České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství
Obor: Aplikace softwarového inženýrství

Nástroj pro nastavení systému triggerů experimentu COMPASS

Configuration Tool For the Trigger System of the COMPASS experiment

BAKALÁŘSKÁ PRÁCE

Vypracoval: Jan Vondruška
Vedoucí práce: Ing. Vladimír Jarý, Ph. D.
Rok: 2022

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství

Akademický rok 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Jan Vondruška**
Studijní program:	**Aplikace přírodních věd**
Obor:	**Aplikace softwarového inženýrství**
Název práce česky:	**Nástroj pro nastavení systému triggerů experimentu COMPASS**
Název práce anglicky:	**Configuration Tool for the Trigger System of the COMPASS experiment**

Pokyny pro vypracování:

1. Seznamte se se systémem pro sběr dat experimentu COMPASS v laboratoři CERN.
2. Seznamte se s nástroji na vývoj webových informačních systémů.
3. Analyzujte stávající desktopovou aplikaci pro nastavení systému triggerů experimentu
4. Proveďte analýzu požadavků kladených na webový nástroj pro nastavení systému triggerů experimentu.
5. S použitím vhodných nástrojů navržený systém implementujte.

Doporučená literatura:

[1] ABBON, P. et al. *The COMPASS experiment at CERN.* Nucl. Instrum. Meth. A577 (2007) 455-518.

[2] SRINIVASAN, M. *Web Technology: Theory and Practice.* Delhi: Pearson Education India, 2012. ISBN 978-933-2508-194.

[3] ESPOSITO, D. *Modern Web Development: Understanding domains, technologies, and user experience (Developer Reference).* Microsoft Press, 2016. ISBN 978-150-9300-013.

[4] ZEMKO, M. et al. Free-running data acquisition system for the AMBER experiment. In: *EPJ Web of Conferences 251*, 04028 (2021)

Jméno a pracoviště vedoucího práce:

Ing. Vladimír Jarý, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Jméno a pracoviště konzultanta:

Ing. Martin Zemko
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Zemko Vladimír J.
.....
vedoucí práce

Datum zadání bakalářské práce: 15. 10. 2021

Termín odevzdání bakalářské práce: 7. 7. 2022

Doba platnosti zadání je dva roky od data zadání.

.....
garant oboru

.....
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská
-1-

.....
děkan

V Praze dne 15. 10. 2021

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ....

Jan Vondruška

Poděkování

Rád bych touto cestou vyjádřil poděkování Ing. Vladimíru Jarému, Ph.D. za jeho cenné rady a doporučení při vedení mé bakalářské práce. Taktéž bych chtěl poděkovat Ing. Martinu Zemkovi za vstřícnost, ochotu a pomoc při získání potřebných informací a podkladů.

Jan Vondruška

Název práce:
Nástroj pro nastavení systému triggerů experimentu COMPASS

Autor: Jan Vondruška

Obor: Aplikace softwarového inženýrství

Druh práce: BAKALÁŘSKÁ PRÁCE

Vedoucí práce: Ing. Vladimír Jarý, Ph. D.
Katedra softwarového inženýrství,
Fakulta jaderná a fyzikálně inženýrská,
České vysoké učení technické v Praze

Konzultant: Ing. Martin Zemko
Katedra softwarového inženýrství,
Fakulta jaderná a fyzikálně inženýrská,
České vysoké učení technické v Praze

Abstrakt: Cílem práce je návrh a implementace webového nástroje pro úpravu nastavení systému triggerů pro experimenty COMPASS a AMBER v laboratoři CERN. Tato aplikace by měla usnadnit úpravu nastavení a následně ho uložit do databáze. Kromě toho nabízí responzivní vzhled pro různé typy obrazovek a lze ji využívat i mimo síť COMPASS. Nejprve se práce zabývá seznámením s laboratoří CERN, experimentem COMPASS a jeho nástupcem AMBER a systémem pro sběr dat těchto experimentů. Dále jsou v práci popsány webové technologie, které byly při vývoji použity. Následně je provedena analýza stávající desktopové aplikace pro nastavení systému triggerů, která je k tomuto účelu v současnosti používána na experimentu COMPASS. Poté je představen návrh webové aplikace, který byl proveden na základě výsledků analýzy požadavků. Práce je zakončena podrobným popisem jeho implementace.

Klíčová slova: CERN, AMBER, COMPASS, DAQ, PHP

Title:
Configuration Tool For the Trigger System of the COMPASS experiment

Author: Jan Vondruška

Abstract: The goal of this project is to propose and implement web application for the configuration of the trigger system for the COMPASS and AMBER experiments at CERN laboratory. This application will simplify the modification of the configuration and will also support the storage in the database. Besides, it will offer responsive design for various screen widths and will be available outside the COMPASS network. First of all, the thesis deals with the introduction of CERN laboratory, COMPASS experiment and its successor AMBER and data acquisition system of both experiments. Next, the web technologies which were used during the development are described. The analysis of the desktop application which is used for that purpose on the COMPASS experiment then follows. Afterwards, the proposal of the web application which was made with respect to the result of the requirement´s analysis is presented. The last part provides detailed description of implementation of the proposal.

Key words: CERN, AMBER, COMPASS, DAQ, PHP

Obsah

Úvod **10**

1 CERN a experiment COMPASS **11**

1.1 CERN 11

1.1.1 Obecné informace 11

1.1.2 Urychlovače v CERN 11

1.2 COMPASS 12

1.2.1 Historie 12

1.2.2 Obecné informace 13

1.3 AMBER 13

2 Systém pro sběr dat experimentu COMPASS **15**

2.1 Hardwarová část 15

2.2 Softwarová část 16

3 Webové technologie **18**

3.1 Historie internetu 18

3.2 Síť 19

3.3 TCP/IP 19

3.4 HTTP 19

3.5 Webový server 20

3.5.1 Statický webový server 20

3.5.2 Dynamický webový server 20

3.5.3 Princip webového serveru 21

3.6 Použité webové nástroje při vývoji webové aplikace 21

3.6.1 Databáze 21

3.6.2 Relační databáze . . . 22
3.6.3 SQL . . . 23
3.6.4 SGML . . . 24
3.6.5 HTML . . . 24
3.6.6 Kaskádové styly . . . 26
3.6.7 JavaScript . . . 27
3.6.8 PHP . . . 29

4 Stávající desktopová aplikace **31**
4.1 Qt framework . . . 31
4.2 Desktopová aplikace . . . 31

5 Požadavky kladené na webový nástroj pro nastavení systému triggerů **34**

6 Návrh a implementace webového nástroje **36**
6.1 Třída HltDatabase . . . 36
6.2 HTML a PHP šablony . . . 37
6.2.1 Úvodní okno . . . 37
6.2.2 Vytváření konfiguračního profilu . . . 38
6.2.3 Správa nastavení profilů . . . 38
6.2.4 Záznamy . . . 42
6.3 Back-endové skripty . . . 43
6.3.1 Skript pro vytváření profilu . . . 43
6.3.2 Skripty pro správu nastavení profilů . . . 43
6.3.3 Skript pro dynamické načítání záznamů . . . 44
6.4 Testování . . . 44

Závěr **45**

Literatura **46**

Přílohy **50**

A Ukázka skriptu general_settings_script.php **50**

B Er diagram databáze HLT **54**

C Metody třídy HltDatabase **55**

C.1 Metody využívající SQL příkaz SELECT 55

C.2 Metody využívající SQL příkaz INSERT 57

C.3 Metody využívající SQL příkaz UPDATE 58

C.4 Metody využívající SQL příkaz DELETE 59

D Používané zkratky **60**

E Obsah přiloženého CD **62**

Úvod

V moderní částicové fyzice je potřeba mít systém pro sběr dat, jehož součástí bývá také trigger systém, který slouží pro výběr fyzikálně zajímavých událostí. Tato práce se bude zabývat systémem využívaným na experimentu COMPASS v laboratoři CERN.

V současné době v laboratoři CERN na experimentu COMPASS neexistuje žádný webový nástroj, který by sloužil k úpravě nastavení systému triggerů. Existuje zde pouze desktopová aplikace, která spouští vysokoúrovňový trigger HLT (high level trigger) a obsluhuje správu nastavení tohoto systému a jeho ukládání do databáze. Cílem této práce je navrhnout a implementovat vhodný webový nástroj, který by usnadnil uživatelům správu nastavení systému triggerů.

První kapitola se zabývá laboratoří CERN a experimentem COMPASS a jejich historií. Následně v druhé kapitole je stručně popsán systém sběru dat tohoto experimentu (DAQ) a jeho vývoj.

V další kapitole je představen vznik a vývoj internetu a webu, jak je známe dnes, a krátký popis webových technologií, které byly použity při vývoji webové aplikace.

Čtvrtá kapitola obsahuje obecné seznámení s frameworkem Qt, ve kterém je napsána již existující desktopová aplikace. Dále se věnuje popisu desktopové aplikace z funkčního i grafického hlediska.

V předposlední kapitole jsou rozebrány požadavky kladené na webový nástroj. Jsou zde uvedeny požadavky na funkčnost dané aplikace a na technologie používané při návrhu vzhledu a funkčnosti jak pro část back-endovou, tak pro front-endovou.

Poslední šestá kapitola obsahuje nejdůležitější část této práce. Zabývá se popisem implementace samotné webové aplikace. Je zde rozebrán návrh vzhledu, třída HltDatabase pro práci s databází, PHP skripty používané jak pro část front-endovou, tak i back-endovou, a využití technologie AJAX pomocí JavaScriptové knihovny jQuery. Kapitola je uzavřena popisem testování implementované webové aplikace.

Kapitola 1

CERN a experiment COMPASS

1.1 CERN

1.1.1 Obecné informace

Po druhé světové válce, při které spousta evropských vědců emigrovala do Ameriky, se mnozí vědci z Evropy a Ameriky shodli, že Evropa potřebuje ústav pro výzkum Fyziky na světové úrovni. V prosinci roku 1951 na zasedání UNESCO bylo přijato usnesení o zřízení Evropské rady pro jaderný výzkum. O dva měsíce později byla podepsána dohoda o zřízení prozatímní rady a zrodila se zkratka CERN (z francouzského Conseil Européen pour la recherche nucléaire). V roce 1953 byl odsouhlasen konečný návrh úmluvy CERN původními 12 členskými státy. Dnes má CERN 23 členských států. [1]

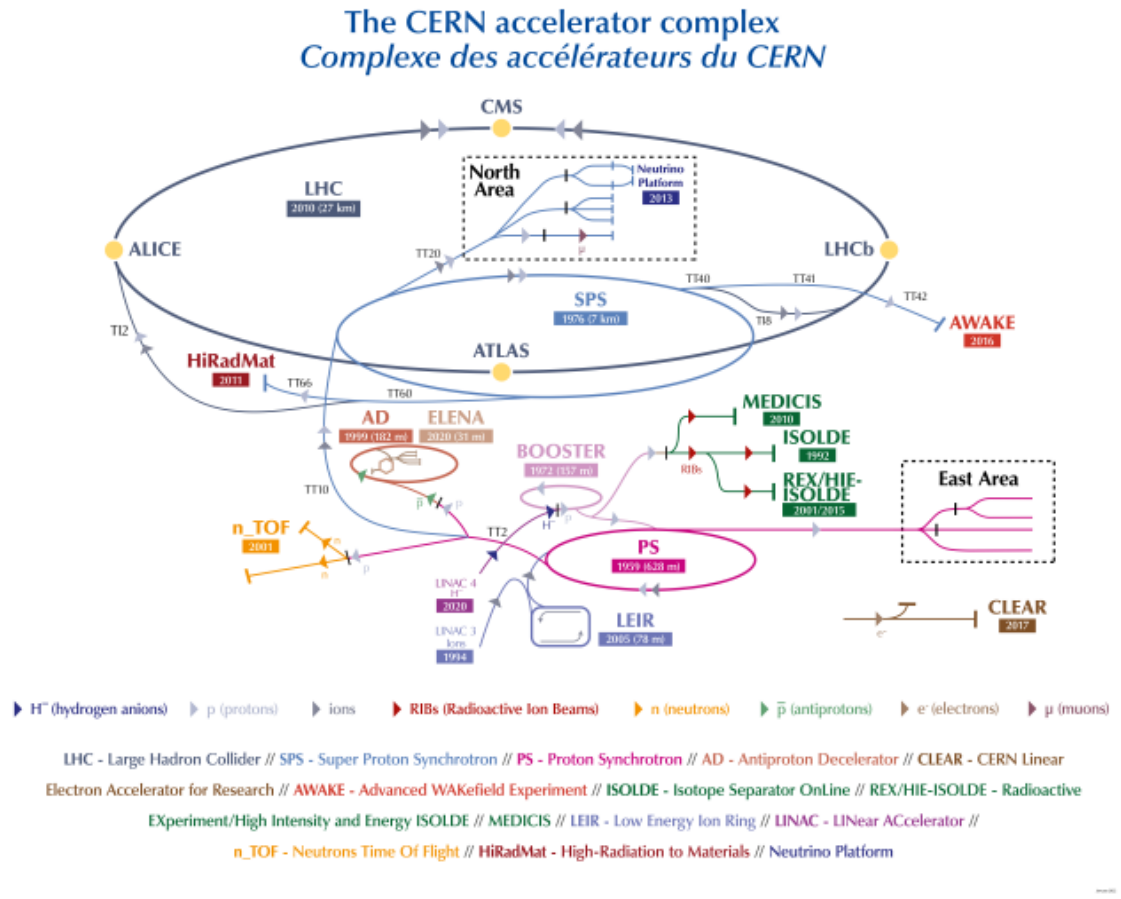
CERN (Evropská organizace pro jaderný výzkum) je mezinárodní organizace sídlící v Ženevě. Laboratoře leží nedaleko Ženevy na francouzsko-švýcarské hranici. Převážnou část výzkumu tvoří oblast částicové fyziky, ale výsledky mají i v oblasti informačních technologiích. V minulém století zde byl vytvořen World Wide Web. [1]

Cílem je spolupráce evropských zemí ohledně fyzikálního výzkumu. Proto organizace přijala politiku otevřeného přístupu. To znamená, že veškeré výsledky z experimentů jsou zpřístupněny veřejnosti. [1]

1.1.2 Urychlovače v CERN

CERN ke svému výzkumu používá komplex s 8 urychlovači a 2 zpomalovači částic. Urychlovače slouží jako doplňky k experimentům nebo urychlují částice pro větší urychlovače, které slouží k samotnému experimentu. Komplex obsahuje i největší urychlovač částic na světě zvaný Velký Hadronový Urychlovač (LHC). [1]

Komplex urychlovačů částic je posloupností zařízení stále urychlujících částice. Každé zařízení urychlí paprsek částic, než jej pošle do dalšího. Koncovým prvkem je LHC urychlovač. [1]



Obrázek 1.1: Komplex urychlovačů částic v laboratoři CERN 2022 [2]

1.2 COMPASS

1.2.1 Historie

COMPASS (Common Muon and Proton Apparatus for Structure and Spectroscopy) byl navržen a schválen v CERNU roku 1997 a od roku 1998 začalo jeho sestavování. V roce 2001 proběhl zkušební běh a od roku 2002 probíhají samotné experimenty. Sběr dat pokračoval do roku 2004. V roce 2005 došlo k roční přestávce. Experiment poté pokračoval v letech 2006 a 2007, kdy byl obnoven sběr dat pomocí mionového[1] paprsku. Do roku 2008 se prováděly experimenty pouze s mionovými paprsky a od tohoto roku započaly experimenty a výzkumy na Hadronovou spektroskopii s pionovými a protonovými paprsky. V roce 2011 bylo rozhodnuto o prodloužení experimentu COMPASS a ten přešel do 2. fáze s označením COMPASS II. [4] [5]

V roce 2012 byl experiment zaměřen na Primakovovy reakce. Od roku 2015 probíhalo měření na Drell-Yanův proces s paprskem negativních pionů a polarizovaným protonovým terčem. [4] [5]

[1] Mion je elementární subatomární částice podobná elektronu, ale 200-krát těžší. Má dvě podoby: záporně nabitý mion a pozitivně nabitý antimion. Byl objeven v kosmickém záření. [3]

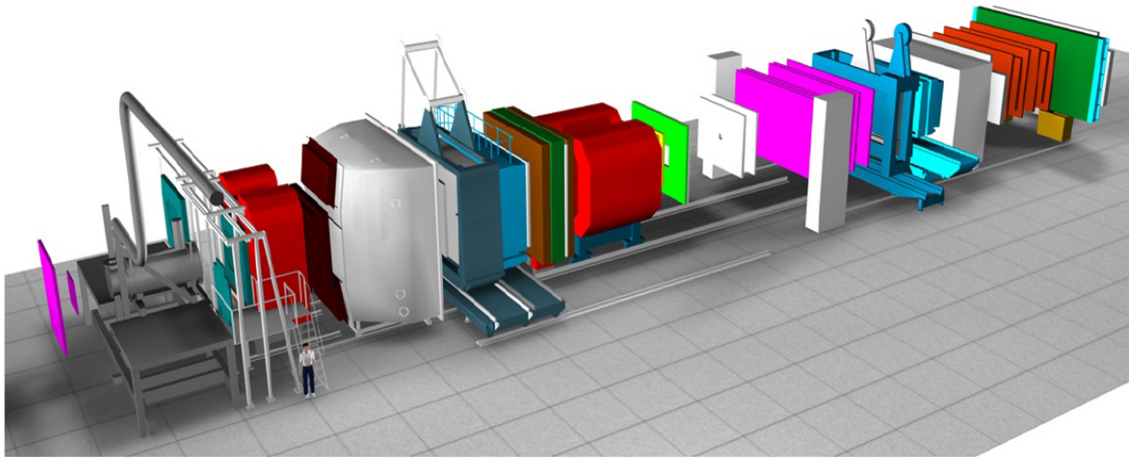
1.2.2 Obecné informace

COMPASS je víceúčelový experiment na urychlovači Super Proton Synchroton, který je druhým největším urychlovačem v komplexu urychlovačů v CERNu. Zabývá se komplexními způsoby, jakými spolu kvarky a gluony spolupracují, aby vznikly námi pozorované částice jako protony a spousta dalších. Hlavním cílem je studium vnitřní struktury protonů a neutronů a snaha zjistit více informací o tom, jak v nich vzniká vlastnost spin, konkrétně jak moc přispívá pohyb kvarků nebo gluonů, které spojují kvarky. [4]

Hlavní budova experimentu COMPASS se nachází v severní části kampusu CERN nedaleko francouzského města Prévessin-Moëns. Nyní na experimentu pracuje okolo 220 vědců ze 13 zemí a 25 institucí. [4]

Experiment poběží naposledy v letech 2021-2022 a bude nahrazen svým nástupcem nazvaným experiment AMBER. [4]

Na obrázku vidíme schéma komplexu detektorů experimentu COMPASS, který má na délku okolo 60 metrů. Úplně vlevo je terč, na který se střílí, a vpravo od něj je soustava detekorů, se kterými se dá pohybovat a v případě potřeby některé i odebrat. [4]



Obrázek 1.2: Schéma detektorů experimentu COMPASS [6]

1.3 AMBER

V prosinci 2020 byla schválena 1. fáze nového experimentu s názvem AMBER (Apparatus for Meson and Baryon Experimental Research), který bude nástupcem experimentu COMPASS. AMBER bude stavět na již existujících komponentách, které doplní o nové detektory a cíle a o modernější technologie sběru dat. [7]

V první fázi experimentu jsou plánovány tři druhy měření.

Při prvním měření se budou miony střílet na vodíkový terč, pomocí čehož se určí s vysokou přesností poloměr protonu. To pomůže vyřešit otázku z roku 2010, kdy se nové výsledky měření lišily od dřívějších hodnot. [7]

Při druhém měření budou protony stříleny na protonové cíle a na cíle z izotopu těžšího helia-4, aby se mohla lépe prozkoumat rychlost vzniku antiprotonů[2] při těchto srážkách, což přispěje k přesnějšímu určení toku antiprotonů v kosmickém záření. [7]

Při třetím budou piony[3] stříleny na jaderné cíle a bude se měřit rozložení hybnosti kvarků a gluonů, které tvoří pion. [7]

Předpokládaný start experimentu je rok 2022, kdy skončí poslední běh experimentu COMPASS. [7]

[2]Antiproton je subatomární částice stejně těžká jako proton s opačným nábojem a opačně orientovaným magnetickým momentem. [8]

[3]Pion je nejlehčím mesonem a důležitou součástí kosmického záření. Vyskytuje se ve třech formách: neutrální, kladně nabitý nebo záporně nabitý. Skládá se z kvarků a gluonů. [9]

Kapitola 2

Systém pro sběr dat experimentu COMPASS

Důležitou součástí fyzikálních experimentů je sběr dat. Na experimentu COMPASS se o to stará DAQ (Data Acquisition System).

2.1 Hardwarová část

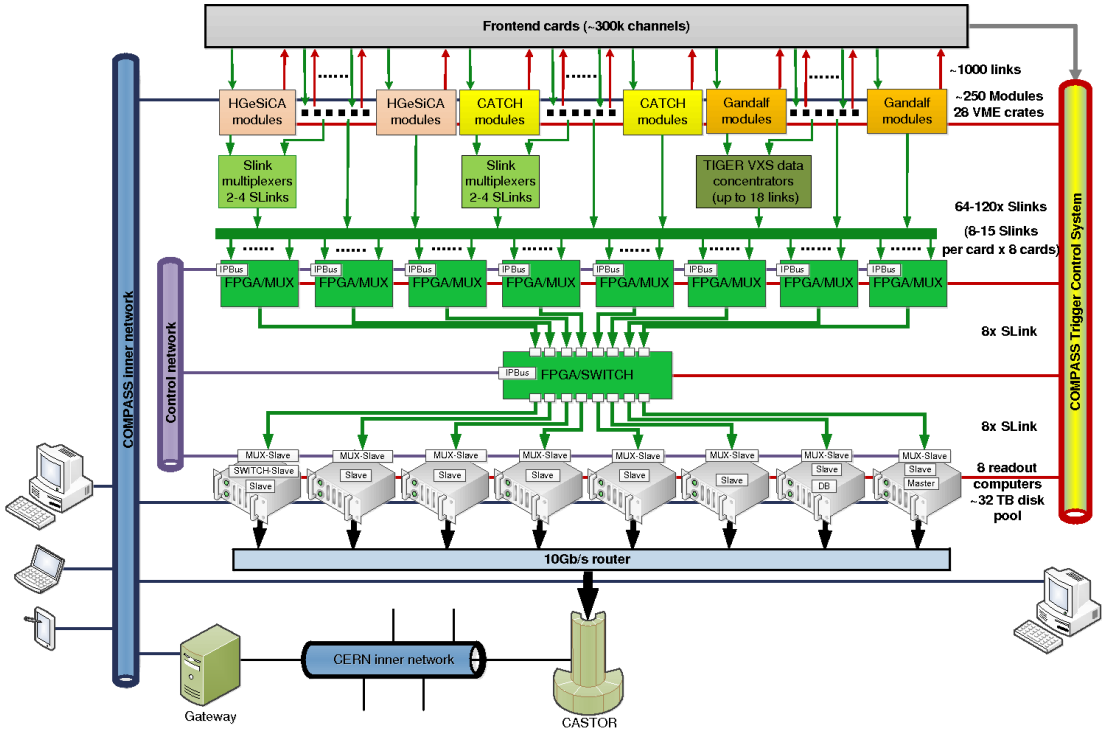
Současný systém se skládá z několika vrstev. Na nejnižší úrovni se nachází přední elektronika detektorů (tzv. Frontendové karty). Ty kontinuálně předzpracovávají, rozlišují a digitalizují data z detektorů. Tato vrstva se skládá zhruba ze 300 000 datových kanálů. Data z těchto kanálů jsou načítána a shlukována pomocí specializovaných VME karet nazývaných CATCH, GeSICA a GANDALF, které se označují jako koncentrátory. Ačkoli na nejnižší úrovni dochází neustále ke tvorbě dat, čtení pomocí koncentrátorů se provádí pouze tehdy, pokud spouštěcí řídící systém TCS (Trigger control systém) vybere fyzikálně zajímavou událost (rozhodnutí je založeno na množství vložené energie v kalorimetrech a na signálech z hodoskopů[1]). TCS dále události přiřadí časové razítko a vlastní identifikátor. Koncentrátory slouží k vytváření dílčích událostí sběrem z různých datových kanálů. [10] [11]

Dříve dílčí události poté putovaly přes optickou linku S-Link do vyrovnávací paměti (ROB). ROBy jsou standardní servery vybavené 4 spillbuffer PCI kartami. Z této vrstvy byly posunuty do další etapy, která se skládá z počítačů nazvaných stavitelé událostí (event builders - EVB). ROB a EVB jsou připojeny do sítě Gigabit Ethernet network. EVB sestavují z dílčích událostí úplné události a katalogové záznamy. Katalogové záznamy jsou ukládány do databáze ORACLE a události jsou odeslány do trvalého úložiště CERN nazvané CASTOR. [10] [11]

[1] Hodoskop je nástroj používaný v detektorech k detekci procházejících nabitých částic a k určení jejich trajektorie. [12]

Okolo roku 2014 byla EVB síť nahrazena speciálními FPGA[2] DHC (data handling card) kartami. Jsou použity pro dvě funkce podle 2 fází procesu stavby událostí. V první z nich multiplexuje[3] až 120 příchozích spojení na 8 odchozích poskytující další úroveň koncentrace dat. Druhá obsahuje pouze jednu DHC kartu, která obstarává vytváření a distribuci událostí. Události jsou pak poslány do čtecích počítačů, kde jsou přijmuty přes Spillbuffer karty, přes DMA[4] zkopírovány do RAM. Dále jsou převedeny do formátu DATE kvůli kompatibilitě s původní architekturou a dočasně uloženy na lokálních discích, než jsou přesunuty na hlavní úložiště CASTOR. [10] [11]

Současná architektura systému DAQ je vidět na obrázku.



Obrázek 2.1: Architektura systému DAQ [13]

2.2 Softwarová část

Softwarová část se skládá z několika procesů: Master, Slave-control, Slave-readout, Runcontrol GUI, MessageLogger a MessageBrowser. Proces Master má úlohu prostředníka mezi GUI a Slave procesy a také Slave procesy monitoruje. Navíc provádí kontrolu běhu, zpracování chyb a výměnu dat s konfigurační databází. [11]

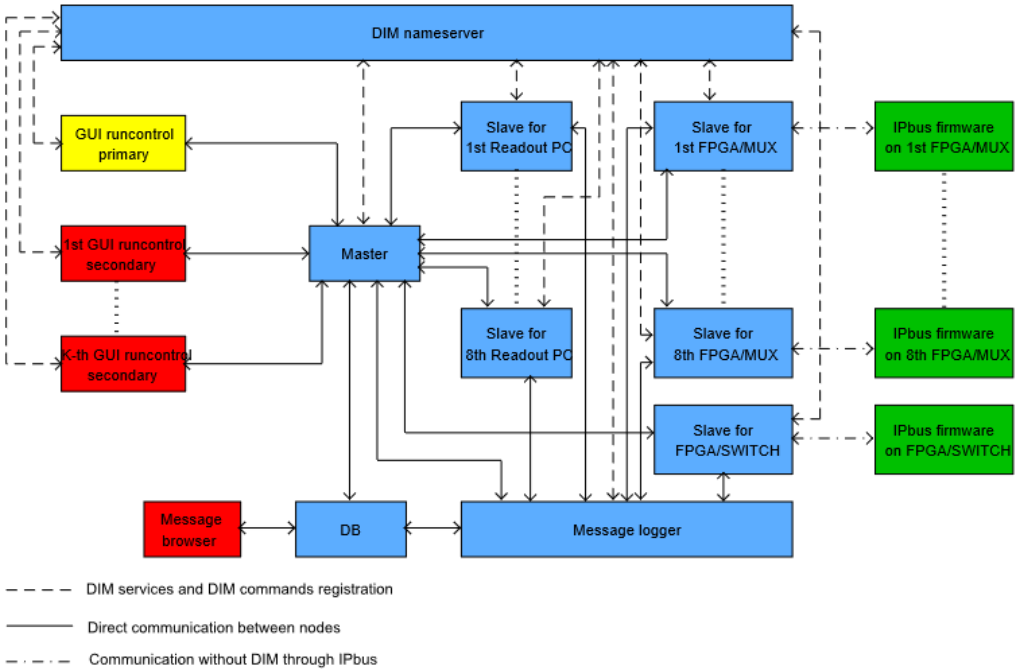
[2]FPGA (programovatelná hradlová pole) je typ elektronického integrovaného obvodu vyrobeného tak, že může být naprogramován až u uživatele.

[3]Multiplexování značí termín, kdy je více signálů kombinováno do jednoho signálu.

[4]DMA je způsob přímého přenosu dat mezi operační pamětí a výstupními zařízeními.

Každý Slave-control proces má na starost jednu FPGA kartu přes přístup k registrům pomocí IPBus. Konfiguruje FPGA, monitoruje zdroje FPGA a informuje proces Master o stavu připojených karet FPGA. Celý systém obsahuje 17 Slave-control procesů, které budou rozděleny na čtecích počítačích. [11]

Všechny softwarové části jsou vidět na obrázku.



Obrázek 2.2: Softwarová část systému DAQ [11]

Kapitola 3

Webové technologie

3.1 Historie internetu

První praktické schéma internetu jako „intergalaktickou síť počítačů“ navrhl na počátku 60. let J. C. R. Licklider z univerzity MIT. Nedlouho poté byl vyvinut koncept přepínání paketů[1], což je metoda pro přenos elektronických dat, která se stala jedním ze stavebních kamenů pro samotný internet. [14]

První funkční model internetu byl navržen koncem 60. let s vytvořením ARPANETu (Advanced Research Projects Agency Network), který byl financován ministerstvem obrany USA. Používal přepínání paketů, aby mohlo více počítačů v jedné síti komunikovat. [14]

Práce na ARPANETu pokračovala i v 70. letech, kdy vědci Robert Kahn a Vinton Cerf přišli s komunikačním modelem TCP/IP (Transmission Control Protoccol). Je to určitý souhrn pravidel pro spojování počítačů v sítích. Podle tohoto protokolu dnes pracuje celý internet. [15]

Podoba webu, jak ho známe dnes, přišla v roce 1989. Začalo to na místě, kde by to čekal málokdo - v CERNu.

Britský vědec Tim Berners-Lee pracující v počítačových službách v CERNu přišel s nápadem umožnit vědcům z navzájem vzdálených pracovišť organizovat a shromažďovat informace. Nikoliv pouze stahovat soubory do jednotlivých počítačů, ale aby zároveň text obsažený v samotných souborech mohl být upravován a spojován vzdáleně. Proto vytvořil v roce 1989 World Wide Web (WWW). K tomu potřeboval vyvinout pár dalších „jednoduchých“ věcí do začátku. Vyvinul si vlastní jednoduchý protokol HTTP (Hyper Text Transfer Protocol) pro získávání textu jiných dokumentů přes hypertextové odkazy. Textový formát, který si vytvořil, je dnes proslulý značkovací jazyk HTML (Hypertext Markup language). [1]

30. dubna 1993 spustil World Wide Web na veřejné doméně a pozdější ho zpřístupnil pod veřejnou licencí, aby umožnil jeho maximální šíření do světa.

[1] Paket je malý dílek z celkové zprávy odeslaný přes počítačovou síť. Pakety jsou poté pospojovány do původní zprávy počítači, které je přijaly. [16]

3.2 Síť

Síť je skupina počítačů, které jsou vzájemně propojeny ať už fyzicky nebo virtuálně. Internet spojuje spousty sítí, které se nazývají uzly. Máme dva hlavní typy sítí. LAN (local area network), neboli místní síť, se obvykle vztahuje k nějaké budově, kde jsou propojeny 3 až stovky počítačů. WAN (wide area network) je síť, která spojuje více různě geograficky rozmístěných sítí LAN. Nejznámější WAN sítí je samotný internet. [17]

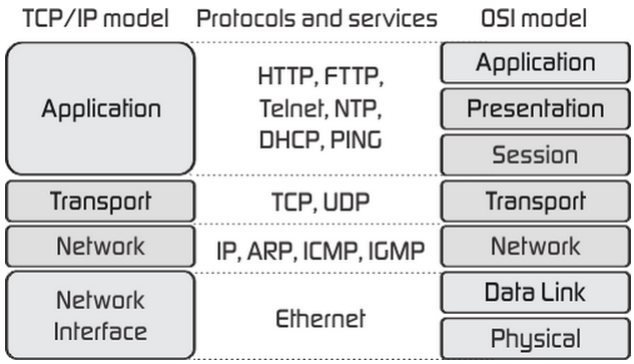
3.3 TCP/IP

TCP a IP jsou dva různé síťové protokoly. IP získává adresu, kam jsou data posílána. TCP se stará o doručení dat, pokud je cílová adresa známa. [17]

TCP (Transmission Control Protocol) je standard, podle kterého si programy a výpočetní zařízení vyměňují zprávy po síti. Je navržen tak, aby posílal pakety přes internet a zajistil úspěšné doručování dat a zpráv přes sítě. [17]

IP (Internet Protocol) je metoda pro odesílání dat z jednoho zařízení na druhé po síti. Každé zařízení má svou IP adresu, podle které je jednoznačně určené a umožňuje mu komunikovat a vyměňovat si data s jinými zařízeními připojenými k síti. [17]

V porovnání se 7 vrstvým OSI modelem má TCP/IP pouze 4 vrstvy. Model OSI (Open Systems Interconnection) obsahuje 7 vrstev, které počítačové systémy používají ke komunikaci po síti. Byl to první standardní model pro komunikaci po síti, ovšem dnešní moderní sítě už na něm nejsou stavěné a používají jednodušší TCP/IP model. [17]



Obrázek 3.1: Struktura TCP/IP a OSI model [18]

3.4 HTTP

HTTP (Hyper Text Transfer Protocol) je protokol, který k přenosu informací mezi počítačemi připojenými k webu (resp. počítače serverů i klientů) používá TCP. Jedná

se o model, podle kterého prohlížeče vyměňují data mezi klienty a webovými servery. Klient skrz prohlížeč pošle HTTP požadavek webovému serveru, který jej zpracuje a odešle požadované informace do prohlížeče. [17]

3.5 Webový server

Webový server používá HTTP a jiné protokoly, podle kterých reaguje na požadavky poslané z webových prohlížečů. Hlavním úkolem je zobrazovat obsah webových stránek pomocí ukládání, zpracování a doručování stránek koncovým uživatelům. [17]

Termín webový server se může použít jak pro hardwarovou část a softwarovou část, tak i pro obě části pracující společně.

V hardwarové části bývá webovým serverem jeden či více počítačů, které obsahují software pro webový server a soubory pro webové stránky - jako jsou například soubory HTML, obrázky, CSS styly webu, JavaScriptové soubory a jiné soubory obsahující skripty v různých jazycích.

Co se týče softwaru, tak webový server obsahuje více částí, které kontrolují přístup uživatele k hostovaným souborům. Vždy obsahuje jistě HTTP server, jenž umí pracovat s URL[2] adresami, a HTTP protokol, který je využíván prohlížeči. Pro přístup k HTTP serverům se využívají doménová jména.[3]

Webové servery zpravidla spravují statické a dynamické obsahy.

3.5.1 Statický webový server

Statický webový server jo složen z počítače a softwaru HTTP. Statický webový server odesílá uživateli soubory tak, jak jsou. Nemění obsah webových stránek, ale zobrazuje uživateli stránku bez možnosti ji nějak modifikovat. [19]

3.5.2 Dynamický webový server

Dynamický webový server je statický webový server doplněný o nějaký software, nejčastěji o aplikační server[4] a databáze. Nazývá se dynamický, protože ho před odesláním obsahu přes HTTP server může změnit a aktualizovat na základě požadavků vytvořených uživatelem ve webovém prohlížeči. [19]

[2]URL je odkaz na webový zdroj souboru, který specifikuje jeho umístění v síti a způsob jeho získání.

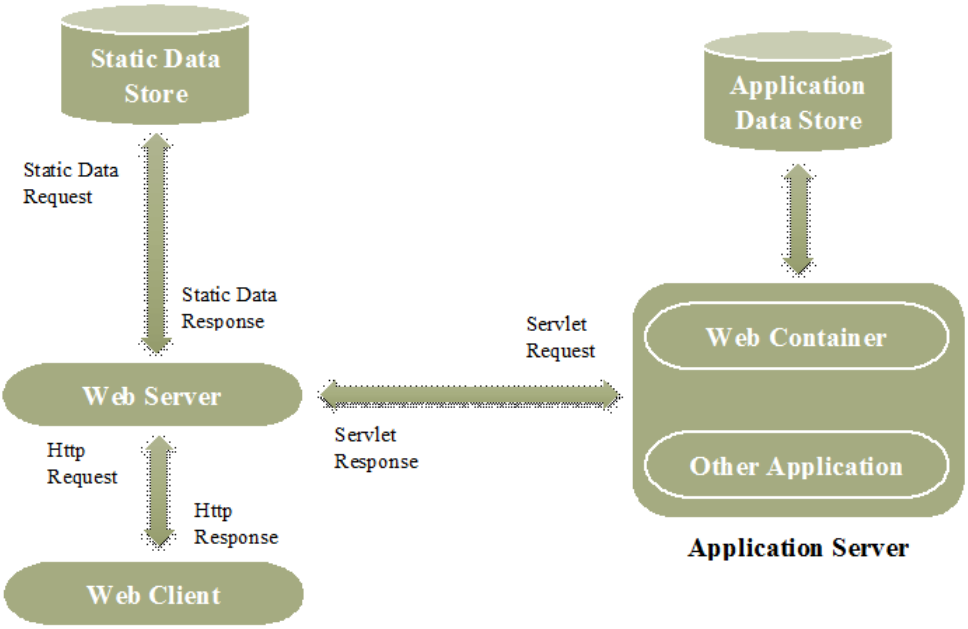
[3]Doménové jméno je kus textu, který je přiřazen k fyzické IP adrese na internetu.

[4]Aplikační servery jsou počítače, které obsahují aplikace používané uživateli webu. [20]

3.5.3 Princip webového serveru

- Pokud je uživatel připojen k internetu a chce načíst obsah stránky, zadáním URL adresy do prohlížeče vytvoří HTTP požadavek. Webový prohlížeč podle doménového jména přiřadí příslušnou IP adresu a lokalizuje webový server, kde se požadované soubory nachází.
- Webový server přijme požadavek pomocí HTTP serveru a najde příslušné soubory.
- Jakmile webový server nalezne požadové soubory, pošle je zpět do webového prohlížeče, který obsah zobrazí uživateli.

V praxi se může stát, že se nepovede nalézt nebo zpracovat vybrané soubory. V takovém případě odešle webový server zpět webovému prohlížeči chybovou hlášku, nejčastěji `404 not found`. [19] [21] [22]



Obrázek 3.2: Schéma fungování webového serveru [22]

3.6 Použité webové nástroje při vývoji webové aplikace

3.6.1 Databáze

Databáze je organizovaný soubor dat, která jsou elektronicky uložena v počítačovém systému (nejčastěji na serverech). Databáze bývá běžně spravována systémem řízení

báze dat[5]. Pokud někdo řekne databáze, obvykle myslí databázový systém.To jsou uložená data společně se systémem řízení báze dat a aplikacemi souvisejícími s nimi.

Máme mnoho různých typů databází - jako například relační databáze, objektově orientované databáze, NoSQL databáze nebo distribuované databáze. Objektově orientované databáze ukládají data stejně jako se ukládají při objektově orientovaném programování - ve formě objektů. NoSQL databáze povoluje ukládat nestrukturovaná data a manipulovat s nimi. Distribuované databáze se skládají ze dvou či více souborů uložených na různých místech. O relační databázi se budeme bavit v následující sekci. [41]

3.6.2 Relační databáze

Je to taková databáze, která data ukládá jako datové položky, které spolu souvisejí, a umožňuje přístup k nim. Definuje vztahy mezi daty.

Datové položky jsou organizovány do sady tabulek se sloupci a řádky. Tabulky se používají k uchovávání informací o objektech, o kterých si potřebujeme uchovávat data. Každý sloupec v tabulce obsahuje určitý druh dat a daná buňka obsahuje skutečnou hodnotu atributu. Řádky v tabulce představují soubor souvisejících hodnot pro jeden objekt. První sloupec v tabulce může zpravidla obsahovat jedinečný identifikátor, takzvaný primární klíč (primary key), jehož hodnota specifikuje daný řádek. Řádky napříč sadou tabulek mohou odkazovat i na řádky z jiných tabulek prostřednictvím tzv. cizích klíčů (Foreign key). [42]

Relační databáze je řízena systémem pro správu relačních databází, zkráceně RDBMS (Relational database management system). Dělá prakticky to stejné, co dělá systém řízení báze dat (DBMS), akorát je odvozen a upraven pro práci s relačními databázemi. [42]

Pro relační databázi je důležitá integrita dat. Integrita dat se dělí na dva typy - logická a fyzikální integrita. Fyzikální integritou rozumíme ochranu dat před vnějšími faktory, což mohou být výpadky proudu, útok hackerů a mnoho dalších. Nás bude spíše zajímat logická[6] integrita, která se může dělit na 3 druhy: entitní, referenční a doménovou integritu. Entitní integrita souvisí s vytvářením primárních klíčů, které identifikují jednoznačně datové položky. Stará se o to, aby nedocházelo k množení stejných záznamů. Referenční integrita se stará o to, že změny v databázi, to jest přidávání, měnění či mazání řádků v tabulce, se provádějí správně podle pravidel. Příkladem může být, že pokud je datová položka z jedné tabulky uvedena jako cizí klíč v druhé tabulce, tak nemůže být daná položka v první tabulce smazána, jinak by to rozbilo strukturu databáze. Doménová integrita se stará o to, že v dané buňce v tabulce je ta hodnota, která tam být má. To znamená, že pokud používáme primární klíče jako čísla v určitém sloupci, nesmíme do tohoto sloupce v žádném řádku vložit textový řetězec. [43]

[5]Zkráceně DBMS z anglického Database Management System. Zajišťuje práci s databází. Je to rozhraní mezi databází a aplikací nebo koncovým uživatelem, které umožňuje spravovat uložená data. [41]

[6]Někdy se zavádí i 4. druh integrity definované uživatelem. Ta je však pro každý systém jiná.

MySQL

MySQL je jedna z nejpoužívanějších a nejoblíbenějších relačních databází, která je postavena na jazyce SQL, který se používá pro práci s daty. Je to otevřený software vyvíjený, distribuovaný a podporovaný společností Oracle. [44]

Databáze MySQL používá převážně dvě úložiště[7] (Storage engines) - InnoDB a MyISAM. Od verze MySQL 5.5 je výchozím úložištěm InnoDB. InnoDB podporuje transakce, referenční integritu s cizími klíči, uzamykání[8] na úrovni řádků a body obnovení. [44] [45]

MySQL podporuje řadu programovacích jazyků, ze kterých se k němu dá připojit. Mezi významné patří PHP, C#, Java, C++, C, Python, Ruby a další. [44]

3.6.3 SQL

SQL (Structured Query Language) je strukturovaný dotazovací jazyk. Je to jazyk, který slouží k ukládání, získávání a manipulaci s daty v relační databázi.

SQL umožňuje programátorovi načíst data z databáze, uložit nové záznamy, aktualizovat a mazat záznamy. Může vytvářet nové tabulky v databázi, vytvářet uložené procedury[9], vytvářet pohledy a nebo může nastavovat oprávnění k přístupu k tabulkám, pohledům a procedurám. [47] [48]

Základní příkazy jazyka SQL: [47]

- SELECT - slouží k získání dat
- INSERT INTO - ukládá data do tabulek
- DELETE - maže záznamy z tabulek
- UPDATE - aktualizuje data v tabulce
- CREATE DATABASE - vytváří databázi
- ALTER DATABASE - upravuje databázi
- CREATE TABLE - vytváří novou tabulku v databázi
- ALTER TABLE - upravuje tabulky
- DROP TABLE - maže tabulku
- CREATE INDEX - vytvoří index. Uživatelé ho nevidí, slouží jen k rychlejšímu vyhledávání.

[7] Úložiště jsou komponenty MySQL, které pracují s operacemi SQL - zajišťují ukládání a správu dat v databázi pro různé typy tabulek. [45]

[8] Uzamykání slouží k tomu účelu, aby nedocházelo k souběžným úpravám dat. [46]

[9] Uložená procedura je kód SQL, který si můžeme uložit. Je tedy možné ho opětovně využít.

- DROP INDEX - maže index

Ukážeme si příklad, jak vytvořit tabulku pro objednávky, kde tabulka bude obsahovat ID objednávky, ID objednaného zboží a ID zákazníka, které jsou uložené v tabulkách Zbozi a Zakaznik.

```
CREATE TABLE Objednavky (
  id int NOT NULL PRIMARY KEY,
  id_zbozi int NOT NULL FOREIGN KEY REFERENCES
    ↪ Zbozi(id_zbozi),
  id_zakaznik int NOT NULL FOREIGN KEY REFERENCES
    ↪ Zakaznik(id_zakaznik)
);
```

Kód 3.1: Jednoduchá ukázka vytvoření tabulky

Kdybychom zapomněli například sloupci id_zakaznik přiřadit cizí klíč, můžeme to udělat pomocí příkazu ALTER TABLE.

```
ALTER TABLE Objednavky ADD FOREIGN KEY (id_zakaznik)
  ↪ REFERENCES Zakaznik(id_zakaznik);
```

Kód 3.2: Přiřazení cizího klíče v tabulce

3.6.4 SGML

SGML (Standard Generalized Markup Language) je mezinárodní standard, podle kterého se vytváření značkovací jazyky. To znamená, že je to metajazyk[10]. Syntaxe značkovacích jazyků je založena na zápisech zvané tagy (značky), které určují funkci nebo způsob zobrazení části textu. Z tohoto standardu se vyvinul například značkovací jazyk HTML a XML. [23]

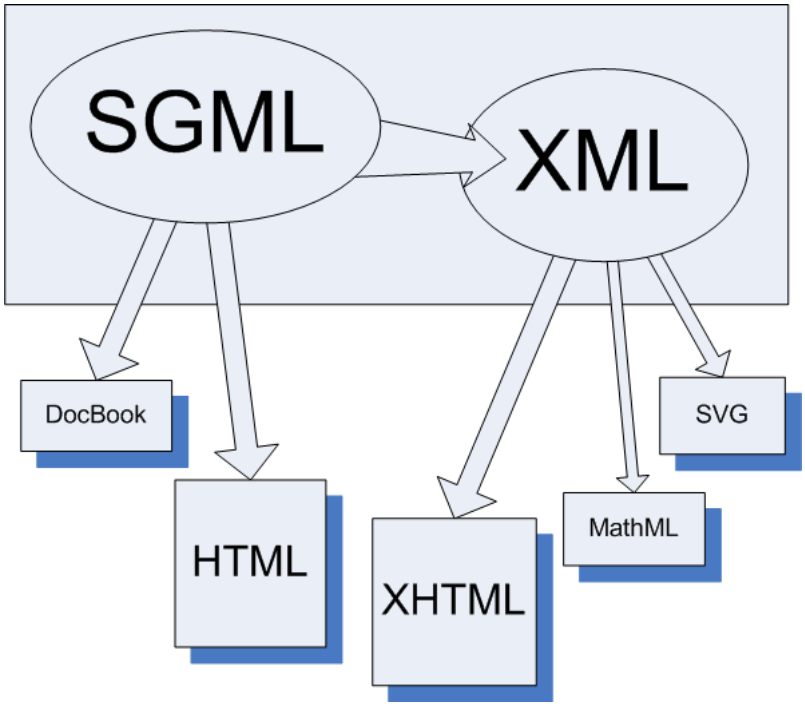
3.6.5 HTML

HTML (Hypertext Markup Language) je převládající značkovací jazyk pro web na straně klienta. Jak již bylo řečeno, vyvinul ho britský vědec Tim Berners-Lee v CERNu v průběhu 80. let minulého století.

Webový nebo aplikační server komunikuje s klientem pomocí jazyku HTML, protože tomuto jazyku prohlížeče rozumí.

Definuje, jak jsou zdroje pospojovány a organizovány. HTML je programovací jazyk, který slouží k zobrazení dat a zodpovídá za strukturu stránky. Je používán k tvorbě webových stránek, které si může zobrazit každý s přístupem k internetu. Hypertext zjednodušuje procházení webových stránek pomocí hypertextových odkazů na jiné stránky. HTML je statický a umožňuje ignorovat malé chyby v zápise. [17]

[10]Metajazyk je jazyk určený k popisu jiných jazyků.



Obrázek 3.3: Potomci SGML [24]

Značky HTML

HTML je jazyk psaný stylem značek (tagů) XML. Vždy se objevují ve dvojici začátek značky a konec značky ve tvaru <značka></značka>. Množina všech značek je pro HTML definovaná a omezená na rozdíl od XML, kde si autor musí každou značku sám definovat. [25]

Všechny soubory HTML musí začínat deklarací <!DOCTYPE>. Není to značka jazyka HTML, ale jen informace pro prohlížeč, aby věděl, jaký typ souboru má očekávat. V každém dokumentu se objevují značky:

- <html></html> - začátek html kodu
- <head></head> - hlavička dokumentu
- <body></body> - mezi tyto značky se píše obsah dokumentu, popřípadě se vkládají dovnitř značky <script></script> pro javascriptový kód

Zmíním pár dalších hodně užívaných značek:

- <h1></h1> - nadpis úrovně 1[11]
- - odstavec
- - sekce v HTML dokumentu

[11] V HTML je 6 úrovní nadpisů. Úroveň 1 má největší text.

- <table></table> - tabulka
- <tr></tr> - řádek tabulky
- <td></td> resp. <th></th> - sloupec s normálním resp. zvýrazněným textem. [25]

3.6.6 Kaskádové styly

Pomocí kaskádových stylů nebo-li CSS (anglicky Cascading Style Sheets) lze upravovat vzhled webových stránek tak, aby vypadal podle požadavků. Kaskádové styly lze do souboru HTML vložit přímo a nebo alternativně mohou být styly uvedené v samostatném souboru s koncovkou .css, na který se odkážeme pomocí značky <link> [12] uvedené v hlavičce HTML. Pro větší množství úprav pomocí kaskádových stylů je vhodnější vytvořit samostatný CSS soubor a v HTML se na něj odkazovat kvůli přehlednosti kódu. [17] [26]

Jeden dokument HTML může být stylován přes více souborů CSS a zároveň více HTML souborů se může odkazovat na jeden a ten samý CSS soubor.

Existují tři způsoby, jak příslušné elementy stylovat. První je zvaný inline. To je, když styl přiřadíme k jedné konkrétní značce přímo uvnitř. [17]

```
<h1 style="color: blue;">Nadpis</h1>
```

Kód 3.3: Inline stylování v CSS

Druhý se nazývá embedded (vnořený), který je lepší pro popis větší skupiny. Ten se používá při definování stylů přímo v hlavičce v bloku <style></style>. Následující kód by převedl všechny nadpisy úrovně 1 v dokumentu do modra. [17]

```
<style>
  h1 {color: blue;}
</style>
```

Kód 3.4: Embedded stylování v CSS

Poslední způsob linked (vázaný) se používá, pokud máme styly definované v samostatném CSS souboru. [17]

Pokud chceme upravovat pouze určitou podmnožinu elementů k dané značce, použijeme třídu (class). K úpravě jediného elementu pak slouží jedinečné ID.

```
<style>
  .nazevtridy {color: blue;}
  #nazevid{background-color: yellow;}
</style>
```

Kód 3.5: Stylování pomocí třídy a id

[12] <link ref=„stylesheet“ href=„mojestyly.css“ > [26]

Bootstrap

Bootstrap je nejpopulárnější CSS, ale i HTML a JavaScriptový framework[13]. Bootstrap obsahuje balíček nadefinovaných stylů pro všechny elementy HTML, které lze využít. Byl vyvinut ve Twitteru v roce 2011 a byl uveřejněn s otevřenou licencí.

Uživatel si ho může do projektu stáhnout přímo podle návodu na oficiálních stránkách. Pokud uživatel nechce nic stahovat, může si ho připojit přes Síť pro doručování obsahu (CDN)[14]. [27]

Bootstrap je velice snadný na naučení se a na udržení konzistence pro různá zařízení a pro různé velikosti monitorů. Používá k tomu responzivní 12 sloupcový mřížkový systém, se kterým se lehce pracuje. Můžeme provádět odsazení i vnořování sloupců s proměnlivou i pevně danou šířkou. Pomocí svých tříd nám také umožňuje, že můžeme zobrazit obsah na určité velikosti obrazovky, ale na menší či větší už se daný obsah nezobrazí a nebo se zobrazí jinak. [27]

3.6.7 JavaScript

JavaScript je skriptovací[15] programovací jazyk, který se stará o funkčnost webové aplikace. JavaScript je velice mocný objektově orientovaný programovací jazyk. Nejčastěji se s ním setkáme u webového skriptování na straně klienta. Nicméně se používá také v desktopových aplikacích například u Qt quick v knihovně QT nebo k vývoji funkčnosti na straně serveru, o tom se ale bavit nebudeme. [17]

Někdo by si mohl myslet, že JavaScript a Java spolu nějak souvisí, ale byl to pouze obchodní tah od vývojářské firmy SUN. [17]

Vlastnosti jazyka:

- Je interpretovaný. To znamená, že se nemusí kompilovat.
- Je objektový.
- V zápise závisí na velikosti písmen (říkáme, že je case sensitivní).
- Syntaxe je podobná jazykům C/C++/Java
- Funguje pouze v prohlížeči a uživatel ho může zakázat.
- Nemůže přistupovat k souborům a ani žádná data ukládat, pouze cookies[16].

[13] Framework je struktura, na které můžeme stavět jiný software. Slouží jako základ, který používáme pro vývoj softwaru. Frameworky jsou obvykle spojeny s určitým programovacím jazykem. [28]

[14] CDN (Content Delivery Network) je geograficky distribuovaná skupina serverů, které společně poskytují rychlé doručování internetového obsahu. [29]

[15] Skriptovací jazyk je takový jazyk, že jeho příkazy lze provádět bez nutnosti kompilace.

[16] Cookie je část informace uložená v počítači klienta. Používá se v několika kontextech, ale hlavním důvodem je, aby si server pamatoval údaje o klientovi. [17]

JavaScriptový kód můžeme vkládat přímo do dokumentu HTML mezi značky <script> </script>. V dokumentu se může vkládat kamkoli do HTML. Při vkládání na konec těla se může trochu zvýšit rychlost načtení stránky. Jinak můžeme mít kód v samostatném souboru a do dokumentu pouze vložíme odkaz na něj pomocí parametru src značky <script> takto: <script src=„script.js“ ></script>. [30]

JavaScript se používá po celém světě při vytváření dynamického a interaktivního webového obsahu. Umožňuje programátorovi přidávat funkce na skrývání nebo odkrývání obsahu, vytvářet galerie obrázků, přibližovat a oddalovat obrázky, přidávat animace nebo vytvářet roletové menu a spoustu dalších věcí. Funkce JavaScriptu se mohou zavolat různými způsoby. Mezi časté způsoby patří, že tlačítku přidáme atribut **onclick** nebo formuláři atribut **onsubmit** a vložíme do nich jméno volané funkce. [30]

Důležité objetky v JavaScriptu:

- window
  - Objekt window představuje okno prohlížeče. Je podporován všemi prohlížeči. Všechny globální objetky, funkce a proměnné jsou členy objektu window.
- location
  - Objekt location je používaný k získání aktuální URL adresy nebo k přesměrování. Patří do objektu window, ale v zápise stačí psát jen location místo window.location.
- document
  - Objekt document představuje danou webovou stránku. Pokud chceme přistupovat k libovolnému elementu HTML, musíme začít s přístupem k objetku document.

Pro JavaScript existuje mnoho frameworků. Já si ve své práci vybral framework jQuery pro jeho jednoduchost.

jQuery

jQuery je rychlý, malý a na funkce bohatý JavaScriptový framework. Je navržen tak, aby usnadnil skriptování v HTML na klientské straně. jQuery je rozšířené a podporované na všech nejznámějších prohlížečích. Díky snadno použitelnému rozhraní API[17] jsou věci jako procházení a manipulace s HTML dokumenty, animace, zpracování událostí a AJAX velmi jednoduché. [31]

jQuery byl vyvinut, aby programátorům šetřil práci. Spoustu běžných věcí, které by zabraly mnoho řádků v čistém JavaScriptu, lze v jQuery často napsat na výrazně menší počet řádků - ne-li pouze na jeden.

[17] API je zkratka pro Application Programming Interface, což je softwarový prostředník, který umožňuje dvěma softwarům spolu komunikovat a vyměňovat si data. [33]

Ukažme si příklad přidání řádku do dvousloupcové tabulky s ID = „pokus“ pomocí čistého JavaScriptu a pomocí jQuery.

```
var table = document.getElementById('pokus');

var row = table.insertRow(-1);
var cell1 = row.insertCell(0);
var cell2 = row.insertCell(1);
cell1.innerHTML = 'Pepa';
cell2.innerHTML = 'Novak';
```

Kód 3.6: Přidání řádku pomocí JavaScriptu

```
$('#pokus').append('<tr><td>Pepa</td><td>Novak</td></tr>');
```

Kód 3.7: Přidání řádku pomocí jQuery

3.6.8 PHP

PHP (Hypertext Preprocessor) je skriptovací jazyk s otevřeným zdrojovým kódem[18], který byl navržený speciálně pro vývoj webových aplikací. Jazyk PHP byl vytvořen v roce 1994 jediným mužem Rasmem Lerdorfem. Ke květnu roku 2022 má zastoupení 77,4% na všech webových stránkách. [32]

PHP pracuje na serverové straně a slouží k k dynamickému vytváření HTML obsahu, který je pak odeslán serverem a prohlížečem zobrazen uživateli. PHP kód lze zapisovat přímo do HTML, ale musí být oddělen značkami určenými pro PHP <?php a ?>. V souboru PHP s koncovkou .php můžeme mít jen HTML obsah (a styly CSS či JavaScript), nebo HTML obsah vypsat pomocí příkazu **echo „HTML kód“;**. [34] [35]

Výhody PHP:

- Není závislé na operačním systému. Při práci s PHP můžeme používat libovolný operační systém.
- Podporuje většinu dnešních webových serverů jako Apache[19] nebo IIS[20].
- Silnou vlastností je podpora širokého spektra všemožných databází.
- Má podporu pro různé protokoly sloužící ke komunikaci s jinými službami. Známé jsou protokoly HTTP, LDAP[21] či IMAP[22].

[18] Máme přístup k jeho zdrojovému kódu. Můžeme ho používat a distribuovat.

[19] Apache HTTP Server je open-source HTTP server pro moderní operační systémy včetně UNIX a Windows. Je to nejpoužívanější server na internetu. [36]

[20] Webový server, který vytvořila společnost Microsoft pro svůj operační systém Windows. [37]

[21] LDAP (Lightweight Directory Access Protocol) je protokol říkající, jak přistupovat k datům nebo jak data ukládat na daném serveru. [38]

[22] IMAP (Internet Message Access Protocol) je protokol sloužící k přijímání e-mailů z e-mailových serverů. [39]

- Pracuje s cookies.

Od verze PHP 5 byly přidány objektově orientované prvky jako dědičnost, soukromé a chráněné vlastnosti a metody, abstraktní třídy a metody, konstruktory a destruktory, rozhraní. Avšak nadále podporuje neobjektové programování a hodně vestavěných funkcí je neobjektového typu. [35]

Syntaxe jazyka PHP vychází zejména z jazyků C a Perl. Avšak při práci s proměnnými, které označujeme znakem $ (např. $proměnná), si programátor nemůže určit typ proměnné. Interpreter si sám určí typ dané proměnné podle přiřazené hodnoty. Problémem může být, že se typ proměnné může za běhu programu měnit. [34]

Zpracování formulářů provádí PHP pomocí dvou metod - **POST** a **GET**. Metoda POST předává serveru data na pozadí. Data nejsou nikdy zobrazena v adrese URL, nejsou uchovávána v mezipaměti a nelze je dohledat v historii prohlížeče. Metoda POST nemá limit pro posílání dat. Je proto vhodné ji používat pro předávání citlivých dat, například v přihlašovacím formuláři, kde se uvádí přihlašovací jméno a heslo. Data poslaná metodou GET se připojí do URL adresy na konec za znak otazníku (?). Data poslaná pomocí metody GET mohou být uchovávána v mezipaměti a lze je dohledat v historii. Obě metody uloží data v PHP do superglobálních proměnných $_GET a $_POST. Lze alternativně použít proměnnou $_REQUEST, která obsahuje data obou proměnných $_GET a $_POST a navíc data proměnné $_COOKIES. Její používání se obecně nedoporučuje. [34] [35]

Knihovna Dibi

Knihovna Dibi usnadňuje práci s databází. Podporuje mnoho významných databází: MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Access a obecné PDO a ODBC.

Byla vytvořena kvůli jednoduššímu zápisu SQL příkazů, snadnému přístupu k metodám nebo také kvůli přenositelnosti mezi databázovými systémy.

Ukážeme si kus kódu, jak se připojit k databázi. Kdybychom chtěli změnit databázi a použít jinou z 8 podporovaných, tak v připojení stačí změnit pouze driver a žádnou jinou funkčnost měnit nemusíme. [40]

```
$database = new Dibi\Connection([
  'driver'   => 'mysqli',
  'host'     => 'localhost',
  'username' => 'root',
  'password' => '***',
  'database' => 'database',
]);
```

Kód 3.8: Připojení k databázi pomocí knihovny Dibi. Převzato z [40]

Kapitola 4

Stávající desktopová aplikace

4.1 Qt framework

Tento framework je oblíbeným rozšířením jazyka C++. Používá se k programování aplikací s grafickým uživatelským rozhraním.

Framework Qt obsahuje celou řadu knihoven a API, které výrazně zjednodušují vývoj samotných aplikací. Implementovat komunikaci mezi objekty a zpracování událostí je velmi jednoduché pomocí pro Qt typických signálů a slotů. Výhodou je, že tento framework je multiplatformní, umožňuje vícevláknové programování a podporuje přístup do databáze. [51]

Velká část systému DAQ experimentů COMPASS a AMBER je implementována v jazyce C++ rozšířeném o framework Qt.

4.2 Desktopová aplikace

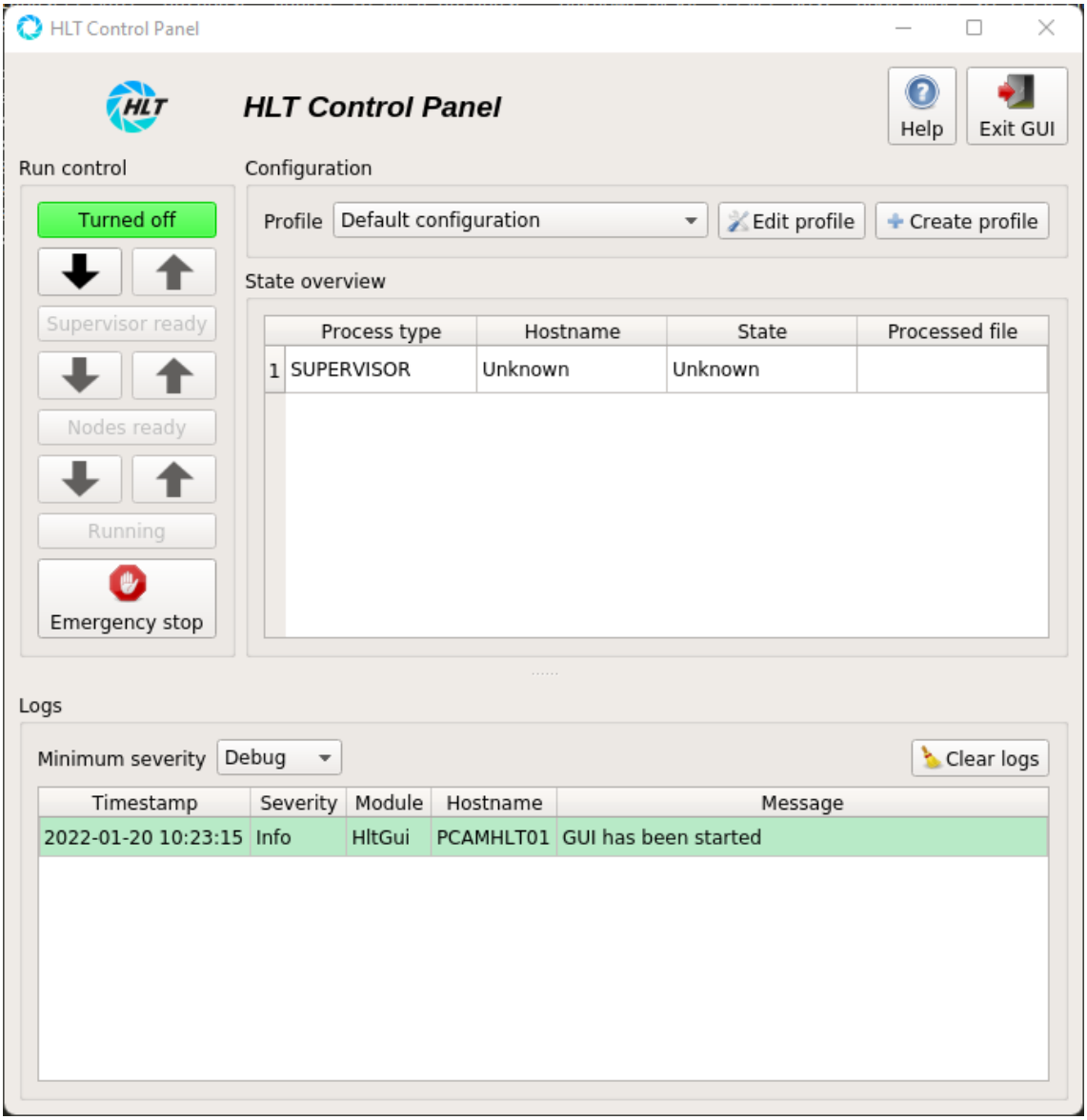
Nedílnou součástí systému DAQ vznikajícího experimentu AMBER je vysokoúrovňový trigger[1] HLT, který má výrazně omezit množství dat. Díky němu lze použít informace ze všech detektorů při rozhodovacím procesu. A právě ke spouštění a úpravám nastavení slouží stávající desktopová aplikace HLT Control Panel. [50]

Desktopová aplikace je napsána v jazyce C++ s využitím frameworku Qt. K vývoji aplikace se použila celá řada prvků Qt, zejména pak třídy QMainWindow, QWidget a QDialog. QMainWindow poskytuje prostředky pro definování rozvržení hlavního okna aplikace. Má své vlastní rozvržení, do kterého si uživatel může přidat panel nástrojů (QToolBar), menu (QMenuBar), stavovou lištu (QStatusBar). QWidget je základem grafického rozhraní, neboť umí zpracovávat události odeslané například klávesnicí nebo myší. Vykresluje se na obrazovce a má obdélníkový tvar. QDialog je třída pro dialogové okno sloužící ke krátkodobé komunikaci s uživatelem.

[1] Trigger je systém, který používá jednoduchá kritéria k rozhodnutí, jaká data má z detektorů zachovat. [1]

Aplikace slouží k vytváření profilů systému triggerů a následně k úpravě nastavení jejich parametrů.

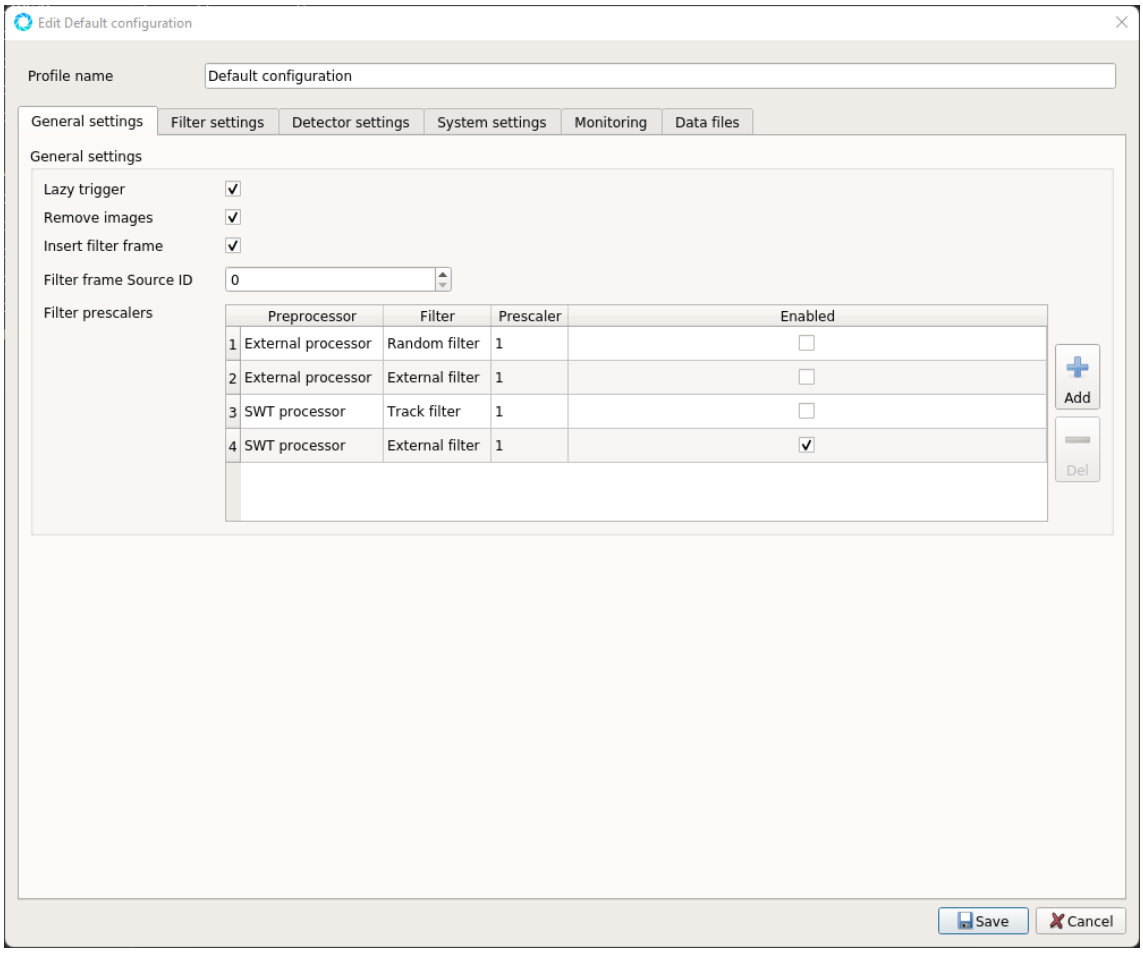
Na úvodním okně se nachází několik bloků. Vlevo je umístěn panel pro řízení běhu HLT. Vpravo od něj leží blok, který slouží pro přechod k upravení daného profilu nebo k vytvoření nového. Hned pod ním se nachází tabulka s přehledem. Zcela dole je umístěna tabulka se záznamy.



Obrázek 4.1: Úvodní okno desktopové aplikace

Pokud chce uživatel upravit nastavení pro daný profil a klikne na tlačítko s označením **Edit profile**, dostane se do okna s obecným nastavením. Ovšem nahoře bude mít panel pro výběr dalšího nastavení - nastavení filterů, nastavení detektorů, systémové nastavení, nastavení sledování a výběr složek a souborů pro ukládání dat. Všechny tyto hodnoty se uloží do tabulek v příslušné databázi MySQL, která obsahuje tabulky pro uložení všech hodnot z aplikace a mnoho dalších jako je tabulka

pro samotné profily nebo tabulka pro výběr počítačů. Kompletní schéma databáze je na ER diagramu B.1.



Obrázek 4.2: Nastavení parametrů systému triggerů

Kapitola 5

Požadavky kladené na webový nástroj pro nastavení systému triggerů

Stávající desktopová aplikace zvládne všechny úkoly splnit. Přesto bylo rozhodnuto převést její funkcionalitu pro konfiguraci nastavení HLT a dohled převést do webové aplikace. Motivací k webovému nástroji byl pohodlnější přístup k aplikaci a jednodušší vývoj aplikace.

Co se týče funkčnosti, tak webová aplikace narozdíl od desktopové aplikace nespouští vysokoúrovňový spouštěč HLT. Tato aplikace poskytuje zobrazení záznamů z databáze, vytváření nových profilů, úpravu nastavení pro existující profily a pro vybraný profil zobrazí v prohlížeči dané tabulky s parametry z databáze HLT s hodnotami přiřazenými k danému profilu. Příslušné hodnoty lze upravovat[1] a měnit v rámci omezení. Webová aplikace umožňuje nové hodnoty vytvářet[2] a ukládat do databáze.

Z hlediska technologií bylo zadavatelem požadováno, aby back-endová strana aplikace byla naprogramována v programovacím jazyce PHP, jelikož většina webových aplikací na serveru experimentu COMPASS běží právě v jazyce PHP. Jelikož je na serveru stále nainstalovaná již poměrně zastaralá verze PHP 5.4.16, tak bylo vhodné volit metody a funkce, které jsou kompatibilní s touto verzí a jinými aplikacemi, ale aby byly zároveň kompatibilní s novejšími verzemi PHP. To v budoucnosti usnadní případný přechod na novější verzi jazyka PHP, který doporučuji.

Co se týče práce s databází, tak je používána relační databáze MySQL. Databáze se všemi tabulkami již existuje a webová aplikace je postavená na této databázi. Pro práci s databází jsem měl za podmínky splnění kompatibility volný výběr frameworku. Rozhodl jsem se pro práci s frameworkem Dibi. Je to malý framework s přehlednou dokumentací, který umí pracovat se všemi typy příkazů, které jsou v aplikaci potřeba.

[1] Za úpravy se považuje měnění hodnot v řádku tabulky ve sloupci pro daný parametr

[2] Vytváření nových hodnot znamená přidání řádku s novými hodnotami do příslušné tabulky

Pro front-endovou část programování, která běží v prohlížeči u uživatele, nebyl požadován konkrétní jazyk, ale spíše samotná funkčnost. Důležitým prvkem bylo, aby při ukládání změn aplikace požadovala potvrzení od uživatele a tím zamezila uložení chybných hodnot. Dále, aby v případě vzniknu chyby byl uživatel přesměrován na chybovou stránku. To lze poměrně jednoduše naprogramovat pomocí jazyka JavaScript, který jsem použil. Pro práci s JavaScriptem jsem se rozhodl ještě použít framework jQuery, který mimo jiné usnadňuje práci s technologií AJAX. Všechna data, která se posílají do PHP skriptů pro práci s databází, jsou posílána přes AJAX metodou POST a právě proto využívám jQuery a jeho metodu ajax(). V případě úspěchu je uživatel ponechán na právě navštívené stránce a je mu sděleno, že vše proběhlo úspěšně. Pokud nastane jakákoli chyba, tak je přesměrován na chybovou stránku.

Dalším požadavkem na front-end byla responzivita uživatelského rozhraní[3] pro různé velikosti obrazovek. K tomu musely být využity prostředky, které jsou podporovány v prohlížeči Mozilla Firefox ve verzi 91.3.0 a novějších, neboť tato verze je v prostředí experimentu COMPASS využívána. Podle tohoto kritéria jsem si mohl kaskádové styly nadefinovat sám a nebo využít libovolný CSS framework. Definovat si vlastní styly by bylo časově náročné a zbytečné, když existuje spousta kvalitních CSS frameworků jako například Bootstrap nebo Tailwind. Po zvážení jsem pro návrh vzhledu aplikace vybral framework Bootstrap. Bootstrap má předdefinované vzory a využívá 12 sloupcovou mřížku. Narozdíl od Tailwindu Bootstrap vždy neumožňuje uživateli měnit vzory a navrhovat stránku od základů tak, jak by si představoval. V tomto ohledu je Tailwind napřed. Nemá předdefinované vzory pro jednotlivé značky (tagy) jazyka HTML, ale využívá předdefinovaných tříd, které se nevztahují k jedné značce, ale mohou se použít u celé řady různých značek. To ho dělá velice oblíbeným pro kreativní jedince, kteří si vzhled stránky chtějí od základů vytvořit podle sebe. Pro mnou navrhovanou aplikaci stačí použít předdefinované vzory v Bootstrapu a využít jeho schopnosti navrhovat responzivní vzhled. Dalším důvodem, proč jsem si zvolil Bootstrap, je jeho precizně zpracovaná online dokumentace.

[3]Responzivita vzhledu webové aplikace znamená, že aplikace provádí změny vzhledu webu dynamicky v závislosti na velikosti obrazovky, na které je spuštěna.

Kapitola 6

Návrh a implementace webového nástroje

6.1 Třída HltDatabase

Nejprve bych zmínil, že třída HltDatabase má dvě verze. Z důvodu kompatibility se staršími verzemi byla navržena jedna třída využívající třídu mysqli, která je součástí jazyka PHP od verze 5.0.0. Druhý návhr pro verze jazyka PHP 7.1 a novější využívá knihovnu Dibi. Oba dva návrhy obsahují stejně pojmenované atributy i metody soužící ke stejnému účelu.

Před samotným návrhem webové aplikace bylo potřeba navrhnout třídu pro práci s databází, neboť samotný obsah stránek je tvořen daty z databáze a navíc uživatel upravuje data a potřebuje je ukládat do databáze. Po doporučení jsem si zvolil pro práci s databází knihovnu Dibi.

Třída obsahuje jediný atribut **$database**, kterému je v konstruktoru přiřazeno připojení do databáze přes knihovní třídu **Connection** (viz kód 3.8) pro jeden návrh a pro druhý návrh je připojení vytvořeno kódem **$database = new mysqli(HOST, USERNAME, PASSWORD, DATABASE, PORT);**. Všechny parametry jsou definované v souboru **database_account.php**. Pokud selže připojení do databáze, je vyvolána výjimka, která je odchycena ještě v konstruktoru a v rámci jejího ošetření je do parametru **$database** uložena hodnota **null**. Dále třída obsahuje téměř 50 metod (viz. příloha C), které jsou navržené vyloženě pro práci s databází HLT viz ER diagram B.1. Obsahuje metody k získávání dat z databáze (SQL příkaz SELECT), k aktualizaci dat v tabulkách databáze (SQL příkaz UPDATE), k vkládání dat (SQL příkaz INSERT) a k mazání dat z tabulek (SQL příkaz DELETE).

Následující příklad ukazuje mazání řádku hodnot z tabulky **monitoring_prescaler** podle primárního klíče $prescaler_id. Metoda **query()** je součástí knihovní třídy **Connection**.

```
public function
    ↪ delete_from_monitoring_prescaler($prescaler_id)
    {
        if($this->database != null)
            $this->database->query('DELETE FROM
                ↪ monitoring_prescaler WHERE prescaler_id =
                ↪ ?', $prescaler_id);
    }
```

Kód 6.1: Metoda delete_from_monitoring_prescaler($prescaler_id)

6.2 HTML a PHP šablony

Při vývoji webové aplikace jsem nejprve začal s návrhem vzhledu stránky. Při návrhu jsem se inspiroval již zmíněnou desktopovou aplikací (viz obrázek 4.1), aby pro uživatele, kteří ji používají, byl přechod k webové aplikaci co nejjednodušší.

Navrhoval jsem tedy úvodní stránku (skript **index.php**) tak, že se podobá úvodnímu oknu desktopové aplikace. Zároveň se liší strukturou od ostatních a nemůže využívat společné funkce pro výpis HTML. Dále jsem musel navrhnout 6 dalších souborů pro úpravu nastavení, které využívají společnou hlavičku, postranní menu a zápatí. Z toho důvodu jsem si vytvořil skript **html_functions.php**, ve kterém jednotlivé funkce generují HTML kód definující obsah pro těchto 6 stránek. Stránky jsou si podobné a liší se obvykle v počtu zobrazených tabulek a jejich rozměrech. Na závěr byl navržen soubor pro vytváření profilu s výběrem, zda chce uživatel vytvořit pouze profil bez nastavení, nebo chce zkopírovat vzorové nastavení od jiného již existujícího profilu.

Všechny stránky byly navrhovány pomocí CSS frameworku Bootstrap, který se stará o responsibilitu stránky. Využívá také stylování pro 3 různé šířky s označením lg (large), md (medium) a sm (small). Pokud si například vezmeme třídu pro margin m, která obstarává odsazení od ostatních prvků na stejné úrovni, tak třída m-sm-2 provede odsazení o 2 jednotky, dokud není okno stránky velikosti small a menší. Poté tato třída přestává fungovat.

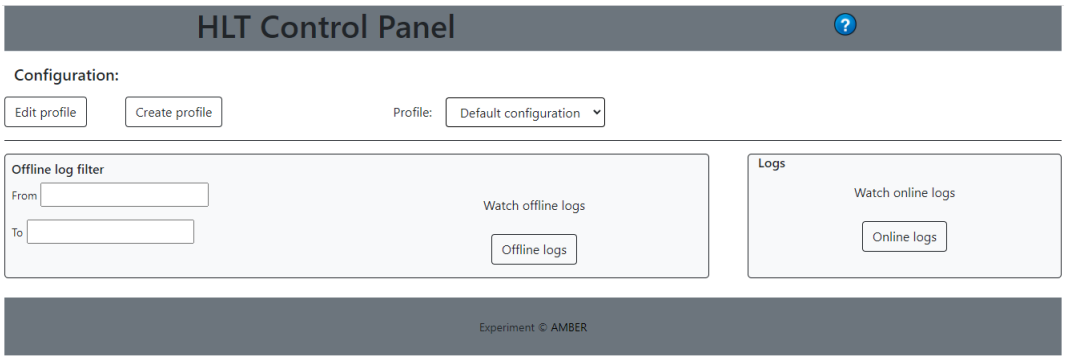
6.2.1 Úvodní okno

Pokud byla spuštěna relace (session), smaže před zobrazením úvodního okna skript **index.php** superglobální proměnnou **$_SESSION['profile']**. Tato proměnná slouží pro skripty pro úpravu nastavení, aby věděly, jaký konfigurační profil spravovat. Poté zničí relaci a zobrazí obsah.

Úvodní stránka obsahuje 2 bloky. V prvním bloku si uživatel zvolí profil, který bude upravovat a případně zvolí možnost pro vytvoření nového profilu. Výběr profilu je

uskutečněn prostřednictvím značky <select> a k vyplnění možností[1] používám PHP a metodu **select_from_configuration_profile()** třídy **HltDatabase**, která načte z databáze všechny existující profily nastavení. Pokud chceme přejít k nastavení profilu po kliknutí na tlačítko s označením **Edit profile**, tak přes jQuery funkcí **ajax()** je profil metodou **POST** odeslán skriptu **general_settings.php** a uživatel je na tento skript i přesměrován. Funkce vytvoření profilu přesměruje uživatele na skript **create_profile.php**.

Ve druhém bloku se nachází dva formuláře pro zobrazení záznamů. V prvním uživatel zadá časový úsek, ve kterém si chce prohlédnout záznamy a ty se mu zobrazí na samostatné stránce. Druhý uživatele přesune na stránku s online záznamy, které se pomocí AJAXu načítají průběžně a zobrazují nejnovější záznamy z tabulky **log** v databázi.



Obrázek 6.1: Úvodní okno webové aplikace

6.2.2 Vytváření konfiguračního profilu

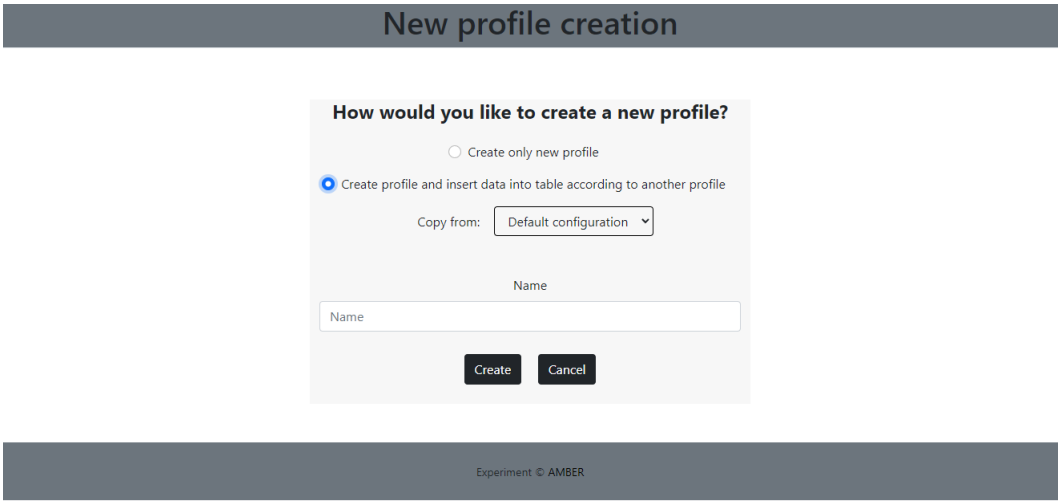
Tuto stránku generuje skript **create_profile.php**, na kterou se dostaneme, pokud na úvodní stránce zvolíme možnost vytvořit profil (**Create profile**).

Aplikace nabízí 2 možnosti, jak vytvářet konfigurační profil. První jednodušší možností je vytvoření samostatného profilu nastavení. Jediné, co se musí v databázi překopírovat, jsou hodnoty v tabulce **configuration_key**. To je z toho důvodu, že do této tabulky se nové hodnoty nevkládají, pouze se upravují a aktualizují. Jako vzorový profil se zde automaticky bere profil s nejvyšším ID. Druhou možností je vytvoření nového profilu jako kopie již existujícího profilu. V tomto případě se všechna data z tabulek odpovídající vzorovému profilu zkopírují a přiřadí se k nově vytvářenému profilu - všechny zkopírované hodnoty budou mít jako hodnoty cizího klíče identifikátor nově vytvořeného profilu.

6.2.3 Správa nastavení profilů

Jedná se o 6 skriptů se stejnou strukturou, které odpovídají záložce nastavení z desktopové aplikace.

[1] Značky <option></option> uvnitř značek <select></select>



Obrázek 6.2: Tvorba konfiguračního profilu

Skripty:

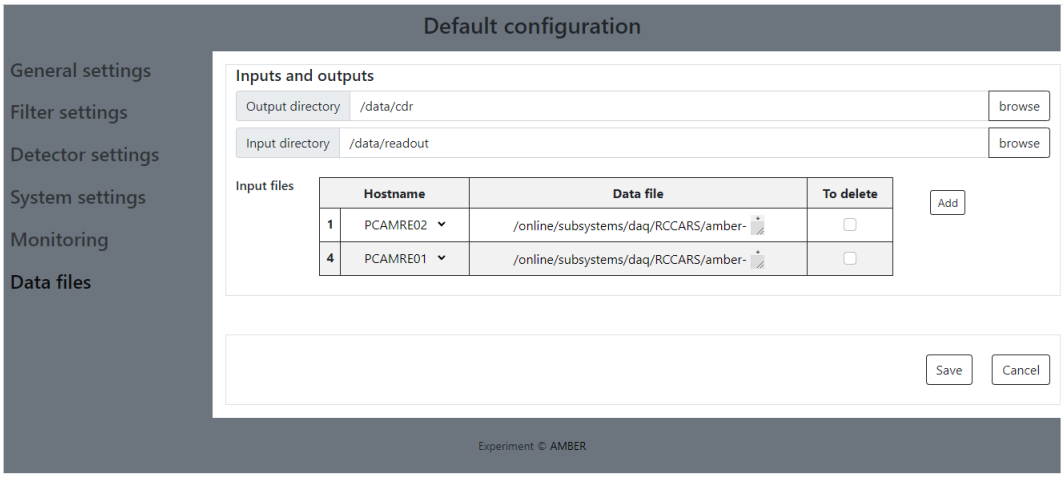
- general_settings.php
- filter_settings.php
- detector_settings.php
- system_settings.php
- monitoring.php
- data_files.php

Uživatel bude z úvodní stránky přesměrován na skript **general_settings.php**, který před zobrazením obsahu zahájí relaci pomocí příkazu session_start() a uloží do superglobální proměnné $_SESSION id upravovaného profilu, kterou používají všechny tyto skripty. Tato proměnná je uchovávána do doby, než uživatel ukončí úpravu nastavení a vrátí se na úvodní stránku, kde je proměnná smazána a relace zničena.

Těchto 6 stránek má stejnou strukturu. Využívají společné funkce pro generování záhlaví, nabídky a zápatí, které se nacházejí ve skriptu **html_functions.php**. Funkce **header_($title)** generuje záhlaví a parametr **$title** obsahuje řetězec s názvem daného profilu. Funkce **sideNavBar($actual)** zobrazuje postranní menu na levé straně s výběrem jako je horní menu u desktopové aplikace. Parametr **$actual** slouží k orientaci uživatele, na které stránce se nachází. Pokud se příliš zmenší okno stránky, postranní menu díky responzibilitě Bootstrapu zmizí a uvolní více prostoru pro samotný obsah a objeví se tlačítko, které dané menu v případě potřeby vysune. A poslední společně využívanou funkcí je **footer($src)**, která zobrazí zápatí stránky. Její parametr **$src** slouží k vložení JavaScriptových skriptů.

Pro každou stránku pak obsah generuje jedna funkce. Pro každý skript se jeho funkce jmenuje stejně.

- general_settings()
- filter_settings()
- detector_settings()
- system_settings()
- monitoring()
- data_files()



Obrázek 6.3: Ukázka rozvržení stránky na stránce Data files

Každá z těchto funkcí využívá příslušné metody třídy **HltDatabase** pro zobrazení dat. Pro získání dat z databáze se využívá příkaz SELECT jazyka SQL. Proto obvykle název metody pro získání dat začíná slovem „select“ a končí buď názvem tabulky nebo názvem skriptu, ve kterém je metoda využita. Například metoda **select_from_filter_prescaler()** získá data z tabulky **filter_prescaler**, která se zobrazuje na stránce generované skriptem **general_settings.php**.

Ke každé této stránce je přiřazen jeden JavaScriptový skript a jeden PHP skript pro back-end, který spravuje odeslaná data a v závislosti na nich v databázi některé hodnoty aktualizuje, maže a nebo přidává nové. Výjimku tvoří stránka generovaná skriptem **filter_settings.php**, která obsahuje ještě 2 další menu uvnitř. Z toho důvodu jsem ji pro lepší čitelnost kódu přiřadil 4 PHP skripty pro back-end.

JavaScriptové skripty:

- add_general_settings.js
- add_filter_settings.js
- add_detector_settings.js

- add_system_settings.js
- add_monitoring.js
- add_data_files.js

PHP skripty:

- general_settings_script.php
- external_filter_script.php - pro stránku generovanou pomocí filter_settings.php
- random_filter_script.php - pro stránku generovanou pomocí filter_settings.php
- pattern_filter_script.php - pro stránku generovanou pomocí filter_settings.php
- track_filter_script.php - pro stránku generovanou pomocí filter_settings.php
- detector_settings_script.php
- system_settings_script.php
- monitoring_script.php
- data_files_script.php

JavaScriptové skripty obsahují funkce pro přidávání řádků s novými hodnotami do tabulek. Dále obsahují funkci, která je zavolána při stisknutí tlačítka pro uložení **Save**. Uživatele se zeptá, zda chce skutečně data uložit - a pokud ano, tak načte všechna data ze stránky. V tabulkách na stránce jsou skryté i původní hodnoty před provedením změn, které si funkce také načte a porovná se současnými, zda došlo ke změnám, či nikoliv, a poznamená si to do kontrolního pole změn. Tabulky obsahují také zaškrtávací políčko, zda má být záznam z databáze vymazán. Tato funkce zkontroluje, zda jsou políčka pro smazání zaškrtlá, a poznamená si do kontrolního pole mazání, které řádky budou smazány. Po kontrole dat zavolá funkci **ajax()**, která metodou **POST** odešle všechna data do příslušného back-end PHP skriptu včetně kontrolních polí o změnách a mazání. Tato funkce také od PHP skriptu obdrží odpověď v podobě stavového kódu (status code), zda vše proběhlo v pořádku či došlo k nějaké chybě, kterou tato funkce zpracuje a uživatele přesměruje na chybovou stránku.

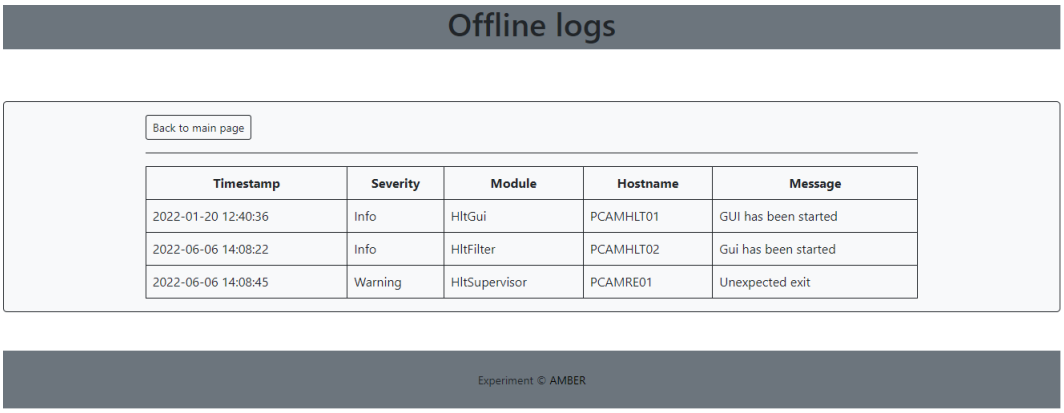
```
$.ajax({
  type: "POST",
  url: "scripts/general_settings_script.php",
  data: {preprocessor:preprocessor, filter:filter,
    ↪ prescaler:prescaler, enabled:enabled, changed:changed,
    ↪ prescaler_id:prescaler_id, to_delete:to_delete,
    ↪ lazyTrigger:lazyTrigger, removeImages:removeImages,
    ↪ filterFrame:filterFrame, frameSource:frameSource},
  statusCode: {
    200: function(){
```

```
      alert('Everything was saved');
      location.href = 'general_settings.php';
    },
    201: function(){
      alert('There was no data sent for saving');
      location.href = 'general_settings.php';
    },
    500: function(){
      alert('Connection to database failed');
      location.href = 'error.php';
    },
    501: function(){
      alert('Some sql code failed');
      location.href = 'error.php';
    }
  }
});
```

Kód 6.2: Odeslání dat funkcí AJAX() ze stránky general_settings.php do příslušného skriptu

6.2.4 Záznamy

Záznamy jsou generovány pomocí skriptů **offline_logs.php** a **online_logs.php**. První zmíněný slouží k zobrazení dat z tabulky **log** na základě uživatelem vybraného časového úseku. Druhý skript zobrazuje dynamicky nově přidávaná data do databáze.



Obrázek 6.4: Stránka s tabulkou záznamů

Stránka s dynamicky načítanými záznamy vypadá stejně. Využívá ale JavaScriptovou funkci **setInterval()**, která každou sekundu volá funkci, jež pomocí **AJAXu** volá skript **online_logs_script.php**, který kontroluje, zda nebyla od poslední změny přidána data do tabulky záznamů (**log**) v databázi, a pokud byla, odešle je zpět a pomocí **AJAXu** zobrazí v tabulce na stránce.

6.3 Back-endové skripty

6.3.1 Skript pro vytváření profilu

Skriptu **create_profile_script.php** jsou předána data funkcí **ajax()** metodou **POST** a nikoliv formulářem. Je to z toho důvodu, že skript po ukončení vrací stavový kód[2] (status code) v závislosti na tom, zda vše proběhlo v pořádku či nikoli, který lze ve funkci **ajax()** dobře zpracovat pomocí objektu **statusCode**, který spustí funkci odpovídající vrácenému stavovému kódu. Ukázka zpracování stavových kódů je ve výpisu kódu 6.2.

Skript je rozdělen na 2 části. V první části se stará o vytváření samotného profilu, pokud byla vybrána první možnost. Skript vyhledá ID posledního profilu v tabulce **configuration_profile** jako vzoru a vloží do ní nový profil s vybraným jménem a do tabulky **configuration_key** nakopíruje hodnoty podle vzoru a přiřadí jim ID nově vytvořeného profilu.

Druhá část je rozsáhlejší. Uživatel zvolil vzorový profil, ze kterého se mají data překopírovat. ID vzorového profilu bylo do skriptu odesláno, tak si ho skript uloží a vytvoří nový profil s vybraným jménem. Poté musí projít všechny patřičné tabulky, získat z nich data odpovídající danému profilu a překopírovat je do tabulky s ID nově vytvořeného profilu.

Pokud vše proběhne v pořádku, skript vrátí stavový kód 200. Nepovede-li se připojit do databáze, vrátí 500 a dojde-li k chybě při spouštění SQL příkazu, vrátí kód 501.

6.3.2 Skripty pro správu nastavení profilů

Každý skript zkontroluje, zda mu data byla odeslána. Poté si data uloží do proměnných a připojí se do databáze. Pokud připojení selže, vyvolá výjimku a v rámci její obsluhy vrátí stavový kód 500. Dále pomocí **for** cyklu zkontroluje kontrolní pole pro změnu a mazání, zda se původní data změnila či je požadováno jejich smazání a zavolá příslušnou metodu třídy **HltDatabase** pro UPDATE nebo DELETE. Nově přidané hodnoty vloží do jednoho pole a to vloží do příslušné tabulky v databázi. Jestliže se vše uloží v pořádku, vrátí skript stavový kód 200. V případě vyvolané výjimky při provádění SQL příkazu je odeslán stavový kód 501.

Skripty ještě musí aktualizovat hodnoty v tabulce **configuration_key**. Do této tabulky se nepřidávají nové parametry a ani se existující nemažou. Tyto hodnoty lze pouze měnit.

Ukázka skriptu **general_settings_script.php** je ve výpisu v příloze A.1.

[2] Je to odpověď serveru na požadavek prohlížeče.

6.3.3 Skript pro dynamické načítání záznamů

Skript **online_logs_script.php** je volán **AJAXem** a pouze načítá data z tabulky **log** v databázi, která byla přidána nově a na stránce ještě nebyla zobrazena. Tento skript opět v případě chyby připojení do databáze vrací stavový kód 500. Pokud vše proběhlo v pořádku, vrací kód 200.

6.4 Testování

Webová aplikace byla nejprve otestována na lokální síti pro různé verze jazyka PHP a na nejpoužívanějších prohlížečích. Následně byla vytvořena zkušební databáze na serveru COMPASS a na ni ve spolupráci s konzultantem Ing. Martinem Zemkem byla webová aplikace otestována. Během testování nebyly nalezeny žádné chyby a výsledky odpovídají akcím prováděným na desktopové aplikaci.

Jako příklad zde uvedu vytváření nového profilu s okopírováním hodnot z jiného již existujícího profilu.

1. V úvodním okně bylo stisknuto tlačítko **Create profile**.
2. V okně pro tvorbu profilu byla zvolena možnost vytvořit profil a okopírovat data z jiného profilu.
3. Byl zvolen vzorový profil z již existujících profilů.
4. Nově vytvářenému profilu se přiřadilo jméno.
5. Dále bylo stisknuto tlačítko **Create** k vytvoření profilu a potvrzeno, že je akce skutečně požadována.
6. Na závěr proběhla kontrola, zda se vytvořil nový profil a zda se ve správných tabulkách okopírovaly hodnoty podle zvoleného vzorového profilu a byly přiřazeny k nově vytvořenému profilu.

Obdobně byly okontrolovány i ostatní funkce webového nástroje.

Závěr

V průběhu práce jsem se seznámil s experimenty COMPASS a AMBER a s laboratoří CERN. Následně byla analyzována stávající desktopová aplikace určená pro nastavení a řízení vysokoúrovňového triggeru experimentu AMBER. Po seznámení se s funkcemi desktopové aplikace jsem navrhl a implementoval požadovaný webový nástroj. Současně jsem se přitom seznámil s vybranými webovými technologiemi.

Tím jsem splnil všechny předem stanovené cíle práce. Zároveň byla aplikace spuštěna a uživatelsky otestována na serveru experimentu COMPASS. Je kompatibilní jednak s verzí jazyka PHP nainstalovaného na serveru a s prohlížečem používaným na lokálních počítačích, tak i s nejnovějšími verzemi jazyka PHP a moderními prohlížeči. Aplikace již na serveru běží a je připravena k použití.

Tato aplikace nabízí pohodlný přístup k nastavení vysokoúrovňového triggeru bez použití desktopové aplikace. Dále nabízí přístup i mimo vnitřní síť bez nutnosti vzdáleného připojení se do sítě.

V budoucnu by se na této aplikaci mohly různé funkce vylepšit, či některé nové doplnit. Za zmínku stojí obnova nastavení, které uživatel neuložil. Všechna data, která uživatel zadá, by se ukládala na určitý časový úsek do souborů **cookies**, odkud by se - například v případě odpojení od internetu - znovu načetla. To by uživateli usnadnilo práci a nemusel by všechny hodnoty vyplňovat znovu. Dále by se pak případně podle nových požadavků mohl rozšířit filtr offline záznamů na úvodním okně.

Literatura

[1] *Home | CERN*. URL: <https://home.cern/> [cit. 2022-04-27]

[2] *The CERN accelerator complex, layout in 2022.* URL: <https://cds.cern.ch/images/CERN-GRAPHICS-2022-001-1> [cit. 2022-04-27]

[3] CURLEY, Robert. *muon | subatomic particle | Britannica.* URL: <https://www.britannica.com/science/muon> [cit. 2022-05-10]

[4] *COMPASS | CERN*. URL: <https://home.cern/science/experiments/compass> [cit. 2022-04-28]

[5] ABBON, P. et al. The COMPASS experiment at CERN. In: *Nuclear Instruments and Methods in Physics Research A577*. 2007, s. 455-518.

[6] *New COMPASS page*. URL: <https://wwwcompass.cern.ch/> [cit. 2022-04-30]

[7] LOPES, A. *Meet AMBER | CERN*. URL: <https://home.cern/news/news/physics/meet-amber> [cit. 2022-04-05]

[8] *antiproton | physics | Britannica*. URL: <https://www.britannica.com/science/antiproton> [cit. 2022-05-06]

[9] Merriam Webster. *Pion Definition & Meaning - Merriam Webster.* URL: <https://www.merriam-webster.com/dictionary/pion> [cit. 2022-05-06]

[10] FROLOV, V., HUBER, S. et al. Data Acquisition System for the COMPASS++/ AMBER Experiment. *IEEE Transactions on Nuclear Science*. 2021, (68). DOI: 10.1109/TNS.2021.3093701.

[11] BODLÁK, M., FROLOV, V., JARÝ, V. et al. FPGA based data acquisition system for COMPASS experiment. *Journal of Physics: Conference Series*. 2013, (513). DOI: 10.1088/1742-6596/513/1/012029.

[12] *Hodoscope - Wikipedia*. URL: <https://en.wikipedia.org/wiki/Hodoscope> [cit. 2022-05-08]

[13] *FPGA based data acquisition system for COMPASS experiment - CERN document server*. URL: <http://cds.cern.ch/record/1606064?ln=sk> [cit. 2022-05-10]

[14] ANDREWS, E. *Who Invented the Internet? - HISTORY.* URL: <https://www.history.com/news/who-invented-the-internet> [cit. 2022-05-12]

[15] *TCP/IP.* URL: <https://www.gvp.cz/ucebnice/Vyptech/internet/tcpip.htm> [cit. 2022-05-12]

[16] *What is packet? | Network packet definition | Cloudflare.* URL: <https://www.cloudflare.com/learning/network-layer/what-is-a-packet/> [cit. 2022-05-12]

[17] SRINIVASAN, M. *Web Technology: Theory and Practice.* Delhi: Pearson Education India, 2012. ISBN 978-933-2508-194.

[18] *OSI cs TCP/IP - javtpoint.* URL: <https://www.javatpoint.com/osi-vs-tcp-ip> [cit. 2022-05-15]

[19] ESPOSITO, Dino. *Modern Web Development: Understanding domains, technologies, and user experience (Developer Reference).* Microsoft Press, 2016. ISBN 978-150-9300-013.

[20] *Definition of Application Server - Gartner Information Technology Glossary.* URL: <https://www.geeksforgeeks.org/web-servers-work/> [cit. 2022-05-15]

[21] *How do Web Servers work? - GeeksforGeeks.* URL: <https://www.geeksforgeeks.org/web-servers-work/> [cit. 2022-05-15]

[22] SINGH, A. *Web Server | Features of Web Server - Tech Blog.* URL: <https://msatechnosoft.in/blog/web-server-features-of-web-server/> [cit. 2022-05-15]

[23] HEMMENDINGER, D. *computer programming language - SGML | Britannica.* URL: <https://www.britannica.com/technology/computer-programming-language/SGML> [cit. 2022-05-15]

[24] *SGML, XML, HTML, and XHTML.* URL: <https://cscie12.dce.harvard.edu/lecture_notes/2007-08/20080130/slide26.html> [cit. 2022-05-15]

[25] *HTML & CSS - W3C.* URL: <https://www.w3.org/standards/webdesign/htmlcss> [cit. 2022-05-16]

[26] *CSS Tutorial.* URL: <https://www.w3schools.com/css/> [cit. 2022-05-17]

[27] *Bootstrap - The most popular HTML, CSS and JS library in the world.* URL: <https://getbootstrap.com/> [cit. 2022-05-17]

[28] *What is a Framework?.* URL: <https://www.codecademy.com/resources/blog/what-is-a-framework/> [cit. 2022-05-17]

[29] *What is CDN | How do CDNs work? | CLoudflare.* URL: <https://www.cloudflare.com/learning/cdn/what-is-a-cdn/> [cit. 2022-05-17]

[30] *JavaScript Tutorial.* URL: <https://www.w3schools.com/js/> [cit. 2022-05-18]

[31] *jQuery.* URL: <https://jquery.com/> [cit. 2022-05-18]

[32] *PHP vs. ASP.NET usage statistics, May 2022.* URL: <https://w3techs.com/technologies/comparison/pl-aspnet,pl-php> [cit. 2022-05-18]

[33] *What is an API? (Application Programming Interface) | MuleSoft.* URL: <https://www.mulesoft.com/resources/api/what-is-an-api> [cit. 2022-05-18]

[34] *PHP: Hypertext Preprocessor.* URL: <https://www.php.net/> [cit. 2022-05-18]

[35] WELLING, Luke, THOMSON, Laura. *Mistrovství PHP a MySQL.* Computer Press, 2017. ISBN 978-80-251-4892-1.

[36] *Welcome! - The Apache HTTP Server Project.* URL: <https://httpd.apache.org/> [cit. 2022-05-18]

[37] *Home: The Official Microsoft IIS Site.* URL: <https://www.iis.net/> [cit. 2022-05-18]

[38] *LDAP.com - Lightweight Directory Access Protocol.* URL: <https://ldap.com/> [cit. 2022-05-18]

[39] *Internet Message Access Protocol (IMAP) - GeeksforGeeks.* URL: <https://www.geeksforgeeks.org/internet-message-access-protocol-imap/> [cit. 2022-05-18]

[40] GRUDL, D. *Dokumentace | dibi.* URL: <https://dibiphp.com/cs/documentation> [cit. 2022-05-18]

[41] *What Is a Database | Oracle Česká Republika.* URL: <https://www.oracle.com/cz/database/what-is-database/> [cit. 2022-05-18]

[42] *What Is a Relational Database | Oracle Česká Republika.* URL: <https://www.oracle.com/cz/database/what-is-a-relational-database/> [cit. 2022-05-18]

[43] NAEEM Tehreem. *Data Integrity in a Database - Why Is It Important | Astera.* URL: <https://www.astera.com/type/blog/data-integrity-in-a-database/> [cit. 2022-05-18]

[44] *MySQL :: MySQL Documentation.* URL: <https://dev.mysql.com/doc/> [cit. 2022-05-19]

[45] *MySQL storage engines - InnoDB, MyISAM, Memory.* URL: <https://zetcode.com/mysql/storageengines/> [cit. 2022-05-19]

[46] *What Is A Database Lock? - rkimball.com.* URL: <https://www.rkimball.com/what-is-a-database-lock/> [cit. 2022-05-19]

[47] *SQL Introduction.* URL: <https://www.w3schools.com/sql/sql_intro.asp> [cit. 2022-05-19]

[48] *SQL Tutorial.* URL: <https://www.tutorialspoint.com/sql/index.htm> [cit. 2022-05-19]

[49] *Qt Framework - One framework to rule all!.* URL: <https://www.qt.io/product/framework> [cit. 2022-05-28]

[50] ZEMKO, M., FROLOV, V. et al. Free-running data acquisition system for the AMBER experiment *EPJ Web of Conferences.* 2021, (251). DOI: 10.1051/epjconf/202125104028

[51] *Qt Documentation | Home.* URL: <https://doc.qt.io> [cit. 2022-05-28]

Příloha A

Ukázka skriptu general_settings_script.php

```
require_once('../database.php');
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

date_default_timezone_set('Europe/Zurich');

http_response_code(201);
if (isset($_POST['prescaler']) && isset($_SESSION['profile']))
{
    try{
        $database = new HltDatabase('mysqli', 'localhost',
           ↪ 'root', 'Nerous253', 'hlt');
        if($database->getDatabase()==null)
        {
            throw new Exception('Connection to database
               ↪ failed');
        }

        $changed = $_POST['changed'];
        $to_delete = $_POST['to_delete'];

        $puvodni_pocet = count($_POST['prescaler_id']);
        $celkovy_pocet = count($_POST['prescaler']);

        $preprocessor = $_POST['preprocessor'];
        $filter = $_POST['filter'];

        $processor_id = array_fill(0, $celkovy_pocet, 1);
        $filter_id = array_fill(0, $celkovy_pocet, 1);
        for($i = 0; $i < $celkovy_pocet; $i++)
        {
            if($preprocessor[$i] == "SWT processor")
```

```
            {
                $processor_id[$i] = 2;
            }

            if($filter[$i] == "Random filter")
            {
                $filter_id[$i] = 2;
            }
            if($filter[$i] == "Pattern filter")
            {
                $filter_id[$i] = 3;
            }
            if($filter[$i] == "Track filter")
            {
                $filter_id[$i] = 4;
            }
        }



        $prescaler = $_POST['prescaler'];
        $enabled = $_POST['enabled'];
        $prescaler_id = $_POST['prescaler_id'];

        $lazyTrigger = $_POST['lazyTrigger'];
        $removeImages = $_POST['removeImages'];
        $filterFrame = $_POST['filterFrame'];
        $frameSource = $_POST['frameSource'];



        for($i = 0; $i < $puvodni_pocet; $i++)
        {
            if($to_delete[$i] == 1)
            {
                $database->delete_from_filter_prescaler(
                    ↪ $prescaler_id[$i]);
                continue;
            }
            if($changed[$i] == 1)
            {
                $database->update_filter_prescaler(
                    ↪ $processor_id[$i], $filter_id[$i],
                    ↪ $prescaler[$i], $enabled[$i],
                    ↪ $prescaler_id[$i]);
            }
        }
```

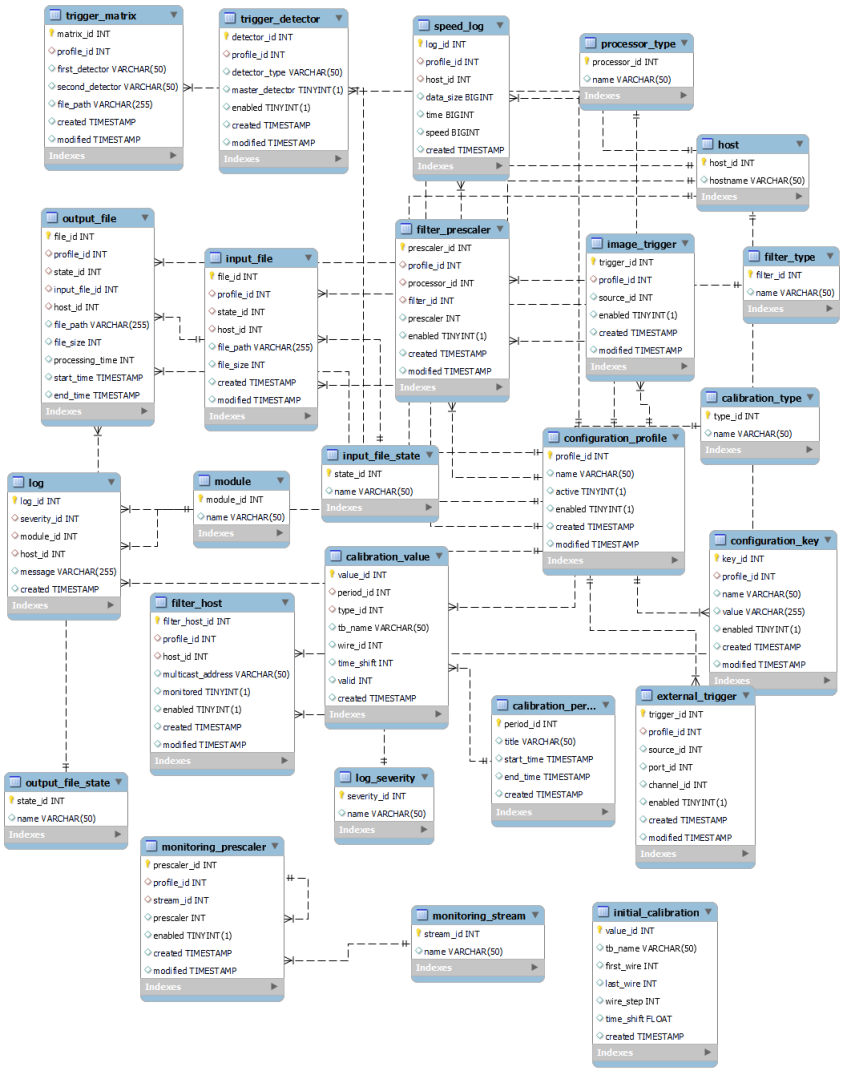
```
$preprocessor_new = array();         //V techto polich
   ↪ budou nova originalni data, ktera se ulozi
   ↪ hromadne do databaze
$filter_new = array();
$prescaler_new = array();
$enabled_new = array();
$j = 1;
for($i = $puvodni_pocet; $i < $celkovy_pocet; $i++)
{

    if($to_delete[$i] == 1)//if($changed[$i] == 1 ||
       ↪ $to_delete[$i] == 1)
    {
        continue;
    }

    $preprocessor_new = array_pad($preprocessor_new,
       ↪ $j, $processor_id[$i]);
    $filter_new = array_pad($filter_new, $j,
       ↪ $filter_id[$i]);
    $prescaler_new = array_pad($prescaler_new, $j,
       ↪ $prescaler[$i]);
    $enabled_new = array_pad($enabled_new, $j,
       ↪ $enabled[$i]);
    $j = $j+1;
}


if(!empty($preprocessor_new))
{
    $timestamp = array_fill(0,
       ↪ count($preprocessor_new), date('Y-m-d
       ↪ H:i:s'));

    $profile_id = array_fill(0,
       ↪ count($preprocessor_new),
       ↪ $_SESSION['profile']);

    $database->insert_into_filter_prescaler($profile_id,
       ↪ $preprocessor_new, $filter_new,
       ↪ $prescaler_new, $enabled_new, $timestamp);

}

//Data for configuration key table
$lazyTrigger_value = 'false';
if($lazyTrigger == 1)
{
    $lazyTrigger_value = 'true';
}
```

```
        $name = 'general/lazy_filtering';
        $database->update_configuration_key($lazyTrigger_value,
            ↪ $_SESSION['profile'], $name);

        $removeImages_value = 'false';
        if($removeImages == 1)
        {
            $removeImages_value = 'true';
        }
        $name = 'general/remove_images';
        $database->update_configuration_key($removeImages_value,
            ↪ $_SESSION['profile'], $name);

        $filterFrame_value = 'false';
        if($filterFrame == 1)
        {
            $filterFrame_value = 'true';
        }
        $name = 'general/insert_filter_frame';
        $database->update_configuration_key($filterFrame_value,
            ↪ $_SESSION['profile'], $name);

        $name = 'general/filter_source_id';
        $database->update_configuration_key(strval($frameSource),
            ↪ $_SESSION['profile'], $name);

        http_response_code(200);
    }
    catch(Exception $e)
    {
        if($e->getMessage()=="Connection to database failed")
        {
            http_response_code(500);
            exit;
        }
        else{
            http_response_code(501);
            exit;
        }
    }

}
exit;
```

Kód A.1: Skript general_settings_script.php

Příloha B

Er diagram databáze HLT



Obrázek B.1: ER diagram HLT databáze

Příloha C

Metody třídy HltDatabase

C.1 Metody využívající SQL příkaz SELECT

select_all_by_profile_id_from_configuration_key($profile_id)
Vybere všechna data z tabulky configuration_key na základě vyžádaného profile_id.

select_all_from_filter_type()
Vybere všechny hodnoty z tabulky filter_type.

select_all_from_processor_type()
Tato metoda slouží k načtení všech dat z tabulky processor_type.

select_all_hostname_from_host()
Vybere sloupec atributu hostname (jména hostitelských PC) z tabulky host.

select_all_logs()
Vybere všechny záznamy z tabulky log.

select_filter_id_from_filter_type($filter)
Načte hodnotu filter_id z tabulky filter_type podle zadaného jména filteru.

select_from_configuration_profile()
Vybere všechna data po řádcích z tabulky configuration_profile.

select_from_external_trigger($profile_id)
Vybere všechny řádky tabulky external_trigger, které mají požadované profile_id.

select_from_filter_host($profile_id)
Vybere řádky z tabulky filter_host se zadaným profile_id.

select_from_filter_prescaler($profile_id)
Vybere všechny hodnoty z těch řádků tabulky filter_prescaler, které mají požadované profile_id.

select_from_host()
Vybere všechny řádky z tabulky host.

select_from_host_where_host_id($host_id)
Vybere ty řádky z tabulky host, které mají zadané host_id.

select_from_input_file($profile_id)
Vybere řádky z tabulky input_file se zadaným profile_id.

select_from_monitoring_prescaler($profile_id)
Vybere řádky z tabulky monitoring_prescaler, které mají zadané profile_id.

select_from_trigger_detector($profile_id)
Vybere všechny řádky tabulky trigger_detector s požadovaným profile_id.

select_from_trigger_matrix($profile_id)
Vybere všechny řádky tabulky trigger_matrix, které mají požadované profile_id.

select_host_id_from_host($hostname)
Vybere atribut host_id z tabulky host, který je přiřazen k danému hostname (jméno hostitelského PC).

select_hostname_from_host($host_id)
Vybere atribut hostname (jméno hostitelského PC) z tabulky host podle host_id.

select_image_trigger($profile_id)
Načte všechny řádky z tabulky image trigger se zadaným profile_id.

select_last_profile_id_from_configuration_profile()
Vybere profile_id posledně vytvořeného profilu v tabulce configuration_profile.

selecet_name_from_configuration_profile($profile_id)
Vybere atribut name (jméno) z tabulky configuration_profile podle profile_id.

select_name_from_log_severity($severity_id)
Vybere atribut name (jméno) z tabulky severity podle zadaného severity_id.

select_name_from_module($module_id)
Vybere atribut name (jméno) z tabulky module podle module_id.

select_name_from_monitoring_prescaler()
Vybere sloupec atributu name (jména) z tabulky monitoring_prescaler.

select_name_from_monitoring_stream_where_stream_id($stream_id)
Vybere atribut name (jméno) z tabulky monitoring_stream podle zadaného stream_id.

select_new_logs($date)
Metoda sloužící k načítání záznamů z tabulky log od určitého časového okamžiku.

select_offline_log($date_start, $date_end)
Vybírá záznamy z tabulky log vytvořené v zadaném časovém úseku parametry $date_start a $date_end.

select_processor_id_from_processor_type($processor)
Tato metoda slouží k načtení hodnoty processor_id z tabulky processor_type podle zadaného jména processoru.

select_stream_id_from_monitoring_stream($stream_name)
Vybere atribut stream_id z tabulky monitoring_stream podle zadaného stream_name (jméno streamu).

select_value_from_conftiguration_key($profile_id, $name)
Vybere atribut value (hodnota) z tabulky configuration_key podle zadaného profile_id a name (jméno).

C.2 Metody využívající SQL příkaz INSERT

copy_new_into_configuration_key($profile_id_new_array, $names, $values, $enabled, $timestamp)
Metoda obdrží parametry (pole o stejné délce), které uloží po řádcích do tabulky configuration_key.

create_profile($name)
Vloží do tabulky configuration_profile nový profil se jménem zadaným parametrem $name.

insert_into_external_trigger($profile_id, $source_id_new, $port_id_new, $channel_id_new, $enabled_new, $timestamp)
Metoda obdrží parametry postupně odpovídající sloupcům tabulky external_trigger a vloží je do ni.

insert_into_filter_host($profile_id, $host_id_new, $multicast_address_new, $monitored_new, $enabled_new, $timestamp)
Metoda přijaté parametry ukládá postupně do řádků tabulky filter_host.

insert_into_filter_prescaler($profile_id, $preprocessor_new, $filter_new, $prescaler_new, $enabled_new, $timestamp)
Obdrží parametry postupně odpovídající sloupcům tabulky filter_prescaler a vloží je do ni.

insert_into_image_trigger($profile_id, $source_id_new, $enabled_new, $timestamp)
Metoda sloužící k ukládání hodnot do tabulky image_trigger.

insert_into_input_file($profile_id, $state_id_pom, $host_id_new, $data_file_new, $file_size_pom, $timestamp)
Metoda vkládá data do tabulky input_file.

insert_into_monitoring_prescaler($profile_id, $stream_id_new, $prescaler_new, $enabled_new, $timestamp)
Metoda vkládá parametry do tabulky monitoring_prescaler.

insert_into_trigger_detector($profile_id, $detector_type_new, $master_detector_new, $enabled_new, $timestamp)
Metoda přijaté parametry ukládá postupně do řádků tabulky trigger_detector.

insert_into_trigger_matrix($profile_id, $first_detector_new, $second_detector_new, $matrix_file_new, $timestamp)
Metoda obdrží parametry postupně odpovídající sloupcům tabulky trigger_matrix a vloží je do ni.

C.3 Metody využívající SQL příkaz UPDATE

update_configuration_key($value, $profile_id, $name)
Aktualizuje atribut value tabulky configuration_key v řádku, kde je daný profile_id a name (jméno).

update_external_trigger($source_id, $port_id, $channel_id, $enabled, $trigger_id)
Aktualizuje řádek tabulky external_trigger, který je označen zadaným trigger_id.

update_filter_host($host_id, $multicast_address, $enabled, $filter_host_id)
Aktualizuje hodnoty v řádku označeném příslušným filter_host_id v tabulce filter_host.

update_filter_prescaler($processor_id, $filter_id, $prescaler, $enabled, $prescaler_id)
Metoda aktualizuje hodnoty v řádku tabulky filter_prescaler, který má zadané prescaler_id.

update_image_trigger($source_id, $enabled, $trigger_id)
Aktualizuje řádek tabulky imagel_trigger, který je označen zadaným trigger_id.

update_input_file($host_id, $data_file, $file_id)
Metoda aktualizující řádek tabulky input_file, který nese označení daného file_id.

update_monitoring_prescaler($stream_id, $prescaler, $enabled, $prescaler_id)
V tabulce monitoring_prescaler aktualizuje hodnoty řádku se zadaným prescaler_id.

update_trigger_detector($detector_type, $master_detector, $enabled, $detector_id)
Tato metoda aktualizuje hodnoty řádku označeného podle zadaného detector_id v tabulce trigger_detector.

update_trigger_matrix($first_detector, $second_detector, $matrix_file, $matrix_id)
Tato metoda aktualizuje hodnoty v řádku se zadaným matrix_id v tabulce trigger_matrix.

C.4 Metody využívající SQL příkaz DELETE

Metoda vždy přijímá parametr pojmenovaný stejně jako primární klíč v dané tabulce.

delete_from_external_trigger($trigger_id)
Metoda maže řádek tabulky external_trigger s jedinečným trigger_id.

delete_from_filter_host($filter_host_id)
Tato metoda maže řádek hodnot označený jedinečným filter_host_id.

delete_from_image_trigger($trigger_id)
Metoda maže řádek tabulky image_trigger s jedinečným trigger_id.

delete_from_input_file($file_id)
Metoda maže řádek hodnot z tabulky input_file, který je označen daným file_id.

delete_from_filter_prescaler($prescaler_id)
Tato metoda maže řádek z tabulky filter_prescaler podle prescaler_id.

delete_from_monitoring_prescaler($prescaler_id)
Maže řádek z tabulky monitoring_prescaler označený zadaným prescaler_id.

delete_from_trigger_detector($detector_id)
Maže řádek tabulky trigger_detector, který je označen zadaným detector_id.

delete_from_trigger_matrix($matrix_id)
Maže řádek tabulky trigger_matrix podle matrix_id.

Příloha D

Používané zkratky

- AJAX - Asynchronous JavaScript and XML
- AMBER - Apparatus for Meson and Baryon Experimental Research
- ARPANET - Advanced Research Projects Agency Network
- CASTOR - CERN Advanced Storage manager
- CDN - Content Delivery Network
- CERN - Conseil Européen pour la Recherche Nucléaire
- COMPASS - Common Muon and Proton Apparatus for Structure and Spectroscopy
- CSS - Cascading Style Sheets
- DAQ - Data Acquisition System
- DBMS - Database Management System
- DHC - Data handling card
- DNS - Domain Name System
- FGPA - Field Programmable Gate Array
- GUI - Graphical User Interface
- HLT - High Level Trigger
- HTML - Hypertext Markup language
- HTTP - Hyper Text Transfer Protocol
- IMAP - Internet Message Access Protocol
- LAN - Local Area Network
- LDAP - Lightweight Directory Access Protocol

- LHC - The Large Hadron Collider
- OSI - The Open Systems Interconnection
- PHP - Hypertext Preprocessor
- RDBMS - Relational Database Management System
- SGML - Standard Generalized Markup Language
- SQL - Structured Query Language
- TCP/IP - Transmission Control Protocol/Internet Protocol
- URL - Uniform Resource Locator
- WAN - Wide Area Network
- WWW - World Wide Web
- XML - Extensive Markup Language

Příloha E

Obsah přiloženého CD

Přiložené CD obsahuje:

- bakalářská práce - soubor Bp_Vondruska.pdf
- zdrojové kódy - složka hltconf