VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Samuel Janek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INFORMAČNÍ SYSTÉM IDENTIFIKACE BEZPEČNOSTNÍCH SYSTÉMŮ PODLE ČSN EN ISO 12100:2011

INFORMATION SYSTEM FOR ASSESSMENT OF SAFETY SYSTEMS ACCORDING TO ČSN EN ISO 12100:2011

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Samuel Janek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radovan Holek, CSc.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Samuel Janek ***ID:*** 211150
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Informační systém identifikace bezpečnostních systémů podle ČSN EN ISO 12100:2011

POKYNY PRO VYPRACOVÁNÍ:

1. Zpracujte rešerši týkající se posouzení rizik strojních zařízení dle podle ČSN EN ISO 12100:2011.
2. Navrhněte webový informační systém pro sběr bezpečnostních parametrů pro další vyhodnocování bezpečnostních systémů. Systém bude vytvořený v C# a databázi MSSQL.
3. Navrhněte a popište řešení back-end a front-end částí informačního systému. Při realizaci front-end části se zaměřte na ochranu před možnými útoky na informační systémy, na ochranu dat i realizovaných formulářů před neoprávněnými přístupy. Zvažte možnosti řízení vlastností front-end aplikace konfiguračními daty, uloženými v back-end části aplikace.
4. Realizujte funkční webovou aplikaci.
5. Ověřte a vyhodnoťte realizovaný informační systém. Popište procesy pro vyhodnocení bezpečnosti konkrétního zařízení prostřednictvím Vámi navrženého informačním systému.

DOPORUČENÁ LITERATURA:

Safebook 4. Principles of Machine Safety – Legislation, Theory and Practice. Rockwell Automation. 2011. 150 s.

RÁČEK, J. Strukturovaná analýza systémů. 1.vyd. Brno Masarykova univerzita. 2006. 103 s. ISBN 80-2010-41-0-0

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Radovan Holek, CSc.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

Abstrakt

Táto práca sa zaoberá návrhom informačného systému, ktorý bude uľahčovať zber parametrov a ich následné použitie pri vyhodnocovaní bezpečnostných systémov. Pri návrhu budeme vychádzať hlavne z noriem EN ISO 13849 a EN 62061. Výsledná aplikácia nebude zahŕňať aj počiatočnú analýzu rizík podľa normy EN ISO 12100, ale primárne sa bude venovať návrhu a vyhodnocovaniu bezpečnostných funkcií pre strojné zariadenia.

Kľúčové slová

Analýza rizík, bezpečnosť strojných zariadení, bezpečnostná funkcia, Performance Level, Safety Integrity Level, informačný systém, webová aplikácia, ASP.NET

Abstract

This work deals with the design of an information system that will facilitate the collection of parameters and their further use in the evaluation of security systems. The design will be based mainly on the standards EN ISO 13849 and EN 62061. The resulting application will not contain an initial risk analysis according to the standard EN ISO 12100, but will primarily focus on safety function design and evaluation.

Keywords

Risk analysis, machine safety, safety function, Performance Level, Safety Integrity Level, information system, web application, ASP.NET

Bibliografická citácia

JANEK, Samuel. Informační systém identifikace bezpečnostních systémů podle ČSN EN ISO 12100:2011 [online]. Brno, 2021 [cit. 2021-05-15]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/134855. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Ing. Radovan Holek, CSc.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Samuel Janek*
VUT ID studenta:	*211150*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2020/21*
Téma závěrečné práce:	*Informační systém identifikace bezpečnostních systémů podle ČSN EN ISO 12100:2011*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 20. května 2020 ----------

podpis autora

Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu bakalárskej práce Ing. Radovanovi Holkovi, CSc. a Ing. Radkovi Štohlovi, PhD. za ústretový prístup, trpezlivosť a hlavne za čas, ktorý mi pri tvorbe tejto práce obaja venovali.

V Brně dne: 20. května 2020 ----------

podpis autora

Obsah

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Bezpečnosť strojných zariadení patrí v dnešnej dobe medzi najdôležitejšie aspekty pri navrhovaní a následnom používaní stroja v bežnej prevádzke. Ide o komplexnú problematiku, ktorá sa v praxi týka v podstate každej časti stroja, ktorý je uvedený do prevádzky alebo na trh.

Účelom tejto bezpečnosti je dosiahnuť čo najnižšie možné riziko zranenia pri používaní stroja v bežnom, ale aj v poruchovom stave. Toto je väčšinou dosiahnuté tým, že v miestach priameho prístupu k pohyblivým alebo nebezpečným častiam stroja, sa tomuto prístupu úplne zamedzí, alebo sa riziko nebezpečenstva v týchto oblastiach dostatočne zníži bezpečnostnými funkciami. Zlyhanie bezpečnosti má často fatálne následky, preto by mal jej návrh robiť osoba s dostatočnými skúsenosťami a s adekvátnym vzdelaním v danej oblasti.

Výber správnej bezpečnostnej funkcie a ich subsystémov je pomerne zložitá záležitosť, ktorá vyžaduje vykonávanie výpočtov, ale hlavne dobrú znalosť noriem, ktoré sa danej problematiky týkajú. V rámci tejto bakalárskej práce, sa budeme zaoberať návrhom informačného systému, ktorý by mal túto prácu uľahčiť.

Základom pre bezpečnosť strojných zariadení sú normy EN ISO 12100, EN ISO 13849, EN 62061. Práve z týchto noriem, budeme pri návrhu vyššie spomenutého informačného systému vychádzať.

V prvej časti tejto práce sa budem zaoberať stručným opisom používaných noriem, ktorých znalosť je pre ďalší postup kľúčová. Zoznámime sa s najdôležitejšími parametrami, ktoré rozhodujú o výslednej bezpečnosti.

V druhej časti sa už budeme venovať samotnému návrhu informačného systému a tvorbe testovacej webovej aplikácie. Na úvod si priblížime proces zberu a vyhodnocovania bezpečnostných parametrov. Potom sa budeme venovať návrhu dátového modelu a následne sa pozrieme na návrh a tvorbu back-endu a front-endu. V rámci tejto kapitoly si tiež zhrnieme základné bezpečnostné prvky vytvorenej aplikácie.

V závere tejto práce si zhrnieme dosiahnuté výsledky a pobavíme sa možných vylepšeniach, ktoré by mohla aplikácia v budúcnosti obsahovať.

1 ÚVOD DO BEZPEČNOSTI STROJNÝCH ZARIADENÍ

Normy harmonizované so strojárenskou smernicou delíme do troch skupín [podľa STN EN ISO 12100:2011]:

- **Normy typu A** (základné bezpečnostné normy) – obsahujú základnú terminológiu a zásady konštruovania strojných zariadený.
- **Normy typu B** (skupinové bezpečnostné normy) **–** zaoberajú sa jedným bezpečnostným hľadiskom, alebo jedným bezpečnostným zariadením.
    - **Normy typu B1** – obsahujú jednotlivé bezpečnostné hľadiská.
    - **Normy typu B2** – obsahujú bezpečnostné zariadenia.
- **Normy typu C** (bezpečnostné normy na stroje) – obsahujú podrobné bezpečnostné požiadavky na konkrétnu skupinu strojov.

1.1 Norma EN ISO 12100

Táto norma je základom pre ďalšie normy typu B a C. Obsahuje základné informácie a terminológiu týkajúcu sa bezpečnosti strojných zariadení. Opisuje všeobecné zásady konštruovania strojov, ktoré uľahčujú prácu konštruktérom pri návrhu strojných zariadení a zároveň zaisťujú, že navrhnutý stroj je v súlade s medzinárodnými štandardami pre bezpečnosť.

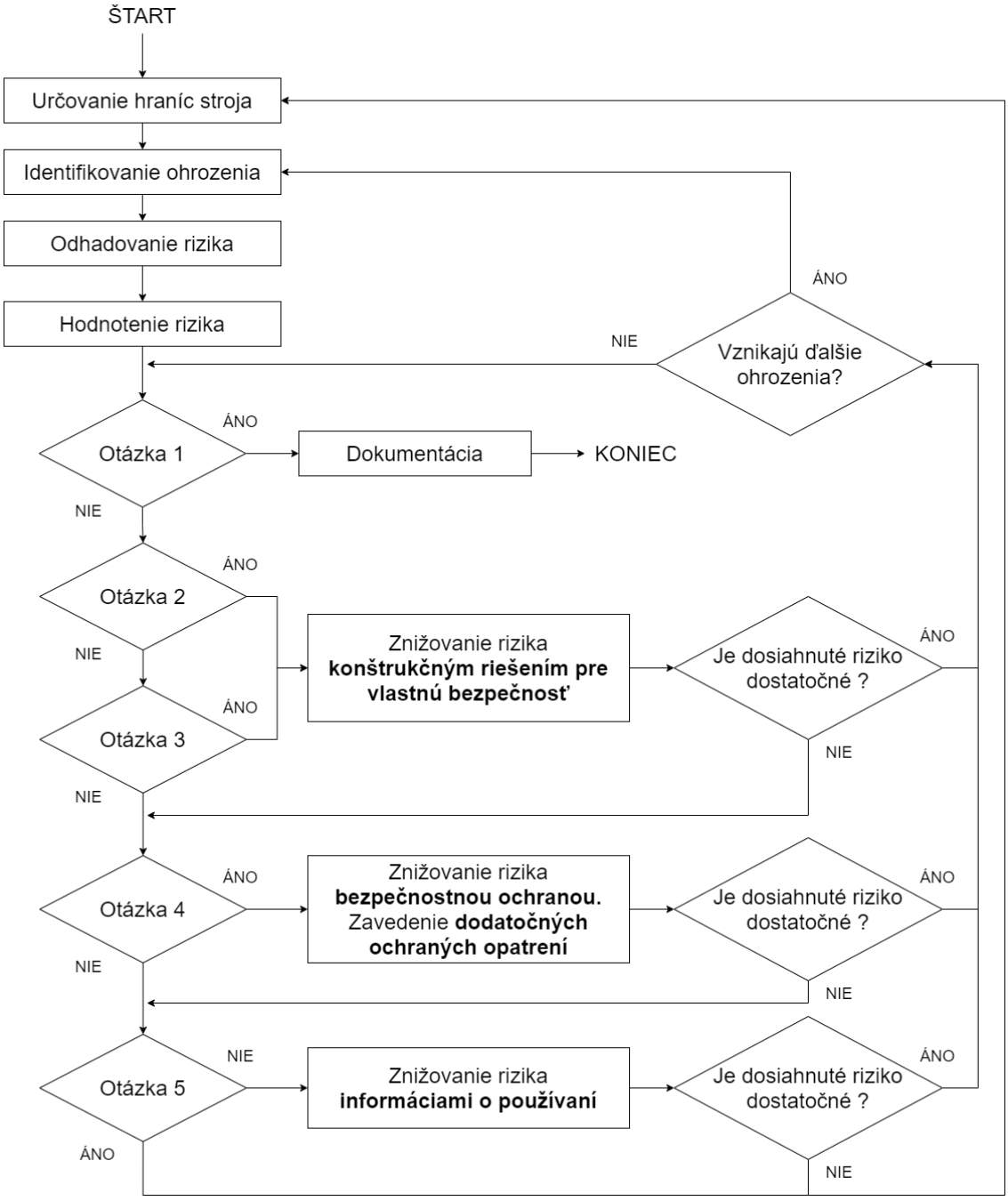
Ďalej sa norma zaoberá stratégiou posudzovania a znižovania rizika. Túto stratégiu je možné rozčleniť do niekoľkých krokov [podľa STN EN ISO 12100:2011]:

1. Vymedzenie hraníc stroja, čo zahŕňa predpokladané používanie stroja a jeho odôvodnene predvídateľné nesprávne používanie.
2. Identifikovať ohrozenia a súvisiace nebezpečné situácie.
3. Odhadnúť riziko pre každé identifikované ohrozenie a pre každú nebezpečnú situáciu.
4. Vyhodnotiť riziko a rozhodnúť o tom, či je potrebné znížiť riziko.
5. Odstrániť ohrozenie (nebezpečenstvo) alebo znížiť riziko súvisiace s ohrozením, ochrannými opatreniami.

Vyššie uvedený postup je graficky znázornený na obrázku číslo 1. Na znižovanie rizika (bod e v stratégii) norma uvádza tzv. trojstupňovú metódu:

- Krok 1: Konštrukčné opatrenia pre vlastnú bezpečnosť.
- Krok 2: Bezpečnostná ochrana alebo doplnkové ochranné opatrenia.
- Krok 3: Informácie o používaní.

Ak sa opatrenia na znižovanie rizika týkajú ovládacieho systému, prichádzajú na rad normy EN ISO 13849 a EN 62061, ktoré sa zaoberajú elektrickými bezpečnostnými riadiacimi systémami. To ktorú normu použiť, záleží na zložitosti a druhu bezpečnostnej funkcie na zníženie rizika.



OTÁZKA ČÍSLO 1: Znížilo sa riziko dostatočne ?

OTÁZKA ČÍSLO 2: Môže sa riziko odstrániť ?

OTÁZKA ČÍSLO 3: Môže sa riziko odstrániť konštrukčným riešením vlastnej bezpečnosti ?

OTÁZKA ČÍSLO 4: Môže sa riziko odstrániť ochrannými krytmi, ochrannými zariadeniami ?

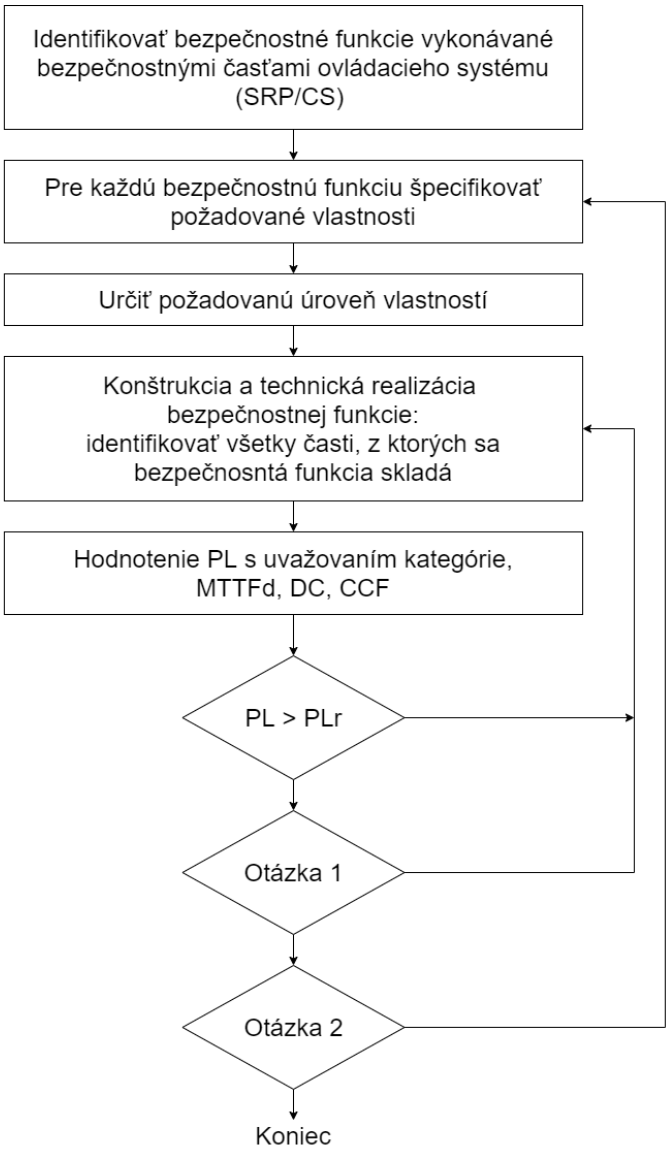
OTÁZKA ČÍSLO 5: Môžu sa znovu určiť hranice ?

Obr. 1: Schematické znázornenie procesu znižovania rizika
[podľa STN EN ISO 12100:2011]

1.2 Norma EN ISO 13849-1

Táto norma obsahuje požiadavky týkajúce sa návrhu a integrácie bezpečnostných častí riadiacich systémov vrátanie niektorých aspektov softwaru [1]. Je to norma typu B1 a je aplikovateľná na pneumatické, hydraulické, mechanické ale aj elektronické systémy.

Aby sme túto normu mohli vôbec v praxi použiť, je nutné mať najskôr urobenú analýzu rizík podľa normy ISO 12100, konkrétne podľa postupu, ktorý je znázornený na obrázku 1. Až potom môžeme pristúpiť na iteratívny proces určovania úrovne vlastností (PL), ktorý je znázornený na obrázku číslo 2.



OTÁZKA ČÍSLO 1: Sú splnené všetky požiadavky normy ?

OTÁZKA ČÍSLO 2: Boli analyzované všetky bezpečnostné funkcie ?

Obr. 2: Iteratívny proces pre konštruovanie bezpečnostných častí ovládacieho systému (SRP/CS) [podľa ČSN EN ISO 13849-1:2006]

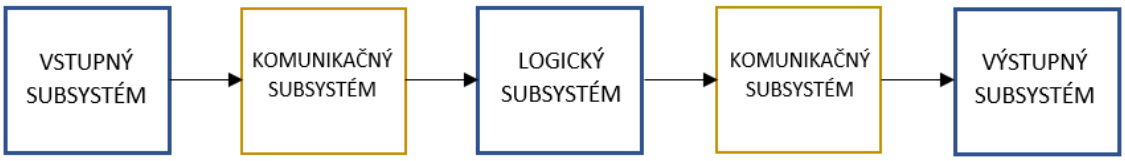
Úrovne vlastností opisujú schopnosť daného systému vykonávať bezpečnostnú funkciu. Na určenie výslednej úrovne vlastností v koku 5 procesu na obrázku 2 je v norme presne definovaná tabuľka, ktorá je uvedená v prílohe A.1. Presný postup, ktorým sa dostaneme až k výslednej PL si dopodrobna rozoberieme ešte v kapitole 2.1 – Proces zberu parametrov a ich vyhodnotenie. Vstupom do spomínanej tabuľky na určenie PL sú parametre, ktoré sa vyskytujú spolu s ďalším parametrami aj v obrázku 2. Keďže navrhovaná aplikácia bude s týmito parametrami priamo pracovať, je vhodné sa s nimi bližšie zoznámiť.

1.2.1 Minimálna požadovaná úroveň vlastností

Tento parameter sa určuje pre každú bezpečnostnú funkciu a poskytuje nám údaj o minimálnej požadovanej úrovni vlastností (PL_r). Čím väčší je požadovaný rozsah zníženia rizika bezpečnostnými časťami ovládacieho systému, tým vyššie musí byť PL_r [podľa ČSN EN ISO 13849-1:2006]. Hodnota tohto parametru sa určuje pomocou metódy uvedenej v prílohe číslo A.2.

1.2.2 Kategória zapojenia

Na to, aby sme sa mohli bližšie zoznámiť s týmto parametrom, si musíme najskôr zadefinovať základnú blokovú štruktúru bezpečnostných funkcií. Každá bezpečnostná funkcia sa skladá z troch alebo piatich hlavných častí (subsystémov), tak ako znázorňuje obrázok 3. Komunikačné subsystémy (na obrázku znázornené žltou farbou) sú nepovinné a ich prítomnosť v bezpečnostnej funkcii je daná tým, či stroj podporuje nejaký druh komunikácie medzi zariadeniami.



Obr. 3: Bloková štruktúra bezpečnostnej funkcie

V tomto bode je potrebné poukázať na fakt, že kategórie zapojenia môžu byť použité pre celý systém (celú bezpečnostnú funkciu), alebo pre jednotlivé subsystémy. V našom prípade budeme tento parameter používať primárne pre subsystémy.

Jednotlivé kategórie stanovujú požadované chovanie bezpečnostných častí ovládacieho systému (SRP/CS) s ohľadom na ich odolnosť proti rôznym poruchám [podľa ČSN EN ISO 13849-1:2006]. Zjednodušene sa dá povedať, že popisujú spôsob, akým je subsystém pripojený ku zvyšku systému. Každá bezpečnostná časť ovládacieho systému musí odpovedať požiadavkám relevantnej kategórie. Jedným z najdôležitejších parametrov pre kategórie je počet kanálov. Tento údaj priamo ovplyvňuje odolnosť proti zlyhaniu so spoločnou príčinou. Ako budeme vidieť v neskorších kapitolách, tento údaj nás bude zaujímať aj pri výbere vhodných logických subsystémov.

1.2.3 Stredná doba do nebezpečnej poruchy

Parameter má skratku $MTTF_D$ (mean time to dangerous failure) a charakterizuje strednú dobu do nebezpečnej poruchy, ktorá môže spôsobiť zlyhanie bezpečnostnej funkcie. Vyjadruje sa v rokoch.

$MTTF_D$ sa určuje pre celú bezpečnostnú funkciu, ale aj pre jednotlivé subsystémy. Na určenie $MTTF_D$ u subsystémov je nutné najskôr určiť túto hodnotu pre každý kanál daného subsystému. U väčšiny dvojkanálových subsystémov je v praxi hodnota $MTTF_D$ obi dvoch kanálov rovnaká.

Tab. 1: Rozdelenie $MTTF_D$ do troch skupín podľa normy EN ISO 13849 [podľa ČSN EN ISO 13849-1:2006]

Označenie úrovne $MTTF_D$ pre každý kanál	Rozsah $MTTF_D$ pre každý kanál
nízka	3 roky ≤ $MTTF_D$ < 10 rokov
stredná	10 rokov ≤ $MTTF_D$ < 30 rokov
dlhá	30 roky ≤ $MTTF_D$ < 100 rokov

1.2.4 Diagnostické pokrytie

Niektoré z vyššie uvedených kategórií zapojenia potrebujú pre svoju správnu činnosť diagnostické testovanie pre kontrolu funkčnosti bezpečnostnej funkcie. Pre popis účinnosti tohto diagnostického testovania sa používa termín diagnostické pokrytie (DC).

Hodnota tohto parametru je vyjadrená v percentách a môže dosahovať maximálne 99%. DC je vyjadrené nasledujúcim vzorcom [2]:

$$DC = \frac{\lambda_{DD}}{\lambda_{DTotal}} \tag{1.1}$$

$\lambda_{DD} = pravdepodobnosť\ detekovaných\ nebezpečných\ zlyhaní$
$\lambda_{DTotal} = pravdepodobnosť\ všetkých\ nebezpečných\ zlyhaní$

Tab. 2: Rozdelenie diagnostického pokrytia podľa normy EN ISO 13849 [podľa ČSN EN ISO 13849-1:2006]

Označenie	Rozsah
žiadne	DC < 60%
nízke	60% ≤ DC < 90%
stredné	90% ≤ DC < 99%
vysoké	99% ≤ DC

1.2.5 Opatrenia proti zlyhaniu so spoločnou príčinou

Ďalej budeme pre tento parameter používať jeho skratku CCF (common cause failure). CCF nám dáva údaj o tom, či je systém dostatočne zabezpečený proti zlyhaniu so spoločnou príčinou, ktoré môže nastať, keď zlyhanie jednej komponenty napr. v subsystéme bezpečnostnej funkcie, povedie k zlyhaniu ďalších komponentov.

Stupeň náchylnosti ku zlyhaniu so spoločnou príčinou vyjadruje beta (β) faktor. CCF je veľmi podstatný parameter, ktorý treba zohľadňovať pri návrhu a konštrukcii bezpečnostných funkcií. Na uľahčenie jeho používania norma špecifikuje tabuľku, ktorá obsahuje základné spôsoby, ako predchádzať zlyhaniam so spoločnou príčinou. Každá položka tabuľky má svoje bodové ohodnotenie.

Tab. 3: Základné opatrenia proti CCF [3]

Opatrenie proti CCF	Bodové hodnotenie
Separácia/segregácia	15
Rôznorodosť	20
Návrh/aplikácie/skúsenosti	20
Posúdenie/analýza	5
Kompetencia/školenie	5
Okolité prostredie	35

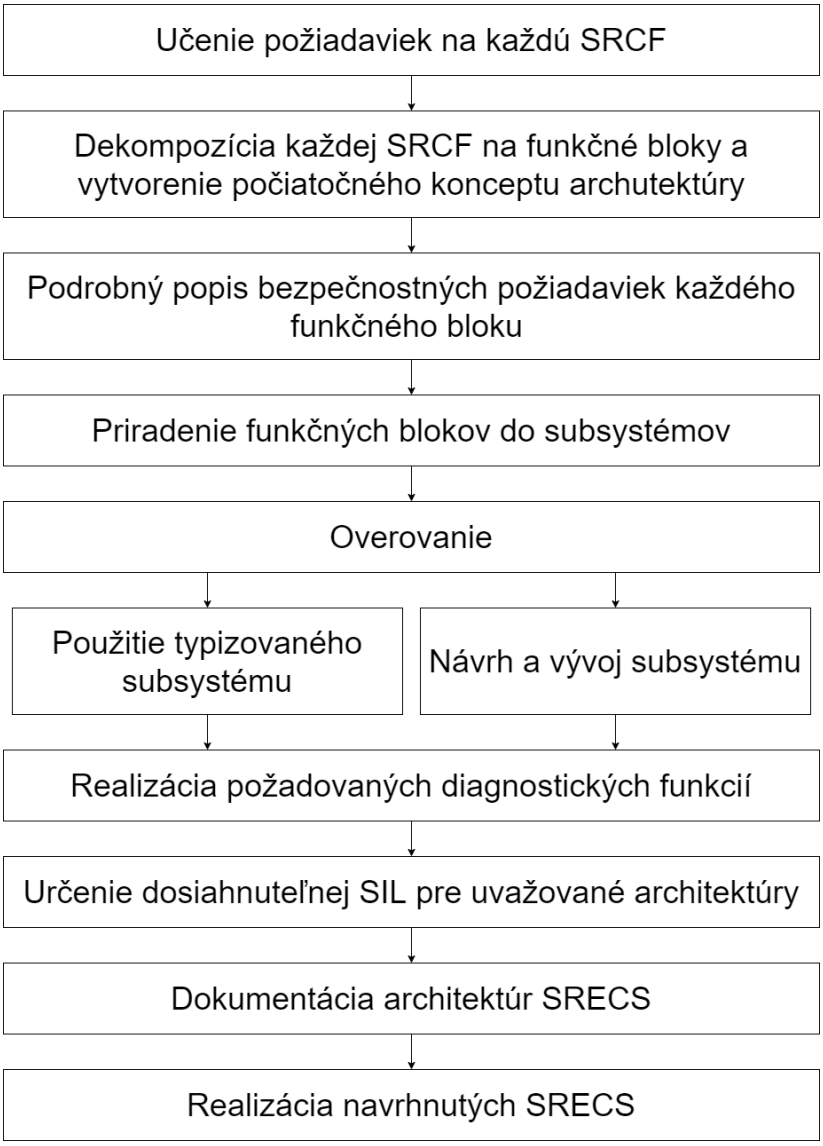
1.3 Norma EN 62061

Opäť ide o normu typu B1, ktorá sa venuje funkčnej bezpečnosti elektrických, elektronických a programovateľných riadiacich systémov. Obsahuje požiadavky, ktoré sú aplikovateľné na návrh systémovej úrovne všetkých typov elektrických bezpečnostných riadiacich systémov a tiež na návrh nie veľmi zložitých subsystémov [4].

Táto norma používa na vyhodnocovanie inú metodiku, ktorej výstupom sú úrovne integrity bezpečnosti (SIL – Safety Integrity Levels). Podobne ako u metodiky PL (norma EN ISO 13849-1), bezpečnostné funkcie sú opäť rozdelené na 3 alebo 5 subsystémov:

- Vstupný subsystém
- Logický subsystém
- Výstupný subsystém
- 2x Komunikačný subsystém (voliteľné)

Na obrázku 4 na ďalšej strane je znázornený postup návrhu a vývoja bezpečnostných funkcií pomocou tejto normy.



Obr. 4: Sled činností procesu návrhu a vývoja SRECS [podľa ČSN EN 62061:2005]

Jednotlivé kroky vyššie popísaného postupu budú ešte detailnejšie rozobraté v kapitole 2.1 – Proces zberu parametrov a ich vyhodnotenie. Cieľom tohto procesu je určenie úrovne integrity bezpečnosti pre všetky subsystémy, kombináciou ktorých potom zistíme výsledné SIL celej bezpečnostnej funkcie.

Na vyhodnotenie SIL potrebujeme poznať nasledujúce parametre:

- Architektúra
- PFH_D
- DC – rovnaké ako u normy EN ISO 13849-1
- CCF – podobné ako u normy EN ISO 13849-1
- SFF
- HFT

1.3.1 Architektúra

Podobne ako v norme EN ISO 13849-1, aj tu sú pre subsystémy špecifikované architektúry, ktoré treba dodržiavať hlavne pri vlastnom návrhu subsystémov. Každá architektúra ma svoje obmedzenia, ktoré potom limitujú jej použitie. Jedným z najdôležitejších parametrov architektúr je ich odolnosť voči vadám hardwaru (HFT), ktorý udáva, ako sa bude systém chovať pri zlyhaní niektorej z jeho častí.

Norma definuje nasledujúce 4 architektúry:

- A – nulová odolnosť proti vadám bez diagnostickej funkcie.
- B – odolnosť proti jednotlivej vade bez diagnostickej funkcie.
- C – nulová odolnosť proti vadám s diagnostickou funkciou.
- D – odolnosť proti jednotlivej vade s diagnostickou funkciou.

Každá architektúra používa pre výpočet parametrov daného subsystému iný vzorec. V norme sú jednotlivé architektúry uvedené aj s obrázkami, avšak je dôležité podotknúť, že ide o čisto logické znázornenie a nie je možné ich považovať za fyzické architektúry.

1.3.2 Pravdepodobnosť nebezpečného zlyhania za hodinu

Skrátene PFH_D (Probability of dangerous Failure per Hour). Je to číselná hodnota, na základe ktorej sa určuje hodnota SIL, podľa nasledujúcej prevodnej tabuľky.

Tab. 4: Prevod PFH_D na úrovne integrity bezpečnosti [podľa ČSN EN 62061:2005]

SIL	PFHD
3	$\geq 10^{-8}$ až $< 10^{-7}$
2	$\geq 10^{-7}$ až $< 10^{-6}$
1	$\geq 10^{-6}$ až $< 10^{-5}$

Na určenie PFH_D je treba vykonať analýzu jednotlivých komponentov, ktoré tvoria subsystém (krok 2 v obrázku 4 – dekompozícia SRCF na funkčné bloky). PFH_D celej bezpečnostnej funkcie sa potom určí ako súčet PFH_D jednotlivých komponentov (subsystémov).

Keďže tento parameter má podobné využitie ako $MTTF_D$ u PL metodiky (viď kapitolu 1.2.3), tak medzi týmito parametrami existuje vzájomný prepočet [5]:

$$PFH_D = \frac{1}{MTTF_D} \tag{1.2}$$

Je dôležité podotknúť, že ak sa jedná o dvojkanálové subsystémy, tento prepočet nie je úplne správny a nemal by sa používať. V nami tvorenej aplikácii ale kombinácia PL a SIL metodiky nebude umožnená takže tieto prepočty nebudú potreba.

1.3.3 Podiel bezpečných porúch

Skrátene SFF (Safe Failure Fraction). Má podobný význam ako DC. Väčšinou je udávaný výrobcom, avšak dá sa zistiť aj pomocou nasledujúceho výpočtu [6]:

$$SFF = \frac{\sum \lambda_S + \sum \lambda_{DD}}{\sum \lambda_S + \sum \lambda_D} \tag{1.3}$$

$\lambda_{DD} = množstvo\ detekovaných\ nebezpečných\ zlyhaní$
$\lambda_D = množstvo\ všetkých\ nebezpečných\ zlyhaní$
$\sum \lambda_S = množstvo\ bezpečných\ zlyhaní$
$\sum \lambda_S + \sum \lambda_D = množstvo\ všetkých\ zlyhaní$

1.3.4 Odolnosť proti vadám hardwaru

Skrátene HFT (Hardware Failure Tolerance). Udáva informáciu o počte porúch, ktorým môže systém odolávať, než dôjde k nebezpečnému zlyhaniu [7].

Tento parameter priamo závisí na počte kanálov jednotlivých architektúr. Napríklad pri subsystéme s architektúrou A, ktorá ma len jeden kanál, je hodnota odolnosti hardwaru proti chybám (HFT) rovná 0, pretože už pri výskyte jednej vady väčšinou systém prestane plniť bezpečnostnú funkciu. Na presné určenie HFT sa používa tabuľka č. 5, ktorej vstupnými parametrami sú SFF a SIL.

Tab. 5: Obmedzenie architektúry na subsystémy [podľa ČSN EN 62061:2005]

SFF	HFT		
	0	1	2
< 60%	nepovolené	SIL1	SIL2
60% - < 90%	SIL1	SIL2	SIL3
90% - < 99%	SIL2	SIL3	SIL3
≥ 99%	SIL3	SIL3	SIL3

2 NÁVRH INFORMAČNÉHO SYSTÉMU

V tejto kapitole sa budeme venovať návrhu informačného systému a tvorbe testovacej webovej aplikácie. Informačný systém je systém na zber, udržiavanie, spracovanie a poskytovanie informácií [8]. Ako príklad môže poslúžiť informačný systém školy alebo kartotéka. Informačné systémy vznikajú aby uľahčovali a častokrát automatizovali každodenné rutinné záležitosti, ktoré musia ľudia vykonávať najčastejšie v zamestnaní. Ich návrh predstavuje komplexnú problematiku, ktorá v praxi zahŕňa ľudí z mnohých oblastí ako napr. programátorov, dátových analytikov, programových manažérov atď.

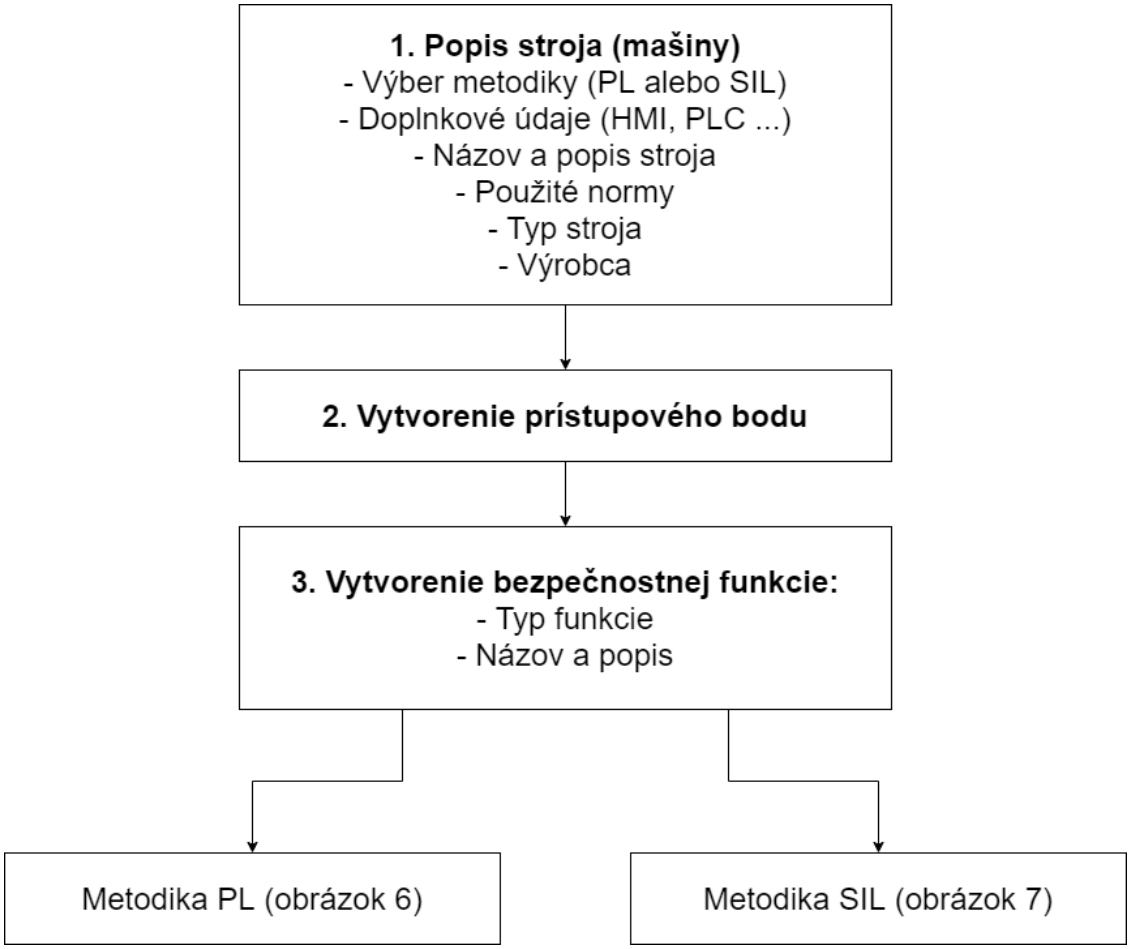
Za vznikom každej aplikácie sú určité dôvody a potreby užívateľov. V nasledujúcich bodoch si zhrnieme dôvody, prečo vzniká práve táto aplikácia:

1. Návrh bezpečnostných funkcií pre akýkoľvek stroj a výber správnych komponentov je relatívne komplikovaná záležitosť a vyžaduje si mať naštudované patričné normy. Aplikácia ktorú budeme v ďalších kapitolách navrhovať by mala túto prácu uľahčiť a vniesť do problematiky väčší prehľad.
2. Vyhodnocovanie bezpečnosti si vyžaduje v praxi niekoľko výpočtov. Využitím aplikácie bude užívateľ od tejto potreby odbremenený a jeho jedinou úlohou bude získať potrebné údaje na výpočty.
3. Samotný návrh bezpečnostnej funkcie si vyžaduje jej zloženie z viacerých subsystémov. Pri tomto procese môže nastať chyba na strane človeka. Týmto chybám sa dá však využitím aplikácie predísť, keďže okrem výpočtov bude aplikácia aj kontrolovať kompatibilitu medzi vloženými subsystémami.
4. V praxi sa často stáva že viaceré bezpečnostné funkcie zdieľajú spoločné subsystémy. Aplikácia bude teda umožňovať užívateľovi opätovné použitie už existujúcich subsystémov, ktoré boli v minulosti vytvorené.
5. Jeden z podstatných krokov pri návrhu stroja je výber vhodnej riadiacej jednotky, ktorá sa bude starať o bezpečnosť. Tento výber predstavuje určitý algoritmus od ktorého bude užívateľ taktiež odbremenený a bude ho vykonávať aplikácia.

2.1 Proces zberu parametrov a ich vyhodnotenie

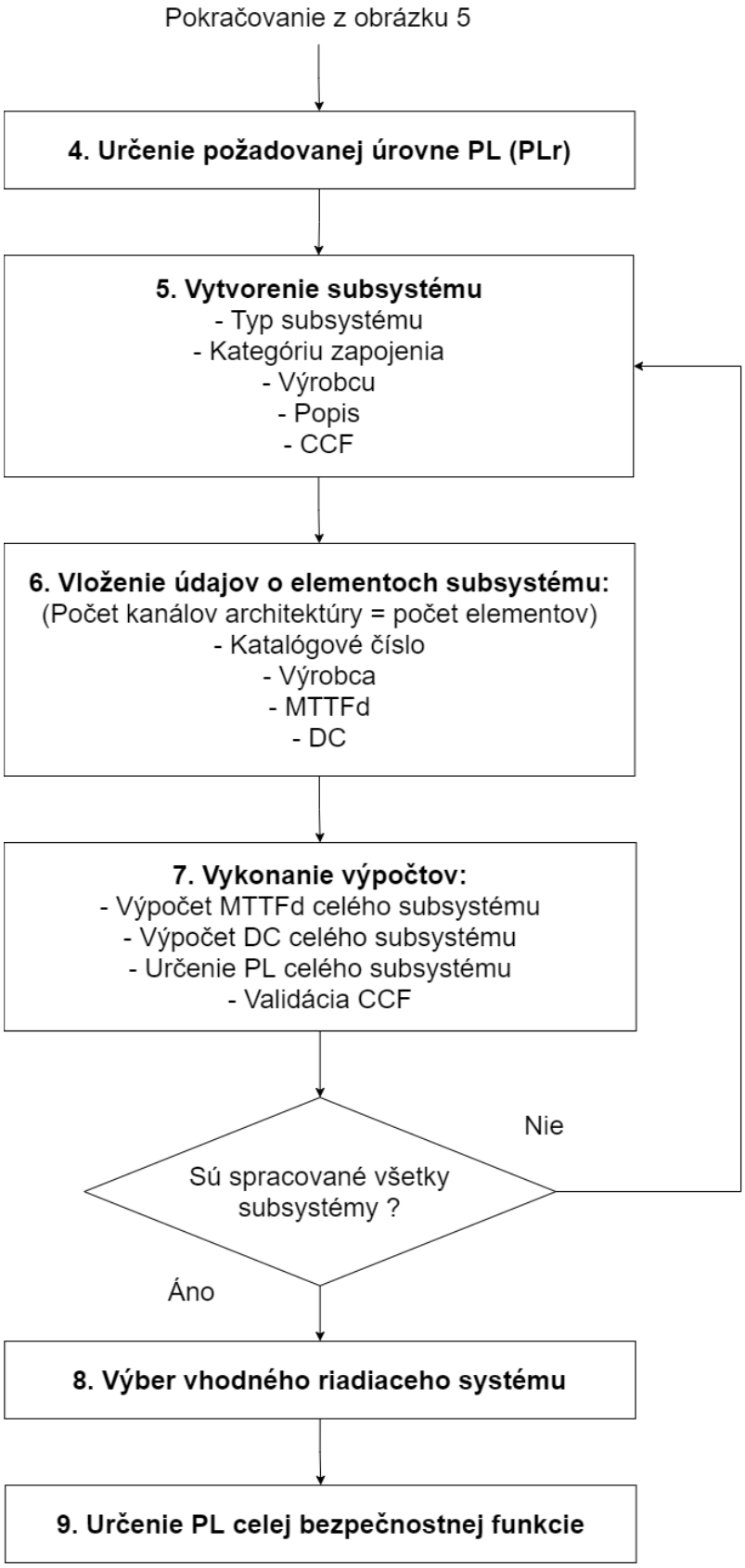
Ako sme mohli vidieť v prvej kapitole, vyhodnocovanie bezpečnostných funkcií si vyžaduje poznať mnohé parametre. Predtým ako sa pustíme do návrhu konkrétneho dátového modelu pre našu aplikáciu je dobré si zostaviť algoritmus (postupnosť krokov) zberu potrebných dát. V našom prípade bude aplikácia umožňovať vyhodnocovať bezpečnosť pomocou dvoch metodík (PL a SIL metodiky). Z toho vyplýva že náš algoritmus bude musieť ponúkať dve možné cesty ako sa dopracovať k výsledku. Prvé tri kroky ktoré sú znázornené na obrázku 5 majú však obe metodiky spoločné.

V každom kroku sú vypísané najdôležitejšie údaje (parametre), ktoré bude musieť užívateľ vkladať do aplikácie. Zjednodušene sa dá povedať že každý krok v tomto algoritme bude reprezentovaný vlastným formulárom. Toto tvrdenie je však len veľmi orientačné a konkrétnom prevedení front-endových formulárov sa budeme baviť v kapitole 2.4 – Návrh front-endu.

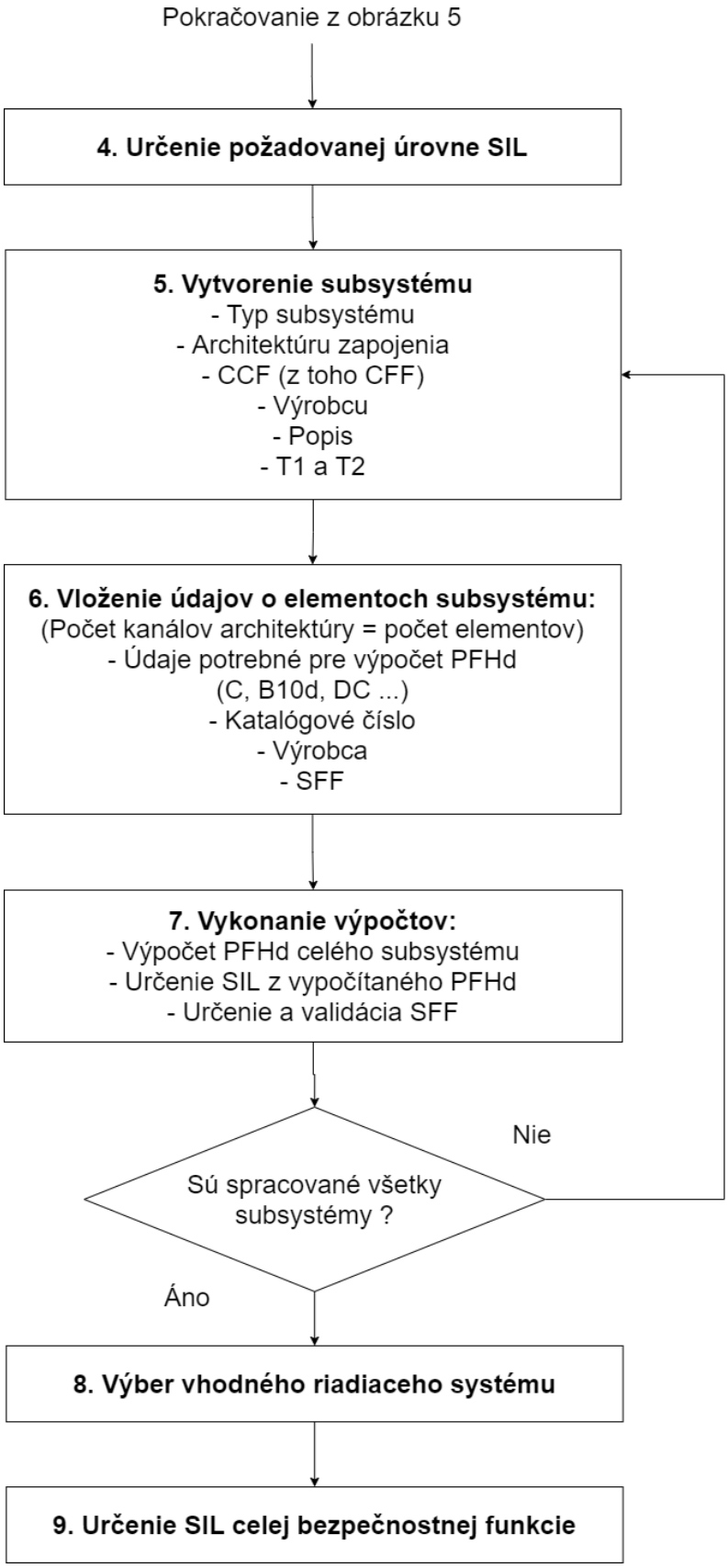


Obr. 5: Prvé tri kroky algoritmu zberu potrebných parametrov

To, ktorou metodikou algoritmus pokračuje určí užívateľ výberom požadovanej metodiky v prvom kroku algoritmu. Tento údaj v neskorších štádiách procesu vyhodnocovania bezpečnosti stroja už nebude možné meniť. Z toho vyplýva, že navrhovaná aplikácia bude umožňovať vyhodnocovanie jedného stroja len jednou metodikou. V praxi je síce vzájomná kombinácia oboch metodík v rámci jedného stroja bežná, toto rozšírenie by však nebolo úplne triviálne a celú aplikáciu by značne skomplikovalo. Na obrázkoch nižšie môžeme vidieť postupnosť krokov pri zbere parametrov už pre konkrétne metodiky.



Obr. 6: Postup zberu dát pri vyhodnocovaní pomocou metodiky PL



Obr. 7: Postup zberu dát pri vyhodnocovaní pomocou metodiky SIL

Ako môžeme vidieť z obrázkov vyššie, postupy sú skoro totožné a líšia sa len v parametroch, ktoré užívateľ vkladá do systému. Kroky 5, 6, 7 sa vykonávajú pre každý jeden subsystém, okrem logického subsystému.

Práve logický subsystém sa vyberá v kroku 8 a jeho výber zabezpečuje samotná aplikácia a nie užívateľ ako je uvedené v dôvodoch, prečo táto aplikácia vzniká. Logický subsystém sa vyberá z možných logických riadiacich systémov, ktoré budú v aplikácii predefinované. Na výber bude z nasledujúcich riadiacich systémov:

- **Relé** – neprogramovateľné jednoduché relé
- **CR30** – relé modul s nastaviteľnými parametrami
- **GMX** – menšie bezpečnostné PLC
- **GLX** – bezpečnostné PLC

Nasledujúca tabuľka znázorňuje základné parametre vyššie uvedených riadiacich systémov. Žltou farbou sú znázornené kritické parametre.

Tab. 6: Riadiace systémy používané v aplikácia a ich základné parametre

Meno	Max. počet vstupov	Max. počet výstupov	Max. počet prístupových bodov	Max. PL	Max. SIL
Relay	4 SI	4 SO	Menej ako 2	e	3
CR30	12 SI	10 SO	Menej ako 5	e	3
GMX	Konfigurovateľné	6144 SI/O	Neobmedzené	e	3
GLX	Konfigurovateľné	65536 SI/O	Neobmedzené	e	3

Hlavým parametrom pri výbere logiky je počet bezpečnostných funkcií, ktoré bude musieť ovládať. Obecne sa dá povedať, že čím viac prístupových bodov, tým viac bezpečnostných funkcií bude treba na ich zabezpečenie pred možnými úrazmi, ktoré môžu v týchto oblastiach vzniknúť. Každá bezpečnostná funkcia je väčšinou zložená z iných subsystémov a s tým aj priamo súvisí počet vstupov a výstupov, ktoré potrebuje pre svoju správnu funkciu.

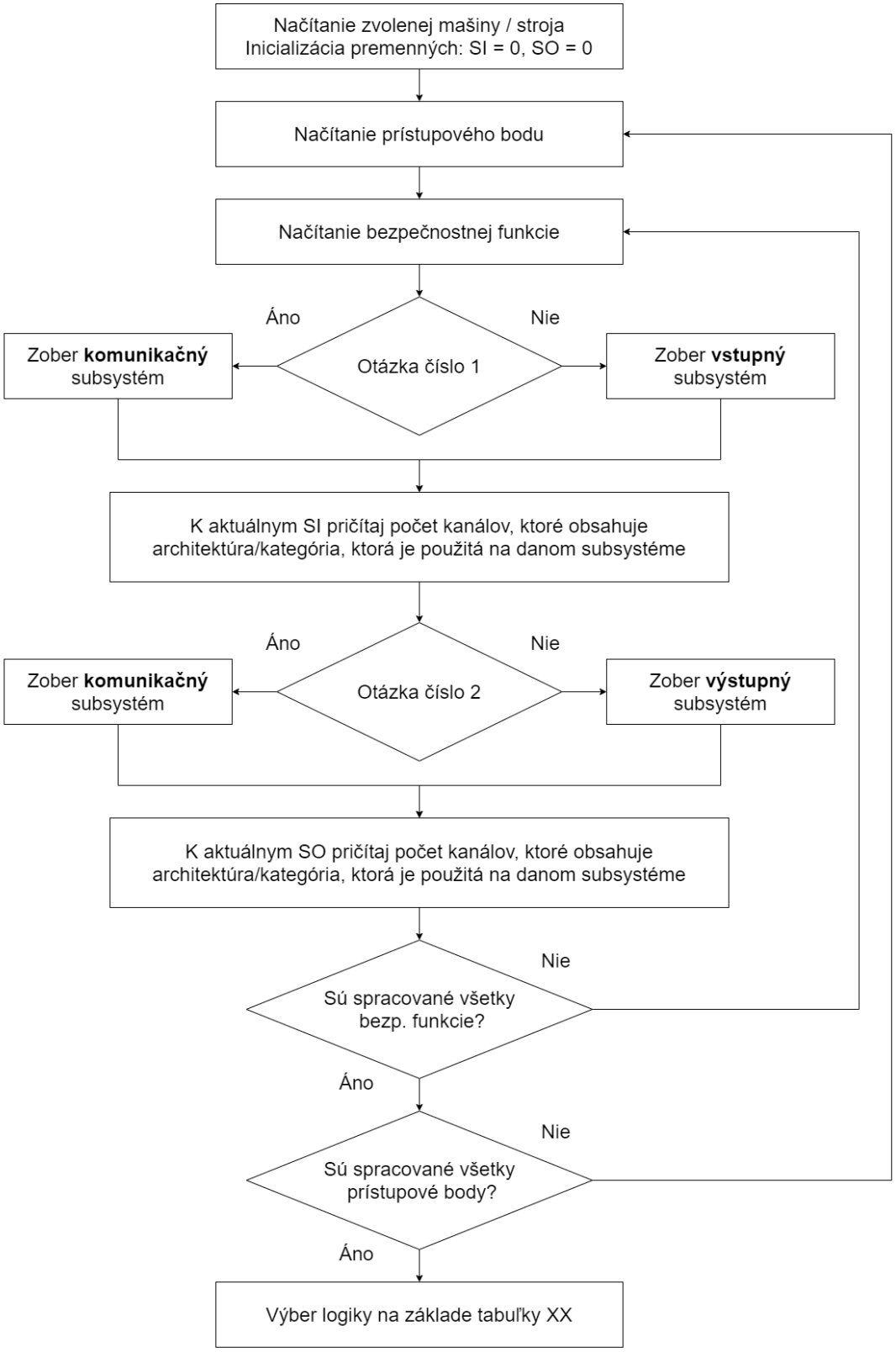
Kľúčovým parametrom teda nebude priamo počet bezpečnostných funkcií alebo prístupových bodov, ale počet štandardných vstupov a výstupov, ktoré bude vybraná logika potrebovať na to, aby sa dala plnohodnotne prepojiť so všetkými subsystémami, ktoré tvoria bezpečnostnú funkciu. Tieto čísla priamo súvisia s architektúrou alebo kategóriou (záleží na metodike), ktorá je použitá na zapojenie daného subsystému do celej bezpečnostnej funkcie. Každá kategória/architektúra totiž presne určuje počet kanálov, ktorými môže byť daný subsystém prepojený so svojím okolím. Bližšie informácie o kategóriách a architektúrach subsystémov môžeme nájsť v predchádzajúcich kapitolách 1.2.2 a 1.3.1.

Celkový počet štandardných vstupov a výstupov teda získame prejdením všetkých bezpečnostných funkcií, ktoré sú použité na vybranej mašine a sčítaním kanálov, ktoré sú využité na prepojenie logiky so vstupným a výstupným subsystémom (prípadne komunikačnými subsystémami, ktoré sú ako ukazuje obrázok 3 umiestnené medzi logickým a vstupným/výstupným subsystémom).

Obrázok číslo 8 na ďalšej strane znázorňuje opísaný algoritmus pomocou data flow diagramu. Pre jednoduchosť nie sú do obrázku zahrnuté kontroly, či daná mašina disponuje aspoň jedeným prístupovým bodom, alebo či spracovávaný prístupový bod disponuje aspoň jednou bezpečnostnou funkciou.

Po vybraní správneho riadiaceho systému pre mašinu aplikácia automaticky doplní vybranú logiku do všetkých bezpečnostných funkcií, ktoré tento subsystém nemajú vybraný užívateľom. Na tomto mieste je teda dôležité poukázať na fakt, že užívateľ bude mať možnosť si logický subsystém pri tvorbe bezpečnostnej funkcie vybrať manuálne, alebo bude automaticky doplnený aplikáciou po jeho vybraní. Akonáhle budú mať všetky bezpečnostné funkcie všetky 3 požadované subsystémy (vstupný, logický a výstupný – nutné minimum na vyhodnotenie bezpečnostnej funkcie) budú všetky bezpečnostné funkcie vyhodnotené podľa metodiky, ktorú užívateľ zvolil pre celú mašinu. Tento proces vyhodnocovania bezpečnosti celej mašiny je znázornený na obrázku 9 na ďalšej strane.

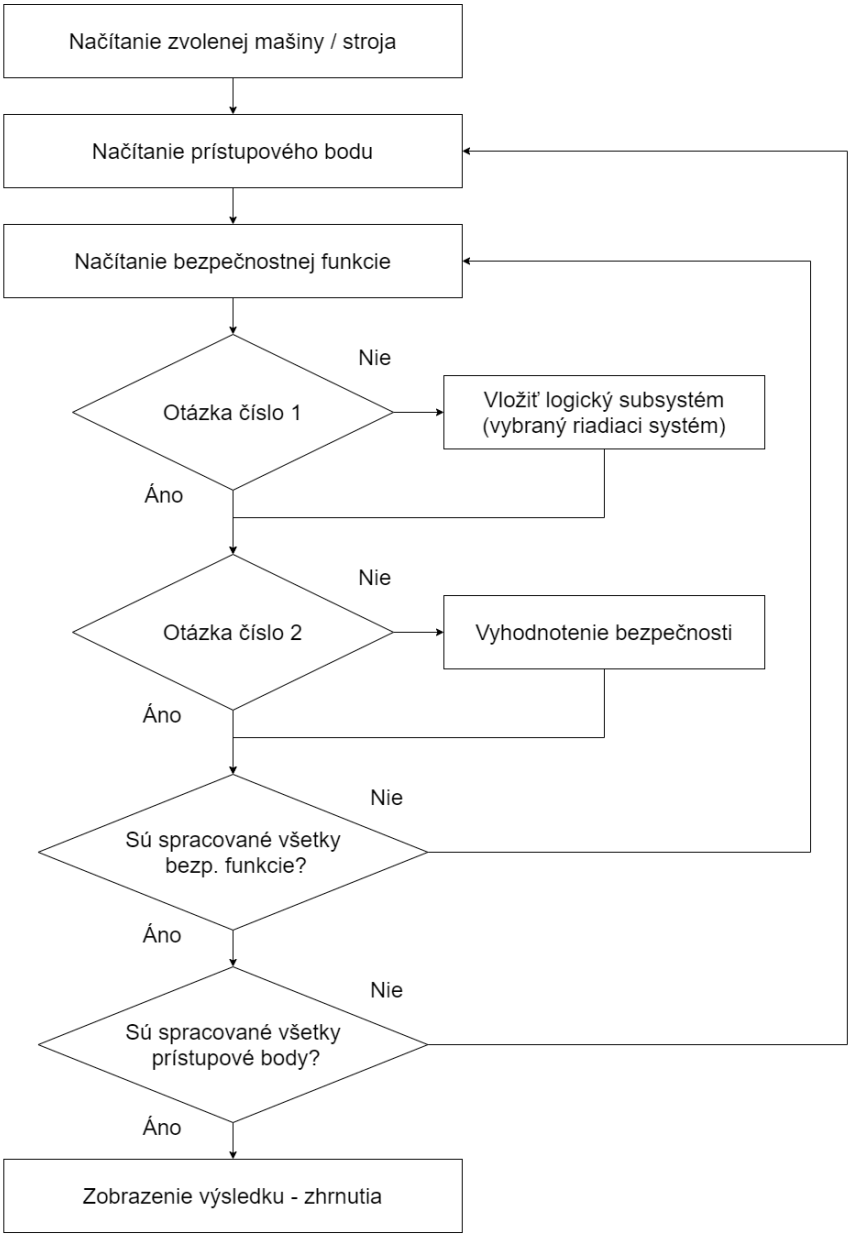
Výsledkom vyššie opísaného postupu bude informácia pre užívateľa, v akom stave sa jeho mašina nachádza. K dispozícii bude zhrnutie, v ktorom si bude môcť užívateľ prezrieť jednotlivé prístupové body a ich bezpečnostné funkcie. Presnú informáciu o tom, ako dopadlo vyhodnocovanie jednotlivých bezpečnostných funkcií bude udávať atribút s názvom stav (po anglicky state, v aplikácií používaný pod názvom CurrentState). Viac o stavoch jednotlivých záznamov si povieme v časti 2.2 – návrh dátového modelu.



OTÁZKA ČÍSLO 1: Existuje komunikačný subsystém medzi **vstupným** a **logickým** subsystémom ?

OTÁZKA ČÍSLO 2: Existuje komunikačný subsystém medzi **výstupným** a **logickým** subsystémom ?

Obr. 8: Algoritmus zhromažďovania parametrov pre výber vhodnej logiky



OTÁZKA ČÍSLO 1: Má bezpečnostná funkcia vybraný logický subsystém ?

OTÁZKA ČÍSLO 2: Má bezpečnosntá funkcia určenú výslednú úroveň bezpečnosti ?

Obr. 9: Postup pri vyhodnocovaní bezpečnosti pre celú mašinu

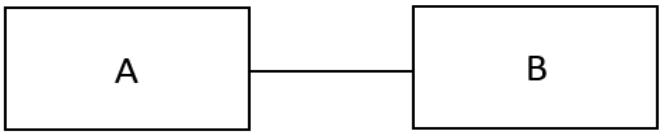
2.2 Návrh dátového modelu

Po tom ako sme si v minulej kapitole opísali základné procesy pomocou ktorých bude aplikácia zbierať dáta a následne ich vyhodnocovať, môžeme začať navrhovať dátový model tak, aby čo najviac vyhovoval naším potrebám a bol pripravený na prípadné budúce modifikácie a rozšírenia.

Dátové modelovanie patrí medzi prvé kroky pri návrhu softwaru a preto je treba tomuto kroku venovať zvýšenú pozornosť, aby sme predišli rôznym problémom, ktoré by mohli nastať v neskorších štádiách vývoju aplikácie. V tomto štádiu sa analyzujú a definujú požiadavky na štruktúru dát.

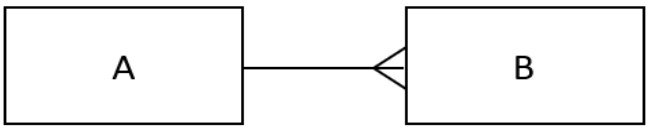
Výsledkom tohto procesu je ERD (Entity-relationship diagram), ktorý znázorňuje väčšinou predbežný obsah jednotlivých entít a ich vzájomné relácie. Ešte predtým ako sa pustíme do opisu kostry dátového modelu je však podstatné, aby sme porozumeli jednotlivým relačným väzbám medzi entitami. V databázovom svete poznáme 3 základné typy relácií (vzťahov) medzi entitami:

- **One-to-one** (1:1) – entita A môže byť spojená s jednou entitou B a naopak.



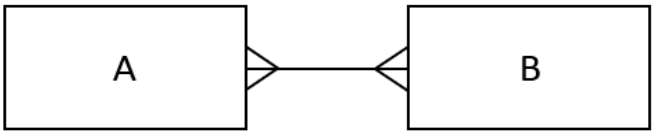
Obr. 10: Grafické znázornenie väzby one-to-one

- **One-to-many** (1:M) – entita A môže byť spojená s mnohými entitami B, ale zároveň entita B môže byť spojená len s jednou entitou A.



Obr. 11: Grafické znázornenie väzby one-to-many

- **Many-to-many** (M:N) – entita A môže byť spojená s mnohými entitami B a naopak. Pri tomto vzťahu sa využíva pomocná prechodová tabuľka.



Obr. 12: Grafické znázornenie väzby many-to-many

Každý dátový model má svoju hlavnú kostru, od ktorej sa odvíjajú jeho zvyšné časti. V našom prípade ju tvorí 8 entít (tabuliek), ktoré sú medzi sebou prepojené relačnými väzbami, tak ako ukazuje obrázok 17. V tejto hlavnej kostre dátového modelu sú zahrnuté len tie najdôležitejšie entity, avšak celý dátový model obsahuje aj rôzne číselníky a niekoľko entít, ktoré zabezpečujú „systémové podpalubie“ aplikácie.

Medzi entity v „systémovom podpalubí“ patria napríklad, entity na ukladanie údajov o užívateľoch, ich prístupových právach atď.

Charakteristickou črtou väčšiny entít, ktoré patria do hlavnej kostry dátového modelu sú nasledujúce atribúty:

- CurrentState – udržuje informáciu o aktuálnom stave daného záznamu..
- IdCreated – ID užívateľa, ktorý záznam v databáze vytvoril
- IdUpdated – ID užívateľa, ktorý modifikoval existujúci záznam
- DateTimeCreated – dátum a čas, kedy bol záznam vytvorený
- DateTimeUpdated – dátum a čas, kedy bol záznam naposledy modifikovaný

O všetky vyššie uvedené atribúty sa stará aplikácia automaticky počas behu programu a užívateľ nemá žiadnu možnosť ich upravovať. Okrem týchto atribútov má každá tabuľka v databázy aj atribút IsRemoved, ktorý udáva informáciu o tom, či je záznam vymazaný alebo nie. Zmazaním akéhokoľvek záznamu sa teda z databázy reálne nič neodstráni, aplikácia funguje na tzv. soft delete. Týmto sa dá predísť problému, ktorý by vznikol ak by užívateľ nechcene zmazal niečo čo nechcel.

Na nasledujúcich riadkoch sa bližšie pozrieme na jednotlivé hlavné tabuľky (entity) kostry dátového modelu. Zároveň sa budeme venovať návrhu ich stavov.

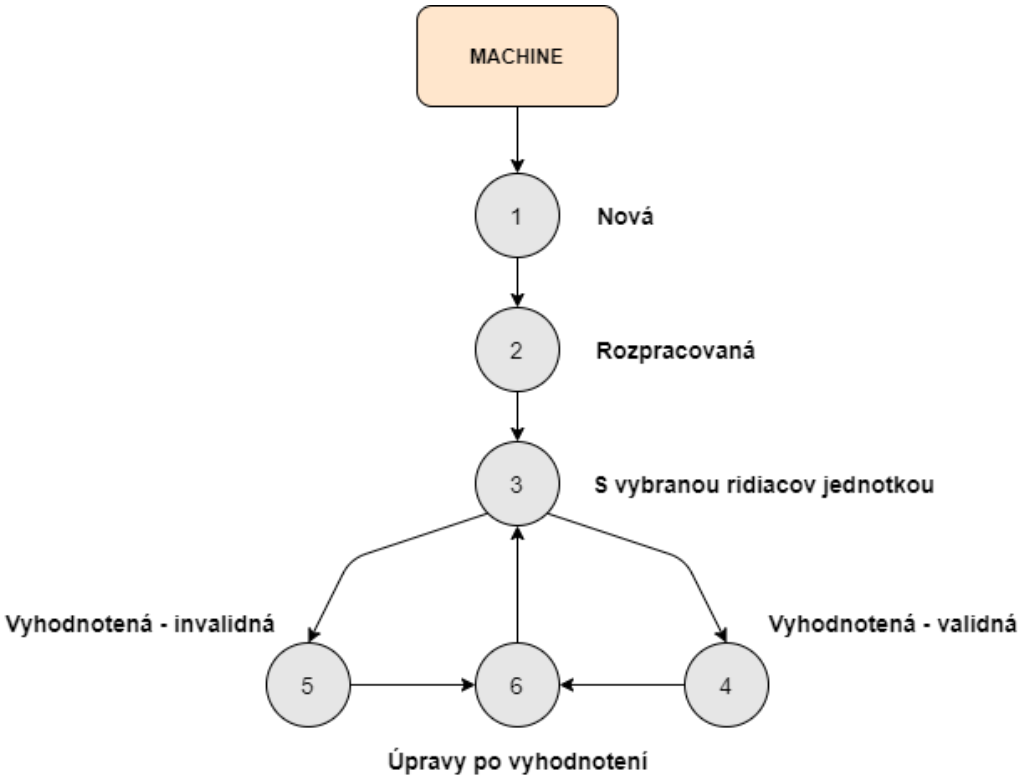
2.2.1 Entita Machine

Táto entita je určená na ukladanie údajov o strojoch (mašinách), ktorým robíme bezpečnostnú analýzu a návrh bezpečnostných funkcií. K najdôležitejším atribútom patrí EvaluationMethod_Id, ktorý nám dáva informáciu o tom, akou metódou je vyhodnocovaná bezpečnostná analýza (PL alebo SIL). Bez znalosti tohto atribútu záznam v tejto tabuľke nemôže byť vytvorený.

Na nasledujúcej strane môžeme vidieť stavovo prechodový diagram pre túto entitu. Prechody medzi jednotlivými stavmi sú riadené nasledovnými pravidlami:

- Počiatočný stav každého nového záznamu v tejto tabuľke je stav 1 (Nová).
- Po pridaní prvej bezpečnostnej funkcie sa záznam dostane do stavu číslo 2. Mašina bude označená ako rozpracovaná. V tomto stave záznam zotrváva po celú dobu, kedy užívateľ pracuje na tvorbe bezpečnostných funkcií.
- Po tom, ako aplikácia vyberie na užívateľovu žiadosť vhodnú riadiacu jednotku pre celú mašinu, prejde mašina do stavu číslo 3 (s vybranou riadiacou jednotkou).
- Po vybraní riadiaceho systému sa spustí algoritmus vyhodnotenia bezpečnosti pre celú mašinu a na základe jeho výsledku mašina skončí v stave číslo:

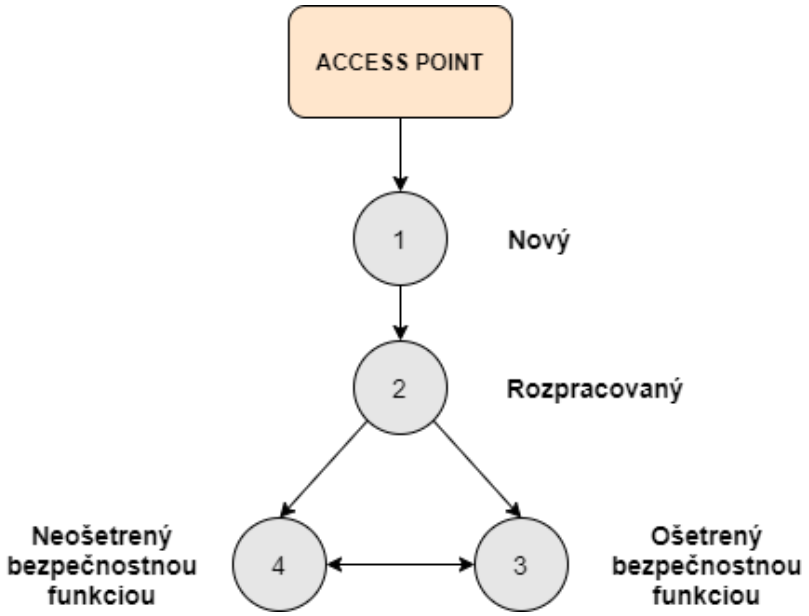
    - 4 (vyhodnotená – validná) – ak sú všetky bezpečnostné funkcie úspešne vyhodnotené a ich výsledná úroveň bezpečnosti splňuje minimálnu požadovanú úroveň.
    - 5 (vyhodnotená – invalidná) – ak sa u niektorej bezpečnostnej funkcie naskytol nejaký problém. Napr. nesplnenie minimálnej požadovanej úrovne bezpečnosti, nekompatibilné subsystémy, neplatné CCF.
- V prípade že je mašina vyhodnotená (nezáleží či validná alebo invalidná) a užívateľ zmení počet prístupových bodov alebo bezpečnostných funkcií, ktoré sú použité v rámci vybraného stroja, záznam prejde do stavu 6 (úpravy po vyhodnotení). Tento stav signalizuje užívateľovi, že vplyvom jeho zmien je nutné opäť vybrať vhodnú logiku a vyhodnotiť nanovo bezpečnosť.
- Po tom, ako sa spustí opätovný výber vhodnej logiky pre mašinu v stave 6, prejde sa cez stav 3 do stavu 4 alebo 5, na základe vyššie uvedených pravidiel.



Obr. 13: Stavovo prechodový diagram pre entitu Machine

2.2.2 Entita AccessPoint

Každý stroj (mašina) má určitý počet prístupových bodov, ktoré musia byť z hľadiska bezpečnosti ošetrené voči možným zraneniam, ktoré môže vzniknúť pri pohybe osoby v danej oblasti. Entita AccessPoint slúži na ukladanie údajov o týchto prístupových bodoch. Medzi entitou Machine a entitou AccessPoint je väzba 1:M, čo v preklade znamená, že jedna mašina môže mať niekoľko prístupových bodov a zároveň jeden prístupový bod môže patriť iba jednej mašine.



Obr. 14: Stavovo prechodový diagram pre entitu AccessPoint

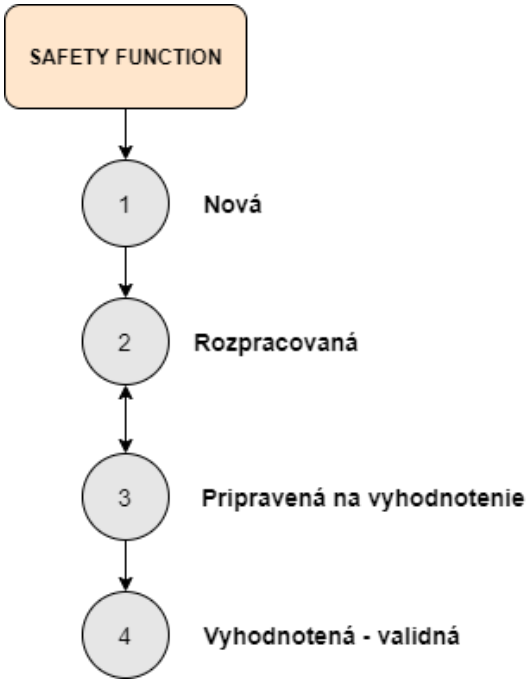
Prechody medzi jednotlivými stavmi sú riadené nasledovnými pravidlami:

- Počiatočný stav každého nového záznamu v tejto tabuľke je stav č. 1 (nový).
- Po pridaní prvej bezpečnostnej funkcie as záznam dostane do stavu číslo 2 (rozpracovaný). V tomto stave záznam zotrváva po celú dobu kedy užívateľ pracuje na bezpečnostných funkciách.
- Po vyhodnotení bezpečnosti pre celú mašinu sa záznam dostane do stavu číslo:
  - 3 (ošetrený bezpečnostnou funkciou) – ak má prístupový bod aspoň jednu bezpečnostnú funkciu, ktorá spĺňa všetky požiadavky.
  - 4 (neošetrený bezpečnostnou funkciou) – ak prístupový bod nemá ani jednu bezpečnostnú funkcia, alebo ak ani jedna jeho bezpečnostná funkcia nespĺňa požiadavky.
- V prípade upravenia počtu bezpečnostných funkcií po vyhodnotení prístupového bodu, je možné prechádzať medzi stavmi 3 a 4.

2.2.3 Entita SafetyFunction

Táto entita je jednoznačne najväčšia a najzložitejšia. Jej rozsiahlosť je spôsobená tým, že každá metodika vyhodnocovania bezpečnosti pracuje s mierne odlišnými parametrami a dátový model musí byť nachystaný na obi dva prípady. Väčšina jej atribútov sú odkazy do číselníkov, ktoré nie sú zahrnuté v hlavnej kostre dátového modelu na obrázku 17.

Z obrázku 17 môžeme vidieť, že medzi entitou SafetyFunction a AccessPoint je ešte jedna tabuľka s názvom AccessPointSafetyFunction. Je to tzv. spojovacia tabuľka. Jej úlohou je zabezpečiť relačnú väzbu M:N medzi entitami AccessPoint a SafetyFunction. V preklade to znamená, že jeden prístupový bod môže mať viacero bezpečnostných funkcií a bezpečnostná funkcia môže byť použitá pre viacero prístupových bodov.



Obr. 15: Stavovo prechodový diagram pre entitu SafetyFunction

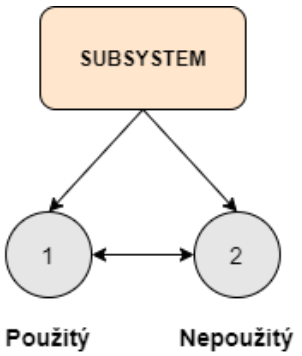
Prechody medzi jednotlivými stavmi sú riadené nasledovnými pravidlami:

- Počiatočný stav každého nového záznamu v tejto tabuľke je stav č. 1 (nový).
- Po pridaní prvého subsystému prejde záznam do stavu číslo 2 (rozpracovaná). V tomto stave záznam zotrváva až do kým užívateľ nevyplní vstupný a výstupný subsystém (nutné minimum na vyhodnotenie bezpečnostnej funkcie).
- V prípade kedy bezpečnostná funkcia obsahuje vstupný aj výstupný subsystém, prejde záznam do stavu číslo 3 (pripravená na vyhodnotenie).
- Bezpečnostnú funkciu ktorá je vstave číslo 3 je možné vyhodnotiť. V prípade úspešného vyhodnotenia bezpečnosti sa prejde do stavu 4 (vyhodnotená – validná). Ak sa vyhodnotenie bezpečnosti nepodarí, záznam ostáva v stave číslo 3 (pripravená na vyhodnotenie).

Stav číslo 4 (vyhodnotená – validná) je v tomto stavovo prechodovom diagrame koncový stav, z ktorého sa nedá prejsť do žiadneho ďalšieho stavu. V tomto stave je bezpečnostná funkcia uzamknutá pred ďalšími úpravami.

2.2.4 Entita Subsystem

Subsystémy sú základnými stavebnými kameňmi každej bezpečnostnej funkcie. Jedným zo základných požiadaviek na túto entitu je znovu použitie jej záznamov. To znamená, že po vytvorení subsystému, ho bude možné zdieľať medzi viacerými bezpečnostnými funkciami. Túto funkcionalitu sme dosiahli pomocou spojovacej tabuľky (join table) SafetyFunctionSubsystem, ktorá zabezpečuje väzbu M:N. Jedna bezpečnostná funkcia môže byť teda (v praxi musí) zložená z viacerých subsystémov, pričom jeden subsystém môže byť požívaný viacerými bezpečnostnými funkciami. Tabuľka opäť obsahuje atribúty pre obe metodiky (PL aj SIL).



Obr. 16: Stavovo prechodový diagram pre entitu Subsystem

Ako môžeme vidieť, stavovo prechodový diagram je pre túto entitu veľmi jednoduchý. Pri vytvorení záznamu sa skontroluje, či bol subsystém vytvorený v rámci nejakej bezpečnostnej funkcie alebo ako samostatný subsystém a na základe toho sa mu priradí stav číslo:

- 1 (použitý) – ak je použitý na nejakej bezpečnostnej funkcii
- 2 (nepoužitý) – ak nie je použitý v bezpečnostnej funkcii

Prechod zo stavu 1 do 2 (alebo naopak) môže spôsobiť odobratie alebo použitie subsystému v nejakej bezpečnostnej funkcii. Detaily vytvoreného záznamu nie je možné po jeho uložení ďalej upravovať.

2.2.5 Entita Element

Je na konci celého reťazca v základnej kostre dátového modelu a je to najmenší stavebný element pri tvorbe bezpečnostných funkcií. Každý subsystém sa skladá buď z jedného alebo dvoch elementov, pričom tento počet závisí na počte kanálov používanej architektúry. Na naplnenie niektorých atribútov tejto tabuľky je potrebné vykonať menšie výpočty. Aj v tomto prípade sú v jednej tabuľke atribúty pre PL aj SIL metodiku.

Táto entita síce taktiež obsahuje atribút CurrentState, ale nedefinujeme pre ňu žiadny stavovo prechodový diagram. Každému novo vytvorenému záznamu sa automaticky priradí stav s názvom „validný“, ktorý sa ďalej nemení. Dôvodom existencie atribútu CurrentState aj napriek tomu že je v podstate nevyužitý je pripravenosť hlavných entít v dátovom modely na budúce rozšírenia aplikácie.

2.2.6 Entita TypeOfLogic

Je to jedna z najdôležitejších entít. Práve v tejto tabuľke budú uložené predefinované logiky, z ktorých bude aplikácia následne vyberať (Relé, CR30, GMX, GLX). Táto entita sa radí medzi tzv. číselníky, ktorých typickou vlastnosťou je, že si z nich užívateľ iba vyberá. V našom prípade to teda bude znamenať, že užívateľ nebude mať v základnej verzii aplikácie možnosť túto tabuľku rozširovať o ďalšie logické systémy. Informácie o použitej logike pre danú mašinu bude potom možné získať cez FK (foreign key – cudzí kľúč) v entite Machine, ktorý slúži ako referencia (odkaz) do tabuľky TypeOfLogic.

Keďže táto entita sa radí medzi číselníky, nedefinujeme u nej žiadne stavy. Namiesto toho obsahuje atribút IsValid, ktorý udáva informáciu o tom, či je záznam aktuálny.

2.2.7 Číselníky

Okrem základnej kostry dátového modelu sú jeho neoddeliteľnou súčasťou aj číselníky. Prístup ku správe týchto číselníkov majú len autorizovaný užívatelia. Hlavným účelom týchto tabuliek je ukladanie záznamov, ktorých obsah sa často nemení a obyčajný užívateľ aplikácie z týchto záznamov len vyberá. Z toho vyplýva, že dáta v týchto tabuľkách sú dopredu nachystané. V kostre dátového modelu na obrázku 17 sú odkazy do číselníkov zvýraznené zelenou farbou. Ako môžeme vidieť, záznamy z číselníkov sú využívané vo všetkých hlavných entitách (okrem spojovacích tabuliek).

Keďže týchto číselníkov je v aplikácii viac ako 20, nie sú zahrnuté do kostry dátového modelu kvôli jeho prehľadnosti. Kompletný dátový model obsahuje príloha B.

Obr. 17: Kostra dátového model

2.3 Návrh back-endu

Takmer každá aplikácia sa delí na tzv. front-end a back-end časť. Back-end predstavuje časť aplikácie ku ktorej nemá užívateľ priamo prístup (na rozdiel od front-endu – grafického rozhrania). Táto časť je v prípade webových aplikácie tiež nazývaná aj ako server-side.

Vo svete softwarového inžinierstva predstavuje rozdelenie na front-end a back-end uplatnenie návrhového princípu Separation of concerns (SoC). Tento návrhový princíp hovorí o rozdelení počítačového programu do samostatných častí, pričom každá z nich rieši samostatný problém [9]. V ideálnom prípade by mali byť tieto časti medzi sebou čo najmenej poprepájané. Výsledkom správneho použitia vyššie popísaného návrhového princípu je dobre štruktúrovaný kód, ktorý je ľahko rozšíriteľný, testovateľný a hlavne udržateľný z dlhodobého hľadiska. Na tomto mieste je treba povedať, že SoC sa dá uplatniť ako na celú aplikácia (napr. vyššie uvedené rozdelenie na back-end a front-end), tak aj na jednotlivé časti z ktorých je výsledná aplikácia ako celok zložená. Práve toto druhé uplatnenie si ukážeme pri návrhu back-endu, ktorému sa budeme venovať v tejto kapitole.

2.3.1 Použitý framework

Výber správneho frameworku je pre stabilnú a rýchlu aplikáciu podstatná vec. Každý framework má svoje plusy a mínus. Ich hlavný rozdiel však predstavuje programovací jazyk, ktorý je daným frameworkom podporovaný. Pre serverovú čas (back-end) nášho informačného systému bol zvolený framework od Microsoftu s názvom ASP.NET core.

ASP.NET core je open-source framework pre tvorbu webových aplikácií. Aktuálne patrí medzi najrýchlejšie webové frameworky. Veľkou výhodou je tiež to, že je cross-platfrom, čo umožňuje jeho nasadenie na akýkoľvek operačný systém. Samotný ASP.NET core framework predstavuje základ pre mnohé jeho nadstavby. Medzi tieto nadstavby patrí napr. ASP.NET core MVC, Web API, Razor Pages atď.

2.3.2 Webové API

Konkrétne riešenie serverovej časti našej aplikácie predstavuje webové API (application programming interface) realizované pomocou ASP.NET core Web API frameworku. Webové API sprístupňuje množstvo URL adries, ktoré môžu byť použité na získanie alebo upravenie dát na serveri (v databáze) [10]. Toto získavanie, prípadne upravovanie dát je realizované pomocou HTTP requestov. HTTP (hypertext trasnfer protokol) je protokol na prenos HTML dokumentov medzi serverom a klientom. Tento protokol definuje niekoľko request metód, ktoré určujú, aká akcia (operácia) sa má s danými dátami vykonať (napr. GET, POST, DELETE atď.). Samotné dáta sú potom medzi serverom a klientom predávané buď ako JSON alebo XML.

Základom webového API sú tzv. endpointy. Pod pojmom endpoint môžeme chápať metódu, ktorá má priradenú unikátnu URL adresu. Túto metódu je potom pomocou jej adresy možné vyvolať z klienta pomocou spomínaného HTTP requestu. V jednoduchosti sa teda dá povedať, že spomínané endpointy „odchytávajú“ HTTP requesty. Každý endpoint má svoju špecifickú úlohu a v mnohých prípadoch môže táto úloha zahŕňať ďalšie menšie úlohy alebo procesy. Ak by sme celú logiku požadovanej operácie mali uloženú v tele metódy daného endpointu, nie len že by sme porušili vyššie opísaný návrhový vzor separation of concerns, ale kód by bol z dlhodobého hľadiska neudržateľný. Z týchto dôvodov je serverová časť nášho informačného systému zložená z viacerých častí, pričom opisované API tvorí len jeho najvrchnejšiu časť, ktorá sa stará práve o komunikáciu s klientom. Viac o jednotlivých vrstvách a ich zložení si povieme v ďalšej kapitole.

2.3.3 Architektúra

Výber vhodnej architektúry je v dnešnom rozmanitom svete softwaru niekedy náročná úloha a častokrát si vyžaduje roky skúseností. Keby sme sa pozreli na aplikácie z rôznych firiem, zistili by sme, že každá firma si razí svoju „správnu“ architektúru, prípadne podobu, ako by mala dobrá aplikácia vyzerať. Všeobecne známou pravdou je, že každá jedna architektúra má svoje výhody aj nevýhody. S novými konceptami architektúr prichádza aj kopec nových vecí okolo, ktoré treba vedieť. Častokrát teda výsledná voľba padne na architektúru, s ktorou má daný tím vývojárov najväčšie skúsenosti. Tak tomu bolo aj v tomto prípade.

Pre náš informačný systém bola zvolená tzv. viacvrstvová architektúra. Výhody tejto architektúry by sa dali zhrnúť do nasledujúcich bodov:

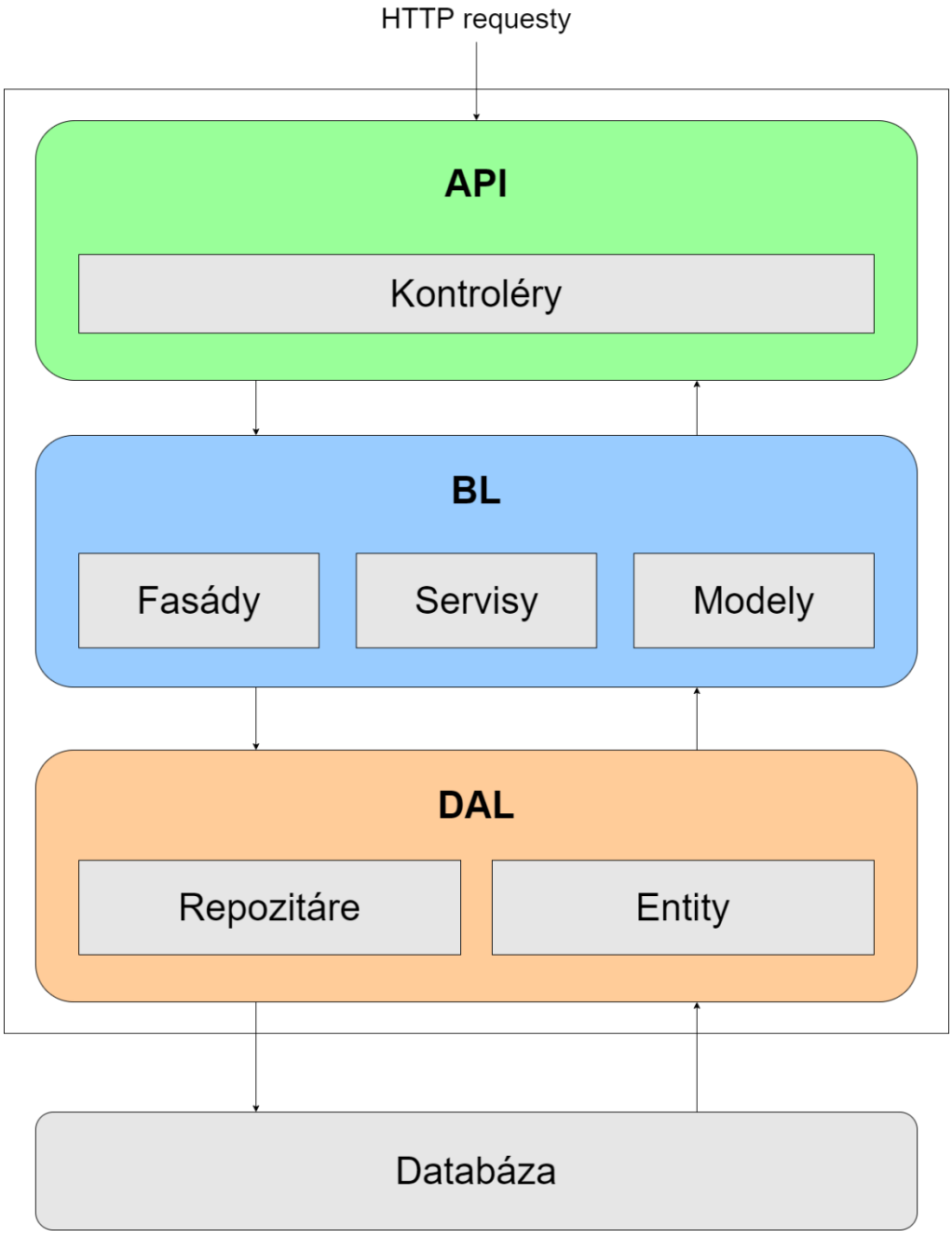
- Kód je rozdelený do viacerých vrstiev (v našom prípade do troch). Každá vrstva združuje kód s podobnou funkcionalitou (dodržanie SoC).
- Izolácia hlavnej (business) logiky od ostatného kódu.
- Výsledný kód je jednoduchší a prehľadnejší na údržbu a debugging.
- Správna realizácia architektúry umožňuje výmenu celej vrstvy bez zásahu do okolitých vrstiev. To uľahčuje prípadné budúce modifikácie a rozšírenia existujúceho kódu.
- Rozdelenie aplikácie na viac vrstiev uľahčuje vývoj v tíme o viacerých ľuďoch, kedy každý vývojár môže tvoriť svoju vrstvu a predísť tak rôznym konfliktom.
- Správne navrhnuté jednotlivé vrstvy sú dobre testovateľné, čím sa dá predísť chýbam v produkcii. Kód sa dá lepšie pokryť testami.

Ako už bolo spomenuté, každá architektúra má svoje výhody aj nevýhody. Niektoré hlavné nevýhody tejto architektúry sú:

- Výsledná aplikácia je väčšinou mierne rozsiahlejšia.
- Kód v niektorých vrstvách môže narastať rýchlejšie ako v iných.

Ako môžeme vidieť na obrázku 17, architektúra sa skladá z troch hlavných vrstiev:

1. API – najvrchnejšia vrstva (zelená farba)
2. Business Logic Layer (BL) – stredná vrstva (modrá farba)
3. Data Access Layer (DAL) – najspodnejšia vrstva (oranžová vrstva)



Obr. 18: Architektúra serverovej časti aplikácie

Na nasledujúcej časti tejto kapitoly si povieme niečo viac o jednotlivých vrstvách a ich komponentoch. Vyššie uvedený obrázok obsahuje v jednotlivých vrstvách len ich najpodstatnejšie časti.

Data Access Layer (DAL)

Predstavuje najspodnejšiu vrstvu celej architektúry. Táto vrstva zabezpečuje odosielanie a prijímanie dát z databázy. Pri konkrétnej implementácii tejto vrstvy bol použitý Entity Framework Core (EF Core), ktorý patrí medzi tzv. OR/M frameworky (Object/Relational Mapping) používané pri vývoji aplikácií na .NET platforme. EF Core zabezpečuje všetku potrebnú komunikáciu medzi databázou a aplikáciou (konkrétne DAL vrstvou). Jeho najväčšou výhodou je odbremenenie programátora od písania storage procedures v databáze a udržovania konzistencie medzi dátovými modelmi v aplikácii a tabuľkami v databáze pomocou tzv. migrácií. Tento prístup má samozrejme aj očividnú nevýhodu a to je menšia kontrola nad databázovými dotazmi (database queries), ktoré EF Core generuje za behu programu. Medzi hlavné časti tejto vrstvy patria:

- Repozitáre – obsahujú metódy, ktoré zapuzdrujú konkrétnu implementáciu kódu na prístup k MSSQL databáze.
- Entity – dátové modely (objekty) na ktoré sa mapuje obsah databázy.

Business Logic Layer (BL)

Patrí medzi najdôležitejšie vrstvy celej architektúry. Názov sám o sebe vypovedá o jeho funkcii a to je združovanie hlavnej logiky, ktorú aplikácia má vykonávať. Táto vrstva komunikuje s DAL vrstvou, pomocou ktorej získava alebo vkladá dáta do databázy. Zároveň komunikuje aj s API vrstvou, ktorej buď predáva spracované dáta z databázy alebo od nej získava dáta. V tejto vrstve sa taktiež odohrávajú všetky výpočty a procesy týkajúce sa vyhodnotenia bezpečnosti. Medzi jej hlavné časti patria:

- Fasády – predstavujú uplatnenie návrhového vzoru s názvom Facade pattern. Tieto fasády predstavujú jednoduché rozhranie na komunikáciu s rozsiahlejším a zložitejším systémom, ktorý zapuzdrujú. V našom prípade sa vo fasádach volajú rôzne metódy zo servisov, pristupuje sa k databáze cez DAL vrstvu atď.
- Servisy – zapuzdrujú funkcionalitu týkajúcu sa nejakého jedného problému. V našej aplikácií je to napr. servis, ktorý obsahuje všetky potrebné metódy a logiku k PL metodike a servis, ktorý obsahuje všetky potrebné metódy a logiku pre SIL metodiku. Rozdelenie týchto implementácií do viacerých servisov je opäť pekná ukážka uplatnenia návrhového vzoru SoC.
- Modely – modely predstavujú dátové štruktúry na ktoré sa mapujú dáta z entít, ktoré sa nachádzajú v DAL vrstve. Dôvodom je, že entity v DAL vrstve majú rovnakú štruktúru a zloženie ako entity (tabuľky) v databáze. Ich štruktúra teda nie je vždy taká, akú by sme potrebovali na ďalšie spracovanie týchto dát, prípadne ich odoslanie na klienta. Preto sa dáta pri prechode z DAL vrstvy, z ktorej vystupujú ako entity, mapujú v BL vrstve na modely. Práca s týmito modelmi je potom značne pohodlnejšia a ich štruktúra môže byť upravená presne podľa potrieb klienta. Ďalším dôvodom je, že odosielaním entít na klienta by sme potenciálnemu útočníkovi odhalili presnú štruktúru dát v databázy.

API vrstva

Predstavuje poslednú vrstvu tejto architektúry. Jej úlohou je predávanie požiadavkou od klienta do fasád, ktoré tieto požiadavky potom ďalej spracovávajú. API vrstva je relatívne jednoduchá a jej hlavnú časť tvoria kontroléri, ktoré v sebe obsahujú jednotlivé endpointy, ktoré boli popísané v kapitole 2.3.2 Webové API.

2.3.4 Open API špecifikácia

Open API špecifikácia definuje štandard, pre tvorbu dokumentačného rozhrania pre webové API, ktoré umožňuje človeku aj počítaču chápať využitie daného API, pre ktoré bola špecifikácia vygenerovaná bez potreby preskúmania kódu alebo podrobnej dokumentácie.

Tento nástroj je veľmi často využívaný v prípadoch, kedy jeden tím pracuje na tvorbe serverovej časti aplikácie (webového API) a druhý tím vývojárov na front-end časti aplikácie. Vygenerovaná open API špecifikácia potom poskytuje spoločné rozhranie, pomocou ktorého je možné tieto dve časti aplikácie spolu prepojiť. Túto špecifikáciu má ASP.NET core Web API v sebe od vydania .NET 5 zabudovanú. To výrazne uľahčuje jej používanie. Pri tvorbe informačného systému bola využitá na popis jednotlivých endpointov v kontroléroch. Zároveň špecifikácia obsahuje aj presnú štruktúru dátových modelov, ktoré sú v aplikácii používané (konkrétne popisuje dátové modely z BL vrstvy).

Medzi ďalšie výhody používania tejto špecifikácie spadá možnosť vygenerovania kódu, ktorý zabezpečuje posielanie HTTP requestov medzi klientom a serverom napr. za použitia nástroja s názvom N-Swag studio.

2.4 Návrh front-endu

V tejto kapitole sa budeme venovať druhej veľkej časti aplikácie a tou je front-end. Pod pojmom front-end môžeme tiež chápať grafické rozhranie, ktoré slúži na pohodlné ovládanie aplikácie užívateľom. Práve táto časť aplikácie predstavuje to, čo užívateľ skutočne vidí a preto sú na ňu častokrát kladené najväčšie požiadavky a jej tvorba zaberá značnú časť vývoja celej aplikácie.

Návrh front-endu sa riadi značne odlišnými pravidlami oproti back-endu. Vo väčšine prípadov slúži front-end len na získavanie a zobrazovanie požadovaných dát a nestará sa o ich ďalšie spracovanie rôznymi procesmi a algoritmami, ktoré bývajú súčasťou back-endu. Z toho dôvodu sa budeme v tejto kapitole venovať hlavne návrhu jednotlivých modulov a formulárov, ktoré sú na spomínané získavanie a zobrazovanie dát používané.

2.4.1 Použitý framework

Rovnako ako u back-endu, aj v prípade front-endu existuje mnoho frameworkov, ktoré sa nám ponúkajú na vývoj tejto časti aplikácie. Doposiaľ bola väčšina front-endových frameworkov založená na programovacom jazyku Javascript (ďalej už len ako JS), avšak v minulých rokoch vznikali mnohé iniciatívy nahradiť JS inými jazykmi.

Jednu z týchto iniciatív predstavuje aj framework Blazor, ktorý bol vybraný pre tvorbu front-end časti informačného systému. Blazor patrí opäť medzi frameworky, ktoré rozširujú ASP.NET core framework, ktorý sme si v krátkosti predstavili v kapitole 2.3.1. Jeho hlavnou výhodou je, že umožňuje tvoriť interaktívne webové stránky s využitím HTML, CSS, a programovacieho jazyka C#. Využitie jazyku C# namiesto JS značne uľahčuje vývoj a aktuálne sa tento framework stáva stále viac a viac populárnym. Na tomto mieste je treba povedať, že jazyk C# ešte nedokáže plnohodnotne nahradiť JS ak sa jedná o tvorbu front-endu (a pravdepodobne sa to ani nikdy nestane). Tento problém však Blazor rieši tým, že programátorovi umožňuje využívať aj JS, ktorý môže byť volaný z kódu napísanom v C#.

Je dobré si ešte uviesť, že Blazor sa delí na dva typy:

- Blazor sever – umožňuje beh klientskej logiky na strane serveru, čím znižuje požiadavky na výkonnosť užívateľovho počítaču.
- Blazor WebAssembly (WASM) – predstavuje Single Page Application (SPA) framework. Využíva WebAssembly, čo je nový typ kódu, ktorý môže bežať v moderných webových prehliadačoch. Vo svojej podstate je WASM nízkoúrovňový jazyk podobný assembleru s kompaktným binárnym formátom, ktorý umožňuje spustenie kódu, ktorý je napísaný napr. v Rust, C#, C++ atď. vo webových prehliadačoch.

Pre naše účely bol vybraný práve Blazor WebAssembly. Aj keď je Blazor WASM relatívne nový framework, na trhu už existuje mnoho knižníc, ktoré vývoj s ním uľahčujú. Jednou z nich je aj MudBlazor, ktorý predstavuje front-endovú knižnicu s mnohými komponentami, ktoré môže vývojár využívať. Keďže je MudBlazor open-source, splňuje podmienku na jeho použitie v tejto aplikácii. Celý front-end nášho informačného systému je teda na tejto knižnici postavený.

2.4.2 Návrh modulov

Keď už vieme, čo je to front-end a na čo slúži, môžeme sa pustiť do jeho návrhu. Je treba si upresniť, že návrhom teraz nemyslíme dizajn a rozloženie komponentov v jednotlivých formulároch ale skôr ich celkový obsah a funkcionalitu.

Návrh front-endu začína jeho rozdelením do modulov. Modul predstavuje skupinu formulárov, ktoré majú niečo spoločné alebo ktoré združujú spoločnú funkcionalitu. Tieto moduly spolu tvoria jadro front-endovej časti aplikácie. V našej aplikácii sme na základe podoby kostry dátového modelu navrhli nasledovné moduly:

- Modul pre mašiny
- Modul pre bezpečnostné funkcie
- Modul pre subsystémy

Typický spôsob ako sa môže užívateľ k jednotlivým modulom v aplikácii reálne dostať je cez menu systém. To kam presne ho menu systém presmeruje závisí na zložitosti jednotlivých modulov a na ich konkrétnom prevedení.

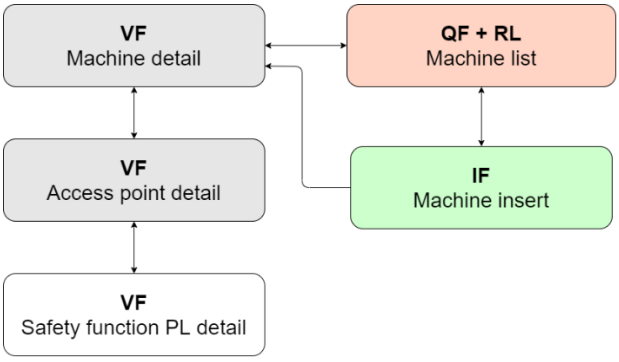
V ďalšej časti tejto kapitoly sa už bližšie pozrieme na formuláre, z ktorých sú jednotlivé moduly zložené. Formulár predstavuje jednu „obrazovku“, cez ktorú môže užívateľ manipulovať alebo prehliadať dáta. V našej aplikácií sú využité nasledovné typy formulárov:

- Query Form (QF) – obsahuje filter na vyhľadávanie
- Record List (RL) – zobrazuje zoznam položiek z určitej tabuľky
- View Form (VF) – poskytuje detailný náhľad na vybranú položku
- Insert Form (IF) – umožňuje vytváranie nových položiek

Keďže naša aplikácia nie je až tak rozsiahla, môžeme Query Form a Record List zlúčiť do jedného formulára („obrazovky“). Ak by však bola aplikácia rozsiahlejšia, je praktickejšie udržovať tieto dva formuláre oddelené, aby neboli položky v rámci jedného formulára príliš nahustené. Keďže cez niektoré z týchto formulárov je možné deštruktívne manipulovať s dátami, nemá každý užívateľ prístup úplne ku všetkým. Viac o bezpečnosti a ochrane dát si povieme v poslednej kapitole s číslom 2.5. V nasledujúcich častiach si popíšeme konkrétnu funkciu a zloženie jednotlivých modulov. Obrázky ktoré budú slúžiť na ich grafické znázornenie obsahujú aj nevyfarbené (biele) bloky. Tieto bloky predstavujú formuláre, ktoré patria do iného modulu než znázorňuje daný obrázok, ale sú v obrázku ponechané pre lepšiu predstavu navigácie medzi modulmi.

Modul pre mašiny

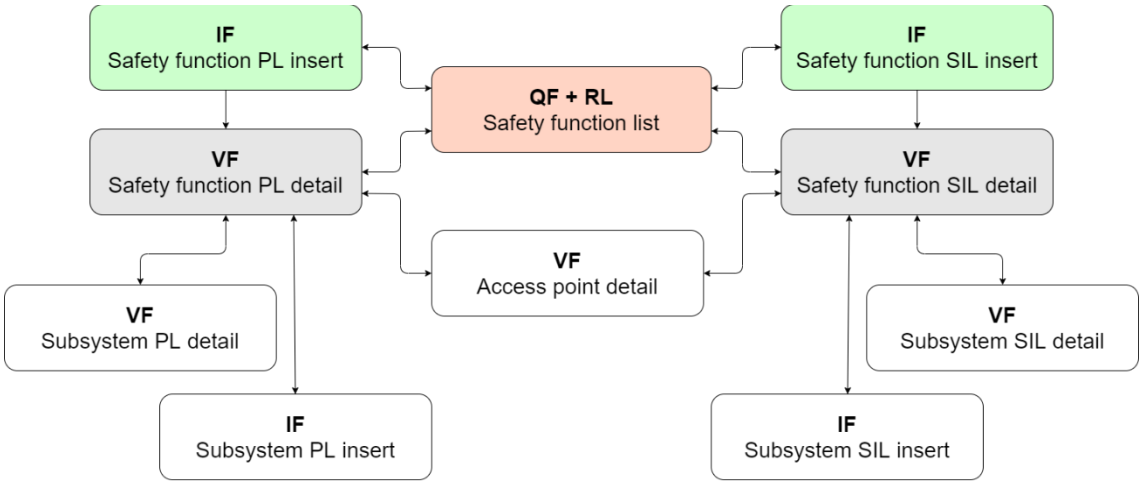
Tento modul slúži na tvorenie, upravovanie a prehliadanie záznamov z tabuľky Machine. Už len pohľadom na kostru dátového modelu môžeme vidieť, že samotná entita Machine využíva mnohé okolité tabuľky a preto je v rámci tohto modulu umožnená aj manipulácia s dátami z týchto okolitých tabuliek. Typickým príkladom môže byť plnenie tabuľky AccessPoint cez tento modul (konkrétne cez Machine detial formulár). Aj napriek tomu, že entita AccessPoint patrí medzi hlavné tabuľky v aplikácii, nemá svoj vlastný modul. Dôvodom je, že záznamy v tejto entite sú priamo závislé na záznamoch v entite Machine a preto je aj jej správa zabezpečovaná cez modul pre mašiny. Nasledujúci obrázok znázorňuje formuláre z ktorých je tento modul zložený. Jednotlivé šípky znázorňujú navigáciu medzi formulármi.



Obr. 19: Formuláre a ich vzájomná navigácia v module pre mašiny

Modul pre bezpečnostné funkcie

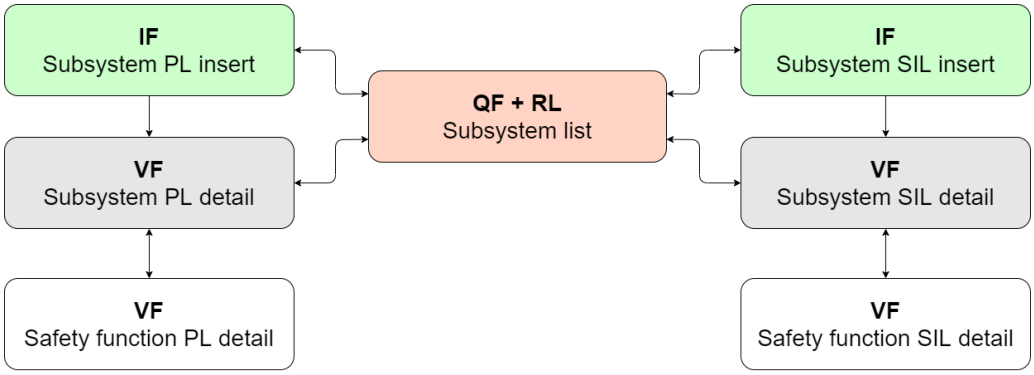
Modul sa stará o základné operácie nad entitou SafetyFunction ako je mazanie, úprava a zobrazovanie dát. Z obrázku číslo 19 ktorý znázorňuje konkrétnu podobu tohto modulu je vidieť, že je relatívne rozsiahly. Dôvodom je podpora pre dve rôzne metodiky (PL a SIL), ktoré sa častokrát príliš líšia na to, aby boli zastrešované spoločným formulárom.



Obr. 20: Formuláre a ich vzájomná navigácia v module pre bezpečnostné funkcie

Modul pre subsystémy

Posledný modul v aplikácii sa stará o správu subsystémov a pridružených entít. Modul musí opäť poskytovať formuláre pre dve rôzne metodiky. V rámci modulu je spravovaná aj entita Element, ktorá ako ďalšia z hlavných entít nemá svoj vlastný modul.



Obr. 21: Formuláre a ich vzájomná navigácia v module pre subsystémy

Z troch posledných obrázkov je zrejmé, ktoré formuláre slúžia v jednotlivých moduloch na vytváranie nových položiek (zelené bloky – IF), detailný náhľad na vybranú položku (šedé bloky – VF) a prezeranie zoznamu položiek (červený blok – QF + RL). Bystrý čitateľ si ale hneď všimne že nikde nie je spomenutá možnosť editácie a mazania položiek. Dôvodom je, že tieto operácie by bolo možné vykonávať na viacerých miestach (vo viacerých formulároch). Často teda záleží na konkrétnej potrebe koncového

užívateľa, alebo na presnej požiadavke od zákazníka. Pri návrhu tejto aplikácie sme mali dosť voľnosti a mohli sme si vybrať. Pre jednoduchosť je mazanie aj upravovanie možné cez detail daného záznamu, teda cez šedé bloky (VF) v obrázkoch vyššie. Konkrétne operácia mazania by mohla byť umožnená aj v zozname všetkých položiek (v červených blokoch), avšak kvôli väčšej bezpečnosti a kontrole nad tým, čo robí užívateľ toto riešenie aplikácia nezahrnuje.

2.5 Bezpečnosť informačného systému

Cieľom tejto práce nie je len vytvoriť testovaciu webovú aplikáciu, ale aplikáciu ktorá splňuje aspoň základné bezpečnostné predpoklady. K ich dosiahnutiu existuje veľa spôsobov a výsledná bezpečnosť dosť záleží aj na použitom frameworku (či na back-end alebo front-end). V rámci tejto aplikácie sme sa zamerali hlavne na ochranu užívateľských dát a zabezpečenie aplikácie pred neautorizovanými užívateľmi.

K procesom na dosiahnutie vyššie uvedených opatrení slúži:

- Autentifikácia – proces vytvárania užívateľský účtov a ich neskoršej identifikovanie na základe prihlasovacích údajov.
- Autorizácia – proces kontroly a prideľovania práv užívateľom.

Ešte predtým než sa bližšie pozrieme ako konkrétne sú v našej aplikácii použité vyššie uvedené procesy, povieme si niečo o bezpečnostnom prvku, ktorý by mala obsahovať každá moderná aplikácia. Tým je HTTPS protokol.

HTTPS (HyperText Transfer Procol Secure) predstavuje rozšírenie štandardného HTTP protokolu, ktorý sme si už zľahka predstavili v kapitole 2.3.2. Pre zopakovanie, HTTP definuje štandard pre komunikáciu a prenos dát medzi klientom a serverom v prostredí internetu. Väčšia bezpečnosť HTTPS protokolu spočíva v šifrovaní dát, ktoré sú pomocou neho prenášané. Toto šifrovanie zabezpečuje bezpečnostný protokol TLS (Transport Layer Security), ktorý je často zamieňaný s SSL (Secure Sockets Layer). Naša aplikácia je od jej vzniku na použitie HTTPS protokolu nakonfigurovaná. Nižšie uvedený obrázok pekne znázorňuje rozdiel medzi zabezpečeným a nezabezpečeným spojením.

Obr. 22: Rozdiel medzi HTTPS a HTTP protokolom

2.5.1 ASP.NET core Identity

ASP.NET core Identity predstavuje systém pre manažovanie a jednoduchú integráciu bezpečnostných prvkov ako autentifikácia a autorizácia do aplikácií, ktoré sú vytvárané v ASP.NET core frameworku. Tento systém poskytuje úplne všetko od potrebných databázových entít až po grafické rozhranie (komponenty ktoré je možné použiť). V našom prípade sme si grafické rozhranie vytvorili sami. Týmto grafickým rozhraním sú myslené formuláre pre prihlásenie a registráciu do aplikácie.

Konkrétne pre autorizáciu nám ASP.NET core Identity ponúka tri možnosti:

- Role-based autorizácia – každý užívateľ má jednu alebo viac rolí.
- Claim-based autorizácia – každý užívateľ má priradené nejaké „tvrdenia“, ktoré aplikácii hovoria o koho sa jedná.
- Policy-based autorizácia – umožňuje kombinovanie rolí a „tvrdení“, pričom vznikne policy (politika), ktorá zhlukuje celú konfiguráciu autorizácie pod jedno meno.

V rámci tejto aplikácie je využitá prvá možnosť z vyššie uvedených, čiže autorizácia založená na užívateľských roliach. Viac o konkrétnych roliach a ich právach si povieme v kapitole 2.5.3.

Ak ide o autentifikáciu (napr. proces prihlasovania sa do aplikácie) ASP.NET core Identity nám opäť ponúka viaceré možnosti. Jednou z nich je aj autentifikácia pomocou JWT (JSON Web Token). Čo to je a ako funguje si opíšeme v ďalšej časti tejto kapitoly.

2.5.2 JSON Web Token (JWT)

JWT je otvorený štandard (RFC 7519), ktorý definuje kompaktný spôsob bezpečného prenosu informácií vo formáte JSON medzi účastníkmi komunikácie. Tieto informácie možno overiť a dôverovať im, pretože sú digitálne podpísané. JWT môže byť podpísaný pomocou nejakého „tajomstva“ (napr. pomocou algoritmu HMAC) alebo páru verejných a súkromných kľúčov pomocou RSA alebo ECDSA algoritmu [11]. V dnešnej dobe sa stáva JWT stále viac a viac populárny hlavne vďaka jeho ľahkej integrácii do už existujúcej aplikácie. Samotný token sa skladá z troch časti, ktoré sú oddelené bodkou:

- Hlavička – typicky obsahuje údaj o type JWT tokenu a o šifrovacom algoritme (napr. HMAC, SHA256 atď.)
- Telo – obsahuje dáta vo formáte JSON. Tieto dáta bývajú najčastejšie údaje o užívateľovi ako napr. jeho meno, role a iné.
- Podpis – vzniká zašifrovaním kombinácie hlavičky a tela.

V nasledujúcich bodoch si stručne opíšeme ako funguje autentifikácia pomocou JWT a tým aj ako funguje autentifikácia v našom informačnom systéme, ktorý JWT využíva:

1. Ak sa chce užívateľ prihlásiť do aplikácie, pošle cez prihlasovací formulár HTTP request ktorý obsahuje jeho email a heslo.
2. Server vykoná všetky potrebné kontroly (existencia užívateľa v systéme, správnosť hesla atď.) a ak je všetko v poriadku, vygeneruje nový JWT token, ktorý v tele obsahuje informácie a role daného užívateľa. Tento token je potom vrátený klientovi v tele odpovede na HTTP request.
3. Po obdržaní odpovede na klientovi sa token z tela HTTP odpovede uloží do pamäte prehliadača. Zároveň sa tento token nastaví ako predefinovaný pre hlavičky všetkých ďalších HTTP requestov. To v preklade znamená, že pri akejkoľvek ďalšej komunikácii medzi klientom a serverom sa v hlavičke HTTP requestu vždy posiela obdržaný JWT token.
4. Pri každom ďalšom HTTP requeste server dešifruje hlavičku requestu v ktorej nájde JWT token. Na základe informácii ktoré nesie ďalej rozhodne či je daný užívateľ autentifikovaný alebo nie, prípadne či má prístup k operácii o ktorú sa daným requestom pokúša (napr. získanie dát, mazanie atď.).

2.5.3 Uživatelské práva a role

Ako už bolo spomenuté v predchádzajúcej kapitole, naša aplikácia využíva autorizáciu založenú na právach a roliach. Každý užívateľ môže mať jednu alebo viac rolí, ktoré určujú, čo všetko môže robiť a k akým dátam má prístup. Kontrola užívateľových práv na základe jeho rolí je vykonávaná na back-ende aj na front-ende.

V základnej verzii naša aplikácia obsahuje nasledujúce role:

- Pozorovateľ:
  - Počiatočná rola každého nového užívateľa, ktorý sa zaregistruje.
  - Môže vidieť všetky vytvorené bezpečnostné funkcie, mašiny aj subsystémy ale nemôže ich upravovať.
  - Nemôže vytvárať nové záznamy v databáze (napr. nové mašiny atď.).
- Normálny užívateľ:
  - Môže vidieť všetky vytvorené bezpečnostné funkcie, mašiny aj subsystémy ale upravovať môže iba svoje záznamy.
  - Môže vytvárať nové záznamy v databáze (napr. nové mašiny atď.).
- Admin pre bezpečnosť:
  - Udržuje poriadok v záznamoch, ktoré sa týkajú bezpečnosti.
  - Môže vidieť všetky vytvorené bezpečnostné funkcie, mašiny aj subsystémy a v prípade potreby ich aj upravovať.
  - V prípade potreby môže mazať cudzie záznamy.
- Admin pre správu užívateľov a rolí:
  - Môže upravovať role jednotlivých užívateľov, prípadne ich blokovať.

3 ZÁVER

Cieľom tejto bakalárskej práce bolo navrhnutie a vytvorenie testovacej webovej aplikácie, ktorá by mala uľahčovať vyhodnocovanie bezpečnosti pre strojné zariadenia.

V úvode sme sa zoznámili s problematikou a predstavili sme si najpodstatnejšie normy z oblasti bezpečnosti strojných zariadení. Medzi najdôležitejšie patria normy EN ISO 13849-1 a EN 62061. Tieto normy nám následne v druhej časti tejto práce poskytovali kvalitný teoretický základ, od ktorého sme sa mohli ďalej odvíjať.

Návrh aplikácie sme začali opisom najdôležitejších procesov, ktoré sa v aplikácii odohrávajú. V tej istej kapitole sme si zostavili aj jednoduché algoritmy, ktoré nám poskytujú prehľad o postupe zberu potrebných údajov, ktoré musí užívateľ do aplikácie vkladať. Následne sme na základe teoretických vedomostí z prvej kapitoly navrhli dátový model, ktorý slúži na ukladanie všetkých potrených dát a informácií. Tento dátový model bol v neskorších štádiách vývoju realizovaný v MSSQL databáze.

V ďalších častiach tejto práce sme sa venovali už konkrétnej realizácii serverovej a front-endovej časti aplikácie. Na vývoj oboch častí nám slúžilo Visual Studio 2019. Zdrojový kód ktorý je výsledkom tejto práce je možné nájsť v prílohe C.

V kapitole o back-ende aplikácie sme si okrem použitého frameworku opísali aj zvolenú architektúru a jej výhody a nevýhody. Následne sme si predstavili jednotlivé vrstvy zvolenej architektúry a ich súčasti.

Druhú veľkú časť aplikácie predstavuje front-end a preto aj nemu bola venovaná samostatná kapitola. Na úvod sme si opäť predstavili použitý framework a následne sme sa hneď venovali návrhu modulov a opisu jednotlivých formulárov. Aj napriek tomu že samotná kapitola nie je veľmi rozsiahla, práve realizácia front-endu tvorila najväčšiu časť z celkového vývoja aplikácie.

V úplnom závere tejto práce sme si povedali niečo málo o bezpečnosti nášho informačného systému. Keďže táto téma je vo všeobecnosti veľmi rozsiahla a zaslúžila by si samostatnú prácu, zamerali sme sa hlavne na jej dve základné časti, a to autentifikáciu a autorizáciu.

Potom, ako bola celá aplikácia realizovaná, bola nasadená na školský server, čím sa stala prístupnou na testovanie kompetentnými osoba, ktoré by mohli byť v budúcnosti jej potenciálnymi užívateľmi. Toto testovanie ešte len odhalí, ako kvalitne bola táto práca urobená a či je aplikácia použiteľná v praxi. Je treba však povedať, že výsledná aplikácia je len základnou verziou toho čím by v skutočnosti mohla byť a počas vývoja sa objavili mnohé nápady a možnosti na jej rozšírenia. Väčšina týchto nápadov však bola len pridaná na zoznam budúcich vylepšení z dôvodu časovej náročnosti ich realizácie.

LITERATÚRA

[1] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Návrh systému podle (EN) ISO 13849]

[2] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Diagnostické pokrytí (DC)]

[3] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Selhání se společnou příčinou (CCF)]

[4] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Kap. 8: Návrh systému podle IEC/EN 62061]

[5] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Pravděpodobnost nebezpečného selhání za hodinu ($MTTF_D$)]

[6] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Podíl bezpečných selhání (SFF)]

[7] *Bezpečnostní řídící systémy pro strojní zařízení: zásady, normy a implementace (revize 5 řady publikací Safebook)*. 5. Praha: Rockwell Automation, 2016. Strojní vybavení (Rockwell). SAFEBK-RM002-CS-P. [Kap. 8: Tolerance poruchy hardwaru]

[8] Informačný systém. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 20.10.2016 [cit. 2021-5-12]. Dostupné z: https://sk.wikipedia.org/wiki/Informa%C4%8Dn%C3%BD_syst%C3%A9m

[9] Separation of concerns. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 18.01.2021 [cit. 2021-5-12]. Dostupné z: https://en.wikipedia.org/wiki/Separation_of_concerns

[10] LOCK, Andrew. *ASP.NET Core in action*. Shelter Island: Manning Publications Co., 2018. ISBN 9781617294617.

[11] JWT. Introduction. *jwt.io* [online]. ©2021 [cit. 2021-5-12]. Dostupné z: https://jwt.io/introduction

ZOZNAM SYMBOLOV A SKRATIEK

Skratky:

PL	Úroveň vlastností
PL_r	Minimálna požadovaná úroveň vlastností
SRP/CS	Bezpečnostné časti ovládacieho systému
$MTTF_D$	Stredná doba do nebezpečnej poruchy
DC	Diagnostické pokrytie
CCF	Opatrenia proti zlyhaniu so spoločnou príčinou
SIL	Úroveň integrity bezpečnosti
PFH_D	Pravdepodobnosť nebezpečného zlyhania za hodinu
SFF	Podiel bezpečných porúch
HFT	Odolnosť proti vadám hardwaru
SRCF	Riadiaca funkcia súvisiaca s bezpečnosťou
ERD	Entity-relationship diagram
FK	Cudzí kľúč
CR30	Relé modul s nastaviteľnými parametrami
GMX	Menšie bezpečnostné PLC
GLX	Bezpečnostné PLC
HMI	Rozhranie človek-stroj
PLC	Programovateľný logický kontrolér
CFF	Iný názov pre beta faktor
T1	Interval kontrolnej skúšky
T2	Interval diagnostickej skúšky
C	Počet zopnutí za hodinu
B10d	Stredný počet cyklov do doby kedy 10% súčastí zlyhá
SI	Počet štandardných vstupov
SO	Počet štandardných výstupov
ID	Unikátny identifikátor
SoC	Separation of Concerns
API	Application Programming Interface
URL	Uniform Resource Locator
HTTP	HyperText Transfer Protokol
HTTPS	HyperText Transfer Protokol Secured
JSON	JavaScript Object Notation
XML	eXtensible Markup Language
BL	Bussiness Logic Layer
DAL	Data Access Layer
JS	JavaScript
HTML	Hyper Text Markup Language

CSS	Cascading Style Sheets
WASM	WebAssembly
QF	Query Form
RL	Record List
VF	View Form
IF	Insert Form
TLS	Transport Layer Security
SSL	Secure Sockets Layer
JWT	JSON Web Token

Symboly:

β	Beta faktor
λ_{DTOTAL}	Pravdepodobnosť všetkých nebezpečných zlyhaní
λ_{DD}	Množstvo detekovaných nebezpečných zlyhaní
λ_{D}	Množstvo všetkých nebezpečných zlyhaní
λ_{S}	Množstvo bezpečných zlyhaní
T_1	Interaval kontrolnej skúšky
T_2	Interval diagnostickej skúšky

ZOZNAM PRÍLOH

Príloha A - Tabuľky k norme EN ISO 13849

A.1 Zjednodušený postup pre odhad úrovne vlastností (PL)

Kategória	B	1	2	2	3	3	4
DC_{AVG}	žiadne	žiadne	nízke	stredné	nízke	stredné	vysoké
$MTTF_D$							
krátka	a	-	a	b	b	c	-
stredná	b	-	b	c	c	d	-
dlhá	-	c	c	d	d	d	e

A.2 Postup pre určenie požadovanej úrovne vlastní (PL_r)