo. Po skončení autentizace se z databáze načtou i přidělená oprávnění uživatele. Podle nich se poté uživateli zobrazí úvodní obrazovka po přihlášení a ovládací prvky, ke kterým má daný uživatel přístup. Před načtením úvodní obrazovky se uživateli ještě zobrazí potvrzení o úspěšném

přihlášení. V opačném případě uživatel není přihlášen a může znovu zadat heslo či zaslat žádost o nové.

Scénáře ke zbylým UseCasům jsou přiloženy v přílohách na konci této práce.



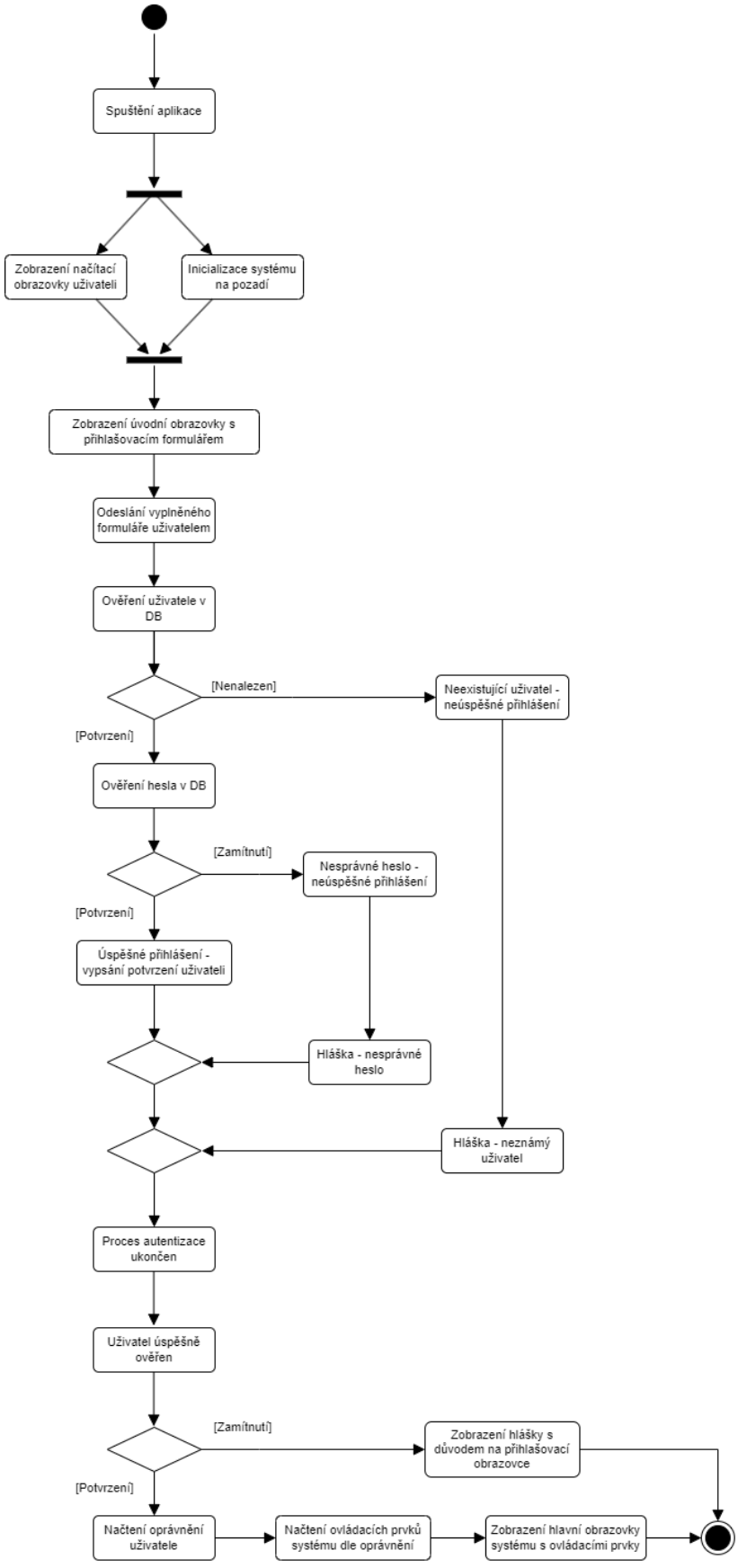

**Obrázek č. 35 - Sekvenční diagram – Přihlášení uživatele**

### e) Diagram aktivit

Diagram aktivit je v této práci vytvořen pouze pro proces *přihlášení uživatele do systému,* jelikož pro účely této práce není u jiných scénářů nutná nejnižší úroveň abstrakce. Tato práce pojednává pouze o určité podskupině funkcionalit vyvíjeného systému a zahrnutí všech potřebných modelů pro vývoj celého systému by bylo nad rámec této práce, proto byla vybrána podmnožina UseCasů, která se týká běžného (tedy nejčastějšího) uživatele.

Na následujícím obrázku, na kterém je zobrazen diagram aktivit pro přihlášení uživatele, jsou vidět jednotlivé kroky jako posloupnost operací, které jsou prováděny. V diagramu je vidět i jednoznačné větvení procesu. Jelikož diagramy aktivit vycházejí z všeobecně známých a hojně využívaných vývojových diagramů je z takto podrobného diagramu pro programátora, který má danou funkcionalitu vyvinout, velmi snadné přepsat diagram do funkčního kódu.

Diagram zobrazuje i činnosti, které jsou provedeny po zadání neznámého uživatele či špatného hesla uživatele. V diagramu je zobrazeno také jedno paralelní zpracovávání činností, kdy je uživateli zobrazována načítací obrazovka, ale na pozadí se systém inicializuje.

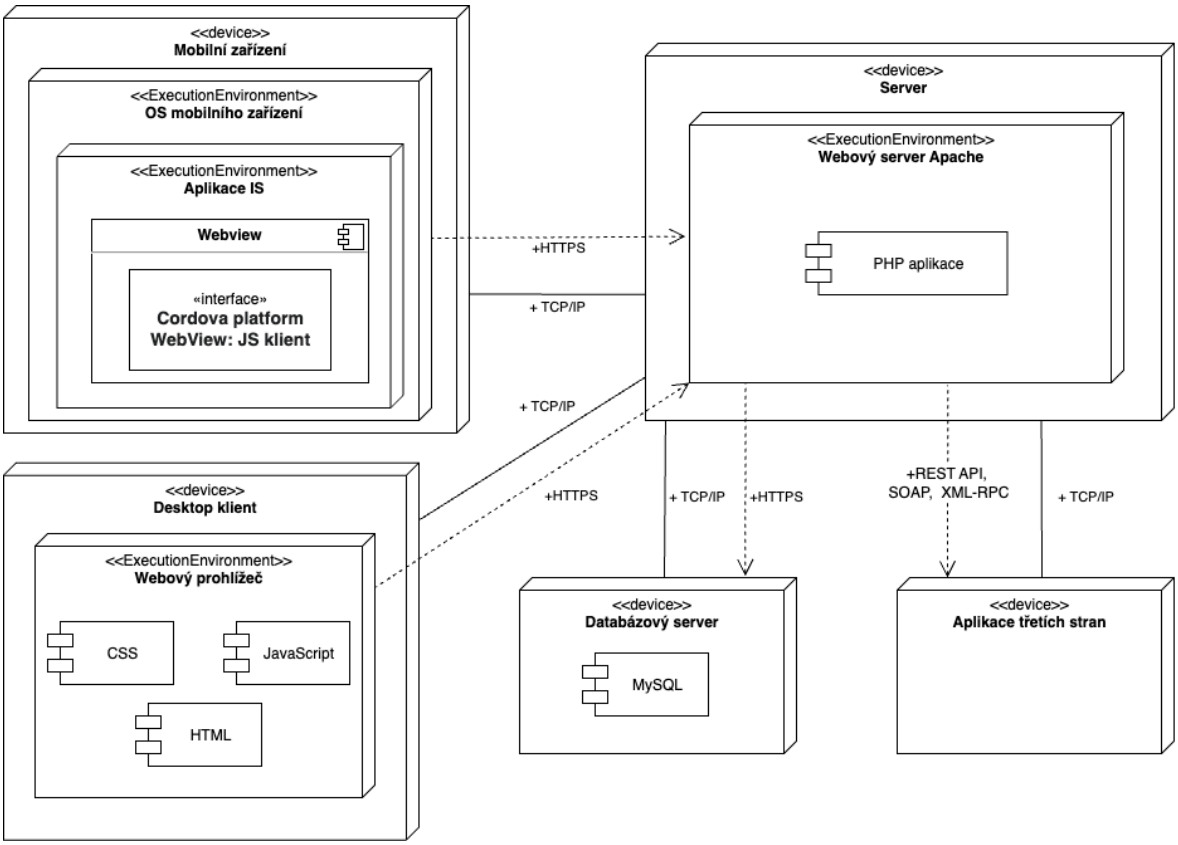



**Obrázek č. 36 - Diagram aktivit – přihlášení uživatele**

## 4.4 Návrh realizace

V rámci návrhu systému byl vymodelován i diagram nasazení, ten zachycuje fyzickou architekturu systému. Diagram nasazení je návrhem systému, který zohledňuje definované nefunkční požadavky na systém. Dle diagramu nasazení lze vidět, že autor doporučuje systém implementovat jako třívrstvou architekturu.

### Diagram nasazení




**Obrázek č. 37 - Diagram nasazení systému**

U desktopového klienta by byly využity technologie, které podporují webové prohlížeče, ty využívají technologie HTML, CSS a JavaScript. U mobilních zařízení je prezentační vrstva zajištěna mobilní aplikací. Ta může být naprogramována samostatně nebo může být realizována například pomocí technologie Apache Cordova. Jedná se o open-source framework pro vývoj mobilních aplikací, který umožňuje tzv. cross-platform vývoj. Klient komunikuje s webovým serverem pomocí protokolu HTTPS. Server může být například hojně využívaného typu Apache. Webový server poté komunikuje s databází vytvořenou například v MySQL, která je umístěná na externím databázovém serveru. Je možné

komunikovat i s aplikacemi třetích stran například pro potřebu komunikace se zdravotními pojišťovnami apod.

## 4.5 Návrh uživatelského rozhraní

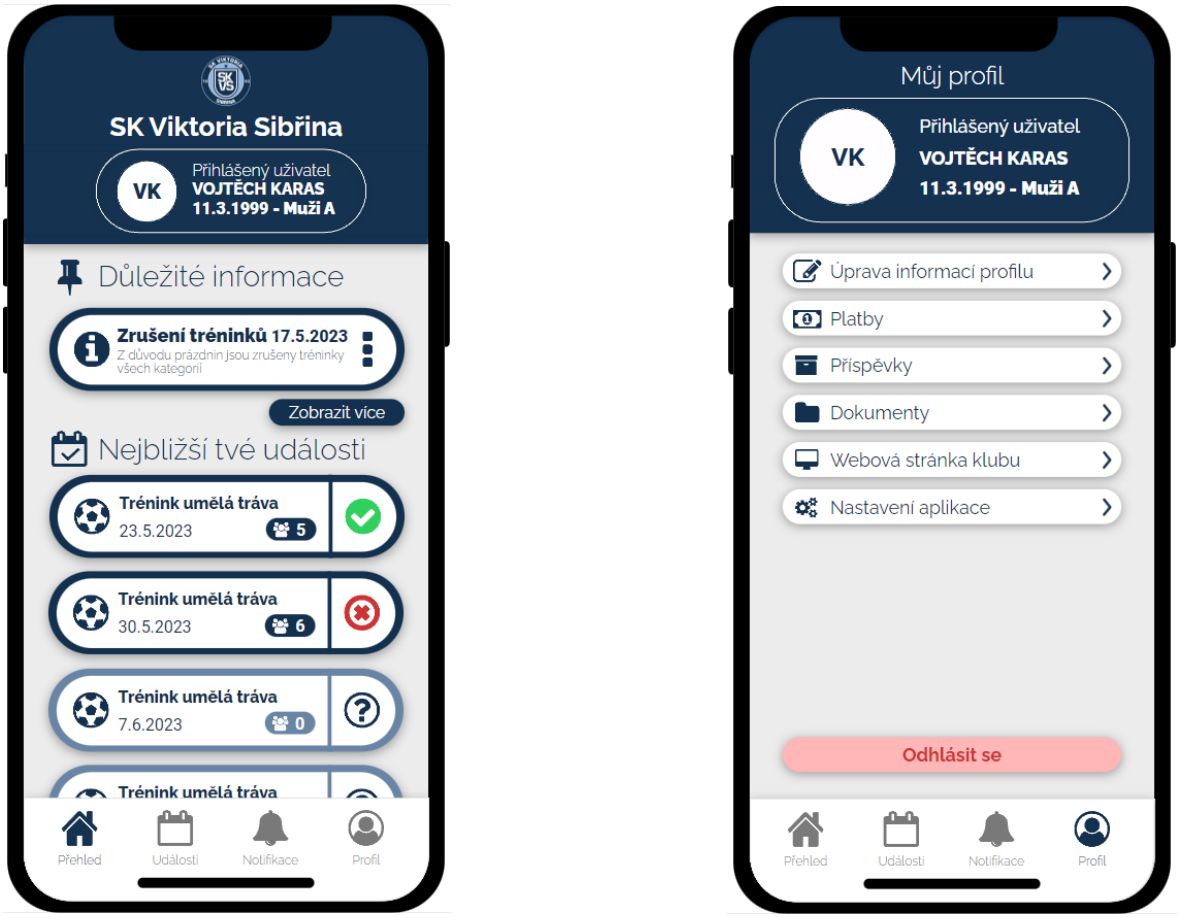
Návrh uživatelského rozhraní se vytváří pro zobrazení možné finální grafické podoby výsledného informačního systému. Díky tomuto návrhu je možné se zadavatelem konzultovat všechny detaily, které se týkají výsledné podoby systému, od rozvržení prvků a jednotlivých obrazovek po výběr barev.

V rámci této práce byl vytvořen interaktivní prototyp pro zachycení chování jednotlivých prvků a systému celkově při interakci s uživatelem. Dynamické chování systému na drátěných modelech nelze zachytit. Výsledný prototyp je tak dobrým nástrojem pro prezentaci výsledné podoby systému i s tím, jak se bude systém reálně chovat po implementaci všech požadovaných funkcionalit. Prototyp vytvořený v rámci této práce zobrazuje systém po implementaci všech funkcionalit vymezených podmnožinou UseCasů pro běžného uživatele viz Obrázek č. 34. Prototyp zobrazuje všechny funkce, které by ve finálním systému měl k dispozici běžný uživatel. Prototyp je navržen pouze jako mobilní aplikace, jelikož desktopová verze není v plánu v první fázi návrhu, který je řešen v této práci.

První obrazovkou je úvodní obrazovka, která slouží pro přihlášení do systému. Je možné se tedy přihlásit do systému uživatelským jménem a heslem. Pokud uživatel zapomene heslo je možné nechat si poslat nové heslo pomocí emailu. V poslední řadě je zde možnost registrace pro nové členy.

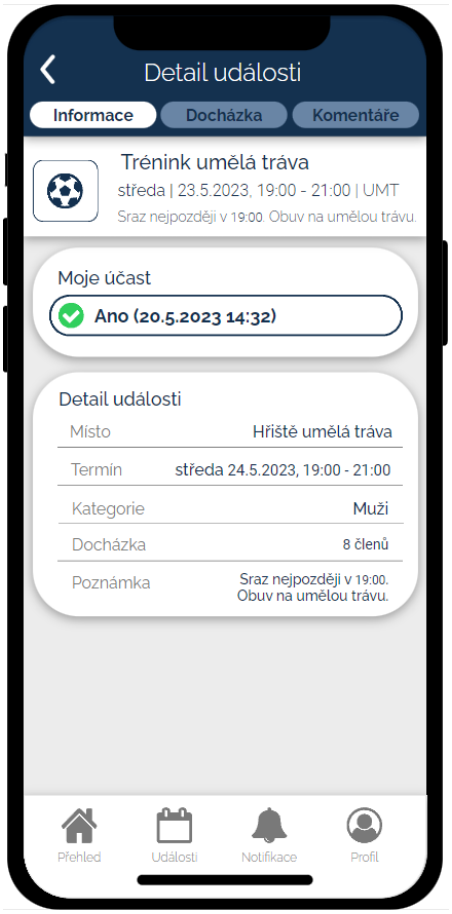
Druhá obrazovka se zobrazí po přihlášení do systému. Jedná se o hlavní obrazovku, která umožňuje veškerou interakci se systémem. Na hlavní obrazovce jsou vidět připnuté příspěvky, nejbližší plánované události a další. U konkrétních událostí může uživatel rychlou volbou potvrdit či odmítnout účast. Důraz je kladen na rychlost práce se systémem. Dále je zobrazena obrazovka profilu uživatele, ze které se uživatel dále může dostat na další obrazovky jakými jsou například platby, dokumenty, příspěvky či nastavení aplikace. Lze také upravit údaje uživatele.


**Obrázek č. 38 - Úvodní obrazovka**




**Obrázek č. 39 - Hlavní obrazovka a obrazovka Můj profil**


**Obrázek č. 40 - Obrazovka události**




**Obrázek č. 41 - Obrazovka detailu události**

Další obrazovkou je obrazovka s událostmi, která je jedna z nejdůležitějších. Události jsou rozděleny na již proběhlé a nadcházející pomocí záložek. Po kliknutí na danou událost se zobrazí detail události, kde může uživatel vidět podrobné informace, docházku k události a komentáře k události. V docházce je seznam nominovaných členů a jejich reakce na danou událost. Zda se zúčastní, nezúčastní či zatím nereagovali. V záložce komentáře lze přidávat komentáře k události např. pro dodatečné otázky.

Prototyp slouží jako doplněk návrhu systému. Nebudou tak popisovány všechny obrazovky. Všechny obrazovky prototypu jsou přiloženy v přílohách na konci práce, stejně tak jako samotný prototyp.

## 4.6 Testování IS

Testování informačního systému spočívá v ověření, že jsou implementovány všechny požadavky zadavatele. Ke každému funkčnímu požadavku by měl být přiřazen alespoň jeden UseCase, který ho pokrývá. Z ověření za pomoci mapování případů užití na funkční požadavky vyplynulo, že každý funkční požadavek je realizován alespoň jedním případem užití, tím pádem by systém po plné implementaci prošel testováním a splňuje tak všechny zadané funkční požadavky.

Ověření, že jsou implementovány všechny nefunkční požadavky, je možné pouze až po plné implementaci systému nebo částečně je lze vyčíst a ověřit v rámci diagramu nasazení. V aktuální fázi tak navrhovaný systém splňuje všechny požadavky zadavatele.

# 5 Výsledky a diskuse

V rámci této diplomové práce byl realizován pouze návrh informačního systému, který nebyl nijak implementován. Nelze tedy zhodnotit kvalitu implementovaného systému, k čemuž se využívají standardizované metodiky pro hodnocení kvality informačních systémů. Nefunkční požadavky na systém jsou tudíž měřitelné až po samotné implementaci systému.

## 5.1 Zhodnocení návrhu informačního systému

Návrh systému byl vytvořen, tak aby splňoval všechny předem stanovené požadavky zadavatele, funkční i nefunkční. Požadavky vzešly z rozhovorů se zadavatelem. Nejdříve byly požadavky podrobeny analýze a na jejím základě byl vytvořen model tříd, který popisuje strukturu vytvářeného systému. Model tříd byl popsán datovým slovníkem. Pro třídy, kde se jejich instance mohou nacházet ve více stavech, byly vytvořeny stavové diagramy, které popisují, za jakých podmínek mezi stavy přecházejí. Následně byl v rámci modelu interakcí vytvořen diagram případů užití. Z celkového diagramu případů užití byla vybrána podmnožina funkcionalit, kterou se detailněji zabývá zbytek práce. Tato podmnožina byla zvolena jakožto seznam základních funkcionalit, které by měly být implementovány v první fázi vývoje do verze 1.0. Jedná se o celkovou funkcionalitu pro běžného uživatele s nejnižšími přidělenými právy v systému.

K vybrané podmnožině případů užití byly vytvořeny odpovídající scénáře a všechny potřebné diagramy. Následovalo navržení uživatelského rozhraní a interaktivního prototypu. Ve vytvořeném prototypu je možné realizovat všechny případy užití z definované podmnožiny funkcionalit pro běžného uživatele. Prototyp byl vytvořen pro použití systému jakožto mobilní aplikace.

Ověření splnění všech definovaných požadavků bylo provedeno pomocí mapování případů užití na funkční požadavky. Požadavky, které spadaly do zmíněné podmnožiny byly otestovány i v rámci vytvořeného prototypu. Z ověření vyplývá, že pro první fázi vývoje je návrh systému úplný a dostatečný.

Systém také počítá i s budoucím rozšířením, což je jeden z nefunkčních požadavků na systém. Snadná rozšiřitelnost je zajištěna díky objektově orientovanému návrhu, díky kterému není problém systém rozšiřovat o další objekty a funkce. Výsledkem práce je návrh

IS, který má předpoklady ke zjednodušení administrativních procesů v rámci klubu a zásadní úspoře času všech účastníků díky jednodušší komunikaci v rámci systému.

## 5.2 Návrhy do budoucího vývoje

Pokud by byl projeven zájem ze strany zadavatele systém začít implementovat do svého prostředí, bylo by žádoucí navázat na výsledky této práce. V první řadě začít s vývojem první verze systému, pro kterou nabízí podklad právě tato práce a následně dokončit v rámci dalších iterací zbytek požadovaných funkcionalit tak, aby byly splněny všechny definované požadavky.

Po implementaci celého systému do mobilní aplikace by pro splnění i posledního nefunkčního požadavku „*N10 – Systém bude v budoucnu dostupný přes webové rozhraní.*“ bylo nutné začít s vývojem i pro verzi systému do webového prohlížeče či rovnou využít k vývoji systému již zmíněný cross-platform postup vývoje.

## 5.3 Porovnání s aktuálním řešením

Motivace pro tuto práci vycházela z vlastní zkušenosti, kdy v době začátku psaní této práce nebyl v rámci klubu využíván žádný informační systém ani jiné digitalizované řešení. V průběhu psaní této práce vedení klubu oslovilo i jednu z firem na trhu, která nabízí jedno z řešení pro správu sportovního klubu. Jelikož se jedná o firmu, která nabízí komplexní řešení pro klubovou platformu za velmi výhodných cenových podmínek, tak se klub rozhodl jít cestou měsíčního poplatku externí firmě. Jedná se o velkou firmu na trhu s těmito systémy, která má spoustu referencí, jen těžko tak lze nabídnout výhodnější, a hlavně rychlejší realizaci řešení. V rámci nabízeného balíčku služeb této firmy je jak systém pro správu sportovního klubu, který je přístupný jako mobilní aplikace i jako webová aplikace, tak i klubový web. Tím značně převyšuje možnosti systému navrhovaného v rámci této práce.

# 6 Závěr

Hlavním cílem této diplomové práce byla analýza a návrh informačního systému v UML, který bude poskytovat podporu pro správu sportovního klubu se zacílením na evidenci událostí a jejich účastníků.

V teoretické části práce byla na základě studia odborných informačních zdrojů zpracována teoretická východiska k tématu, kterým se zabývá tato práce. Těmi byly softwarové inženýrství, návrh informačních systémů, analýza požadavků, UML, systémové modelovaní a návrh uživatelského rozhraní. Nejdříve byl vysvětlen pojem softwarové inženýrství. Následně byla probrána problematika rozdělení informačních systémů, jejich životního cyklu a metodik využívaných k tvorbě informačních systémů. Poté byla podrobně rozebrána první fáze vývoje IS – analýza a sběr uživatelských požadavků. Poté byly popsány principy modelovacího jazyka UML a systémového modelování. Na závěr teoretické části bylo vysvětleno několik pojmů z oblasti tvorby uživatelských rozhraní a také bylo popsáno několik základních pravidel pro jejich tvorbu. Definování teoretických východisek bylo nutné pro správné uchopení praktické části práce, při jejíž tvorbě byly využity získané znalosti. Zároveň byl tímto splněn i jeden z dílčích cílů práce.

V praktické části práce byla provedena analýza a návrh informačního systému. Nejdříve byl proveden sběr a následná analýza uživatelských požadavků. Na základě analýzy požadavků byl vytvořen návrh systému v modelovacím jazyce UML. V rámci návrhu byl vytvořen model tříd, stavový model a model interakcí navrhovaného systému. Vytvoření těchto tří modelů bylo zároveň dílčím cílem práce.

Dalším krokem bylo vytvoření logických návrhů k jednotlivým obrazovkám systému. Na základě logických návrhů byl vytvořen plně funkční interaktivní prototyp, čímž byl splněn další dílčí cíl práce.

V závěru této práce byla popsána aktuální situace a aktuálně používané řešení v rámci zvoleného sportovního klubu, které bylo porovnáno s navrhovaným řešením. Tím byl splněn i poslední vedlejší cíl této práce. Nakonec byly formulovány doporučení pro budoucí vývoj systému. Autor také doporučil konkrétní implementaci fyzické architektury systému, která je vyobrazena diagramem nasazení.

Všechny cíle stanovené v zadání této práce tak lze dle autora považovat za splněné.

# 7 Seznam použitých zdrojů


AMBLER, Scott W., 2005. *The elements of UML 2.0 style*. Cambridg: Cambridge University Press. ISBN 05-216-1678-6.

ARLOW, Jim a Ila NEUSTADT, 2007. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press. ISBN 978-80-251-1503-9.

BOEHM, Barry, 2007. *Software engineering: Barry W. Boehm's lifetime contributions to software*. Wiley-IEEE Computer Society Press. ISBN 978-0-470-14873-0.

BOOCH, G., J. RUMBAUGH a I. JACOBSON, 1999. *Unified modeling language user guide*. Vyd. 1. Massachusetts: Addison-Wesley. ISBN 0-201-57168-4.

BRUCKNER, Tomáš, 2012. *Tvorba informačních systémů: principy, metodiky, architektury*. 1. vyd. Praha: Grada. Management v informační společnosti. ISBN 978-80-247-4153-6.

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

DEEP SOURCE, 2019. *The exponential cost of fixing bugs* [online]. In: . [cit. 2022-12-11]. Dostupné z: https://deepsource.io/blog/exponential-cost-of-fixing-bugs/

DOUCEK, Petr, 2004. *Řízení projektů informačních systémů*. 1. vyd. Praha: Professional Publishing. ISBN 978-80-86419-71-8.

EXTENDOFFICE.COM, 2021. Jak automaticky obarvit střídavé řádky / sloupce v aplikaci Excel?. In: *ExtendOffice.com* [online]. [cit. 2023-03-21]. Dostupné z: https://cs.extendoffice.com/documents/excel/3247-excel-automatically-color-alternating-rows-columns.html

FOWLER, Martin, 2009. *Destilované UML*. 1. vyd. Praha: Grada. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

HOADLEY, Paul, 2008. *Vodopadový model: Wikimedia Commons* [online]. In: . [cit. 2022-12-11].

CHLAPEK, Dušan, Václav ŘEPA a Iva STANOVSKÁ, 2011. *Analýza a návrh informačních systémů*. 1. vyd. Praha: Oeconomica. ISBN isbn978-80-245-1782-7.

IEEE XPLORE, 2002. *IEEE Standard Glossary of Software Engineering Terminology* [online]. In: . [cit. 2022-12-11]. 978-0-7381-0391-4. Dostupné z: https://ieeexplore.ieee.org/servlet/opac?punumber=2238

KADLEC, Václav, 2004. *Agilní programování: metodiky efektivního vývoje softwaru*. Vyd. 1. Brno: Computer Press. ISBN 8025103420.

KRÁL, Jaroslav, 1998. *Informační systémy: specifikace : realizace : provoz*. 1. vyd. Veletiny: Science. ISBN 80-86083-00-4.

LANO, Kevin, 2009. *UML 2 Semantics and Applications*. ISBN 9780470409084.

LIDWELL, William, Kritina HOLDEN a Jill BUTLER, 2011. *Univerzální principy designu: 125 způsobů jak zvýšit použitelnost a přitažlivost a ovlivnit vnímání designu*. Vyd. 1. Brno: Computer Press. ISBN 978-80-251-3540-2.

MICROSOFT, 2021. *Vytvoření diagramu nasazení UML* [online]. In: . [cit. 2022-12-11]. Dostupné z: https://support.microsoft.com/cs-cz/office/vytvořen%C3%AD-diagramu-nasazen%C3%AD-uml-ef282f3e-49a5-48f5-a6ae-69a6982a4543#OfficeVersion=Desktop

MIDDLEWARE.CZ, 2015. *VÝVOJOVÝ CYKLUS SOFTWARE: DOMINUJÍ AGILNÍ METODIKY TRHU?* [online]. In: . [cit. 2022-12-11]. Dostupné z: https://www.middleware.cz/projektove-rizeni0/24-pristupy-k-vyvoji-software-dominuje-agilni-vyvoj-trhu

MOLNÁR, Zdeněk, 1992. *Moderní metody řízení informačních systémů*. V Praze: Grada. Nestůjte za dveřmi (Grada). ISBN 80-85623-07-2.

MVČR, 2023. *Co je GDPR* [online]. In: . [cit. 2023-01-24]. Dostupné z: https://www.mvcr.cz/gdpr/clanek/co-je-gdpr.aspx

OMG, 2017. *ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION VERSION 2.5.1* [online]. In: . [cit. 2022-12-11]. Dostupné z: https://www.omg.org/spec/UML/

POŽÁR, Josef, 2010. *Manažerská informatika*. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk. ISBN 978-80-7380-276-9.

*Principy stojící za Agilním Manifestem.* [online], 2001. In: . [cit. 2022-12-11]. Dostupné z: https://agilemanifesto.org/iso/cs/principles.html

PROCHÁZKA, Jaroslav a Cyril KLIMEŠ, 2009. *SOFTWAROVÉ INŽENÝRSTVÍ. 2., aktualiz. a dopl. vyd* [online]. In: . Ostrava [cit. 2022-12-11]. Dostupné z: https://web.osu.cz/~Zacek/6swen/skriptaSWENG.pdf

ŘEPA, Václav, 2012. *Procesně řízená organizace*. 1. vyd. Praha: Grada. Management v informační společnosti. ISBN 9788024741284.

SIBERA-SERVIS.CZ, 2016. Co je barevný kruh pro kombinaci barev – systém RYB. In: *Sibera-servis.cz* [online]. [cit. 2023-03-21]. Dostupné z: https://www.sibera-servis.cz/co-je-barevny-kruh-pro-kombinaci-barev/

ŠOCHOVÁ, Zuzana a Eduard KUNCE, 2014. *Agilní metody řízení projektů*. 1. vyd. Brno: Computer Press. ISBN 978-80-251-4194-6.

TOPS INFOSOLUTIONS PVT LTD, 2020. *Methodology* [online]. In: . [cit. 2022-12-11]. Dostupné z: https://www.topsinfosolutions.com/wp-content/uploads/2017/05/Iterative-Incremental-Development.png.

VONDRÁK, Ivo, 2002. *Úvod do softwarového inženýrství* [online]. In: . Ostrava [cit. 2022-12-11]. Dostupné z: http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf

VRANA, Ivan, 2008. *Projektování informačních systémů s UML*. Vyd. 1. V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta. ISBN 978-80-213-1817-5.

VYMĚTAL, Dominik, 2009. *Informační systémy v podnicích: teorie a praxe projektování*. 1. vyd. Praha: Grada. Průvodce (Grada). ISBN 978-80-247-3046-2.

WIEGERS, Karl Eugene, 2008. *Požadavky na software*. Vyd. 1. Brno: Computer Press. ISBN 978-80-251-1877-1.

ZVESELA.CZ, 2010. *Postupy pro sběr požadavků* [online]. In: . [cit. 2022-12-11]. Dostupné z: http://zcu.arcao.com/_statnice_ing_fav/nis/wiki_zvesela/09.htm

# 8 Přílohy

## Příloha A – scénáře k případům užití zpracovaných v této práci

| **Případ užití** | **Registrace uživatele** |
|---|---|
| **ID** | UC1 |
| **Stručný popis** | Nový uživatel se registruje do systému. |
| **Hlavní aktéři** | Člen |
| **Vstupní podmínky** | Uživatel musí spustit systém. |
| **Hlavní scénář** | 1. Po stisknutí tlačítka Registrovat se na úvodní obrazovce se zobrazí obrazovka s formulářem pro registraci<br>2. Uživatel vyplní údaje dle vzoru a poté formulář stiskem tlačítka odešle.<br>3. Systém zkontroluje správnost uživatelských vstupů a uloží nového uživatele do databáze.<br>4. Systém zobrazí uživateli notifikací potvrzení o zadaní registrace nového uživatele. |
| **Výstupní podmínky** | Je zobrazeno potvrzení o registraci nového uživatele. |
| **Alternativní scénáře** | Uživatel nesprávně vyplnil údaje – registraci je nutné opakovat. |

| **Případ užití** | **Editace účtu** |
|---|---|
| **ID** | UC19 |
| **Stručný popis** | Uživatel edituje údaje na svém profilu. |
| **Hlavní aktéři** | Člen |
| **Vstupní podmínky** | Uživatel musí být přihlášen. |
| **Hlavní scénář** | 1. Po stisknutí tlačítka Úprava informací profilu se zobrazí obrazovka s formulářem pro úpravu údajů.<br>2. Po přepsání hodnoty v daném poli se potvrdí změna tlačítkem uložit.<br>3. Systém zobrazí potvrzení o uložení změny údajů.<br>4. Systém přesune uživatele zpět na předchozí obrazovku. |
| **Výstupní podmínky** | Je zobrazeno potvrzení o uložení změny údajů. |
| **Alternativní scénáře** | Uživatel nesprávně vyplnil údaje – změny se neuložili.<br>Uživatel zrušil proces změny údajů. |

| **Případ užití** | **Zapomenuté heslo** |
|---|---|
| **ID** | UC18 |
| **Stručný popis** | Uživatel zapomněl své heslo a žádá o zaslání nového. |
| **Hlavní aktéři** | Člen |
| **Vstupní podmínky** | Uživatel musí být registrovaným uživatelem v systému. |
| **Hlavní scénář** | 1. Po stisknutí tlačítka Zapomenuté heslo se zobrazí obrazovka s formulářem pro vyplnění emailu, ke kterému je požadováno zaslání nového hesla.<br>2. Po vyplnění emailu a potvrzení odeslání nového hesla je zobrazeno potvrzení o zaslání žádosti.<br>3. Systém přesune uživatele zpět na předchozí obrazovku. |
| **Výstupní podmínky** | Je potvrzeno zaslání žádosti o nové heslo do emailu. |
| **Alternativní scénáře** | Uživatel zrušil proces k zaslání nového hesla. |

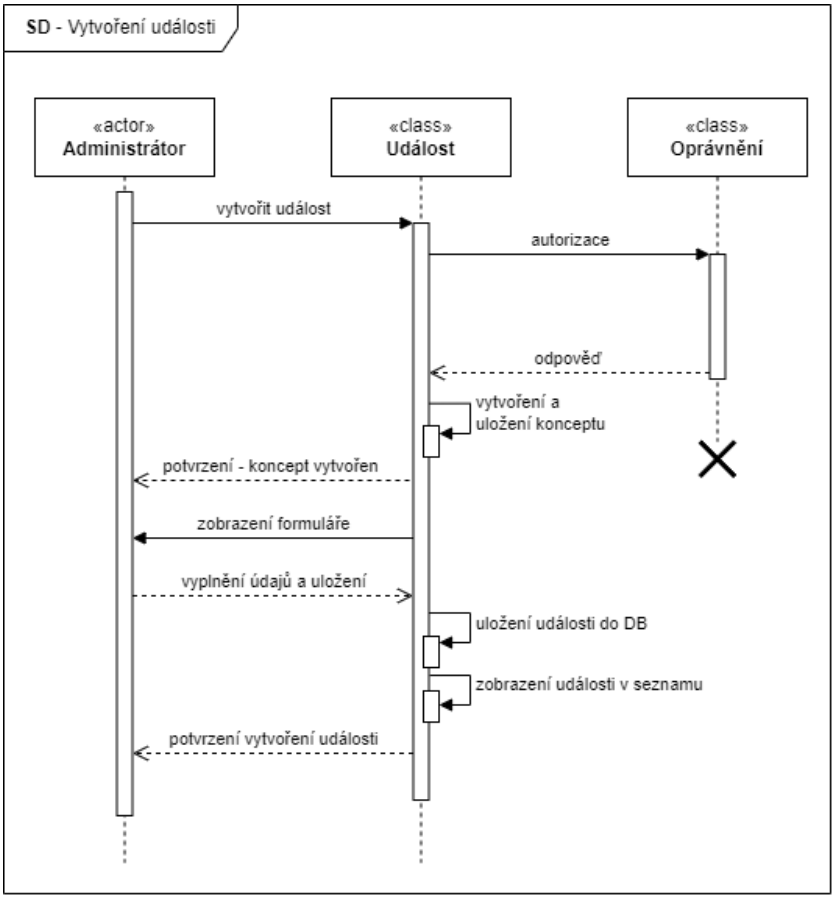
| Případ užití | Změna hesla |
|---|---|
| **ID** | UC15 |
| **Stručný popis** | Uživatel mění své přihlašovací heslo. |
| **Hlavní aktéři** | Člen |
| **Vstupní podmínky** | Uživatel musí být přihlášen. |
| **Hlavní scénář** | 1. Po stisknutí tlačítka Úprava informací profilu se zobrazí obrazovka s formulářem pro úpravu údajů.<br>2. Po napsání aktuálního hesla, nového hesla a potvrzení nového hesla a následném stisknutí tlačítka pro uložení, systém uloží do databáze hash nového hesla uživatele.<br>3. Systém zobrazí potvrzení o uložení změny hesla.<br>4. Systém přesune uživatele zpět na předchozí obrazovku. |
| **Výstupní podmínky** | Je potvrzena změna aktuálního hesla uživatele. |
| **Alternativní scénáře** | Uživatel zadal nesprávné aktuální heslo.<br>Uživatel zadal nové heslo, které neodpovídá požadavkům.<br>Zadané nové heslo nesouhlasí s potvrzením nového hesla. |

| Případ užití | Přidání komentáře |
|---|---|
| **ID** | UC10 |
| **Stručný popis** | Uživatel přidává komentář k události či příspěvku. |
| **Hlavní aktéři** | Člen |
| **Vstupní podmínky** | Uživatel musí být přihlášen.<br>Událost či příspěvek musí být veřejný. |
| **Hlavní scénář** | 1. Po zobrazení záložky komentáře u zveřejněné události či příspěvku je uživateli umožněno přidat nový komentář.<br>2. Po napsání textu komentáře do textboxu a následném stisknutí tlačítka Okomentovat je zveřejněn komentář.<br>3. Komentář u dané události či příspěvku je uložen do databáze.<br>4. Komentář je viditelný i pro ostatní uživatele systému. |
| **Výstupní podmínky** | Komentář je publikován u události či příspěvku. Je viditelný pro další uživatele systému. |
| **Alternativní scénáře** | Uživatel zrušil proces přidání komentáře.<br>Komentář má moc znaků k publikování – systémové omezení. |

| Případ užití | Reakce na docházku |
|---|---|
| **ID** | UC13 |
| **Stručný popis** | Uživatel potvrzuje nebo ruší účast u dané události. |
| **Hlavní aktéři** | Člen |
| **Vstupní podmínky** | Uživatel musí být přihlášen.<br>Uživatel musí být nominován na událost. |
| **Hlavní scénář** | 1. Po stisknutí symbolu otazníku u události zavolá operaci pro reakci na událost.<br>2. Systém zobrazí formulář pro potvrzení či odmítnutí účasti u události.<br>3. Po uživatelově volbě se zvolená odpověď uloží v jeho docházce u události<br>4. Systém dle volby zobrazí příslušnou odpověď se symbolem u události u daného uživatele. |
| **Výstupní podmínky** | Je zaevidována reakce uživatele na docházku – možnost potvrzení nebo odmítnutí. |
| **Alternativní scénáře** | Uživatel zrušil proces Reakce na docházku.<br>Událost byla během procesu smazána. |

# Příloha B – dodatečné diagramy

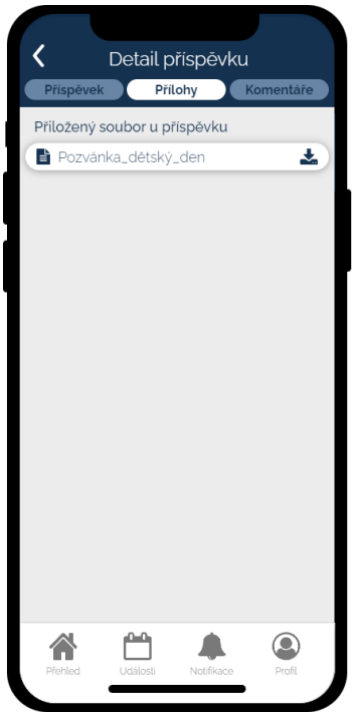
| Případ užití | Vytvoření události |
|---|---|
| **ID** | UC21 |
| **Stručný popis** | Administrátor vytváří novou událost v systému. |
| **Hlavní aktéři** | Administrátor |
| **Vstupní podmínky** | Administrátor musí být přihlášen |
| **Hlavní scénář** | 1. Po stisknutí tlačítka pro vytvoření události, systém zavolá operaci pro vytvoření nové události.<br>2. Systém ověří, že má uživatel přidělená dostatečná práva.<br>3. Systém zobrazí formulář pro vyplnění požadovaných údajů k události.<br>4. Administrátor vyplní systémem požadovaná pole a uloží událost.<br>5. Systém uloží událost do databáze a zobrazí ji v seznamu událostí s vyplněnými údaji a jejím aktuálním stavem.<br>6. Systém zobrazí uživateli potvrzení o správném vytvoření události. |
| **Výstupní podmínky** | Vytvoření události je potvrzeno a uloženo do systému. |
| **Alternativní scénáře** | Chyba při uložení nové události.<br>Chyba aplikace kvůli neočekávanému ukončení aplikace.<br>Smazání rozpracovaného konceptu nové události. |




**Sekvenční diagram a scénář pro vytvoření události – administrátor**

# Příloha C – UI návrh obrazovek

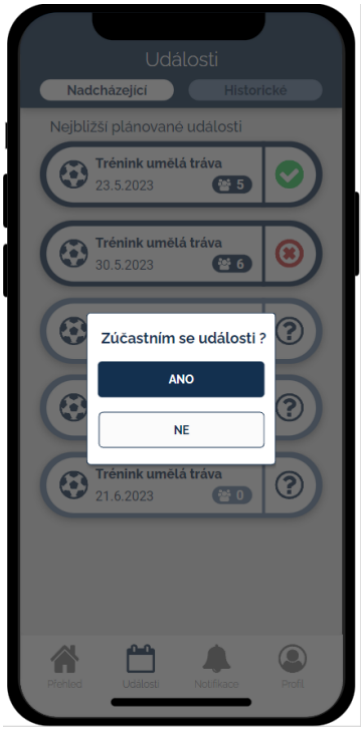


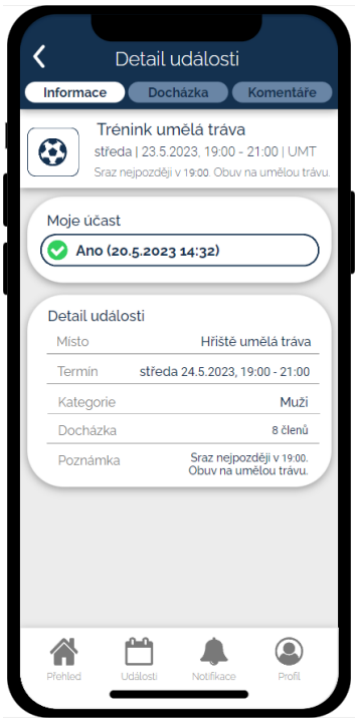



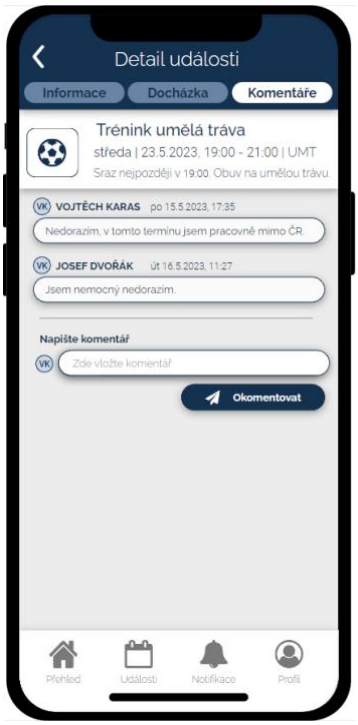


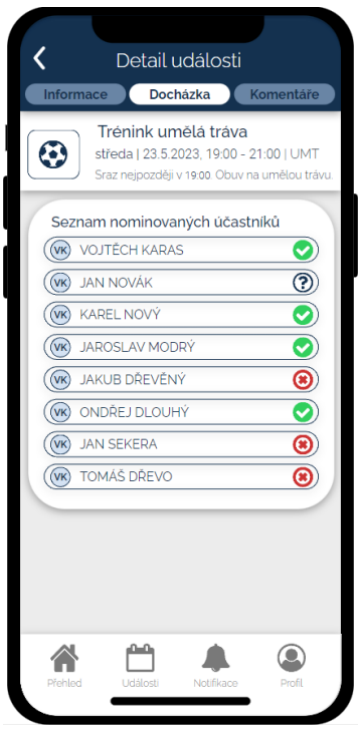

Události
Nadcházející
Historické
Nejbližší plánované události
Trénink umělá tráva
23.5.2023
4
Trénink umělá tráva
30.5.2023
6
Trénink umělá tráva
7.6.2023
0
Trénink umělá tráva
14.6.2023
0
Trénink umělá tráva
21.6.2023
0
Přehled
Události
Notifikace
Profil

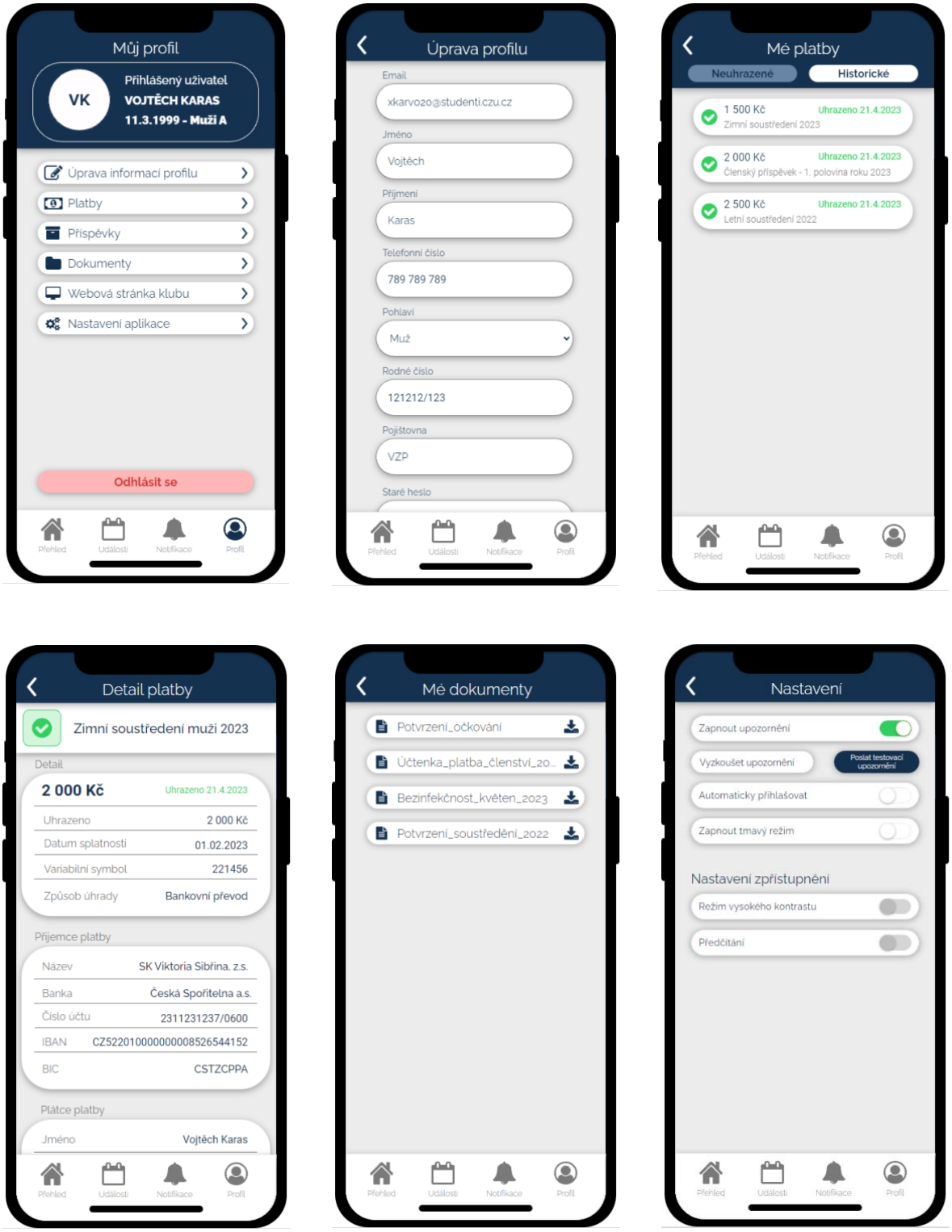

Události
Nadcházející
Historické
Historické již proběhlé události
Trénink umělá tráva
14.4.2023
8
Trénink umělá tráva
21.4.2023
11
Přehled
Události
Notifikace
Profil


Události
Nadcházející
Historické
Nejbližší plánované události
Trénink umělá tráva
23.5.2023
5
Trénink umělá tráva
30.5.2023
6
Zúčastním se události ?
ANO
NE
Trénink umělá tráva
21.6.2023
0
Přehled
Události
Notifikace
Profil


Detail události
Informace
Docházka
Komentáře
Trénink umělá tráva
středa | 23.5.2023, 19:00 - 21:00 | UMT
Sraz nejpozději v 19:00. Obuv na umělou trávu.
Moje účast
Ano (20.5.2023 14:32)
Detail události
Místo
Hřiště umělá tráva
Termín
středa 24.5.2023, 19:00 - 21:00
Kategorie
Muži
Docházka
8 členů
Poznámka
Sraz nejpozději v 19:00.
Obuv na umělou trávu.
Přehled
Události
Notifikace
Profil


Detail události
Informace
Docházka
Komentáře
Trénink umělá tráva
středa | 23.5.2023, 19:00 - 21:00 | UMT
Sraz nejpozději v 19:00. Obuv na umělou trávu.
VOJTĚCH KARAS po 15.5.2023, 17:35
Nedorazím, v tomto termínu jsem pracovně mimo ČR.
JOSEF DVOŘÁK út 16.5.2023, 11:27
Jsem nemocný nedorazím.
Napište komentář
Zde vložte komentář
Okomentovat
Přehled
Události
Notifikace
Profil


Detail události
Informace
Docházka
Komentáře
Trénink umělá tráva
středa | 23.5.2023, 19:00 - 21:00 | UMT
Sraz nejpozději v 19:00. Obuv na umělou trávu.
Seznam nominovaných účastníků
VOJTĚCH KARAS
JAN NOVÁK
KAREL NOVÝ
JAROSLAV MODRÝ
JAKUB DŘEVĚNÝ
ONDŘEJ DLOUHÝ
JAN SEKERA
TOMÁŠ DŘEVO
Přehled
Události
Notifikace
Profil




**Wireframy pro mobilní aplikaci**