VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VIRTUÁLNÍ LABORATOŘ PRO PYNQ

REMOTE LABORATORY FOR PYNQ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **BORIS VESELÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VOJTĚCH MRÁZEK, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

152803

Ústav:	Ústav počítačových systémů (UPSY)
Student:	**Veselý Boris**
Program:	Informační technologie
Název:	**Virtuální laboratoř pro PYNQ**
Kategorie:	Návrh číslicových systémů
Akademický rok:	2023/24

Zadání:

1. Seznamte se s možnostmi sdílení výpočetních prostředků více uživateli.
2. Seznamte se s technikami programování a evaluace na zařízení Xilinx Zynq a nadstavby PYNQ.
3. Zpracujte studii na výše uvedená témata.
4. Navrhněte systém umožňující většímu množství uživatelů testovat jejich návrhy číslicových systémů, zaměřte se zejména na škálovatelnost a robustnost systému.
5. Navržený systém implementujte.
6. Na vzorku uživatelů vyhodnoťte vlastnosti implementovaného systému.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Mrázek Vojtěch, Ing., Ph.D.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2023
Termín pro odevzdání:	16.5.2024
Datum schválení:	25.4.2024

Abstrakt

Táto práca slúži na poskytovanie zariadení PYNQ viacerým používateľom na výučbu predmetu IVH. Aplikácia virtuálneho laboratória poskytuje prístup k zariadeniam PYNQ pomocou tunelových spojení. V aplikácii je možné spravovať súbory uložené na serveri a registrovať si časové úseky, počas ktorých bude používateľovi umožnený prístup ku zariadeniam PYNQ pomocou tunelových spojení. Aplikácia umožňuje aj rolu administrátora, ktorý môže spravovať používateľov, ich rezervácie a jednotlivé zariadenia PYNQ. Aplikácia je napísaná pomocou jazykov PHP, rámca Nette a Go. Prínosom práce je zjednodušenie prístupu ku zariadeniam PYNQ v rámci výučby predmetu IVH.

Abstract

This thesis is used to provide PYNQ devices to multiple users for teaching the IVH subject. The remote laboratory application provides access to the PYNQ devices using tunnel connections. The application can manage the files stored on the server and register the time slots during which the user will be allowed to access the PYNQ devices using tunnel connections. The application also allows an administrator role that can manage users, their reservations and individual PYNQ devices. The application is written using PHP, the Nette framework and Go. The contribution of the work is to simplify access to PYNQ devices in the context of teaching the IVH subject.

Kľúčové slová

virtuálne laboratórium, FPGA, Zynq, PYNQ, rezervačný systém, PHP, Nette, Go, JavaScript, démon, tunel, maškaráda, presmerovanie komunikácie, REST API, webová aplikácia

Keywords

remote lab, FPGA, Zynq, PYNQ, reservation system, PHP, Nette, Go, JavaScript, daemon, tunnel, masquerade, communication redirection, REST API, web application

Citácia

VESELÝ, Boris. *VIRTUÁLNÍ LABORATOŘ PRO PYNQ*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vojtěch Mrázek, Ph.D.

VIRTUÁLNÍ LABORATOŘ PRO PYNQ

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vojtěcha Mrázeka, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Boris Veselý
14. mája 2024

Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu práce Ing. Vojtěchovi Mrázekovi, Ph.D., ktorý mi poskytol odbornú pomoc pri písaní tejto práce. Ďalej by som sa chcel poďakovať mojej rodine a blízkemu kruhu priateľov, ktorí mi boli oporou počas celého štúdia.

Obsah

Kapitola 1

Úvod

Na výučbu na fakulte sa používajú fyzické zariadenia Zynq s nadstavbou PYNQ, ktoré si každý študent musí vypožičať a každé zariadenie musí byť nastavené lektorom vyučovaného predmetu. Čo vytvára prácu naviac pre lektora aj pre študenta. Táto práca naviac by sa dala minimalizovať vytvorením systému, do ktorého by mali prístup študenti a lektori, ktorý by poskytoval prístup cez webový prehliadač k týmto zariadeniam, čím by bolo študentom umožnené pracovať so zariadeniami v rezervovaných časových úsekoch.

Táto práca slúži na poskytovanie zariadení PYNQ viacerým používateľom na výučbu predmetu IVH. Na týchto zariadenia si používatelia testujú svoje projekty. Preto bolo cieľom práce nájsť spôsob a vytvoriť aplikáciu, ktorá umožní používateľom férový a bezpečný prístup ku zdielaným zariadeniam, čím sa umožní používateľom takmer neobmedzený prístup k zariadeniam a to bez nutnosti ich vypožičania.

Pre aplikáciu je dôležité, aby každý používateľ mal férový prístup do systému, kde si môže vytvoriť rezerváciu na daný časový úsek, počas ktorého môže neprerušovane a bezpečne pracovať so zariadením PYNQ. Je dôležité, aby aplikácia bola bezpečná a neumožňovala používateľovi prístup k dátam iného používateľa, alebo prístup viacerým používateľom naraz k jednému zariadeniu. Aplikácia bude poskytovať rozhranie ako pre obyčajných používateľov, tak aj pre lektora, ktorý bude môcť v aplikácii vykonávať administratívne úlohy.

Nasledujúca kapitola 2 sa venuje problematike zdieľania prostriedkov viacerým používateľom, kde sú popísané možnosti, ako sa dajú dané prostriedky zdieľať. Kapitola 3. popisuje dosky ZYNQ a ich nadstavby PYNQ, kde je popísaná ich architektúra, ako fungujú a prečo môžu byť dobré pre výučbu hardvéru popisného jazyka. V kapitole 4 je riešený návrh samotného systému aplikácie virtuálneho laboratória pre študentov. Konkrétna implementácia tejto aplikácie je následne riešená v kapitole 5. Posledná kapitola 6 sa venuje otestovaniu systému na vzorke používateľov a následnému vyhodnoteniu vlastností implementovaného systému.

Kapitola 2

Zdieľanie prostriedkov viacerým užívateľom

Ak chce používateľ zdielať prostriedky viacerým používateľom, môže ich zdieľať pomocou niekoľkých možných implementácií. Napríklad pomocou zhlukov komoditných počítačov a plánovacích frontov, ktoré vystupujú ako jeden systém, do ktorého je nahraná úloha, ktorá je zaradená do frontu, systém túto úlohu vykoná a používateľ dostane výsledok vykonania tejto úlohy. Ďalej môže používateľ tieto prostriedky zdieľať pomocou rezervácií jednotlivých časových úsekov v rámci virtuálneho laboratória, v ktorom si používateľ rezervuje a počas tohto časového úseku je mu pridelené zariadenie z daného virtuálneho laboratória, na ktorom môže vykonávať prácu. Používatelia sa takto môžu jednoducho striedať a nie je potrebné implementovať napríklad ďalej popísaný algoritmus na určenie poradia úloh vo frontoch, ale je potrebné zabezpečiť iné opatrenia systému.

2.1 Prenosný dávkový systém

Zhluky komoditných počítačov sa objavili ako hlavne paralelné a distribuované platformy pre vysoko výkonné a vysoko dostupné výpočtové systémy. Sú prezentované spoločne ako jeden zjednotený zdroj pre koncových používateľov pomocou middlewarových technológií, ako je správa zdrojov a plánovací systém. Používatelia odosielajú úlohy v dávkach do frontu systému správy zdrojov a centralizovaný plánovač rozhodne, ako uprednostniť a prideliť prostriedky na vykonanie úloh. Plánovače by mali byť schopné poskytnúť signál spätnej väzby, ktorý používateľom bráni v odosielaní neobmedzeného množstva práce. [11]

Prenosný dávkový systém (Portable batch system - PBS) je flexibilný, POSIX kompatibilný systém dávkového zaraďovania a riadenia pracovného zaťaženia. Funguje na sieťových, multiplatformových prostrediach UNIX, vrátane heterogénnych zhlukov pracovných staníc, superpočítačov a masívne paralelných systémov. Tento projekt bol iniciovaný s cieľom vytvoriť flexibilný, rozšíriteľný systém dávkového spracovania, ktorý spĺňa jedinečné požiadavky heterogénnych počítačových sietí. Účelom PBS je poskytnúť dodatočné kontroly nad inicializáciou, alebo plánovaním vykonávania dávkových úloh a umožniť smerovanie týchto úloh medzi rôznymi hostiteľmi. Má tiež priateľské grafické rozhranie. [11]

2.1.1 Fronty a plánovanie

Predvolený plánovač v PBS je FIFO, ktorého správaním je maximalizovať využitie CPU. To znamená, že prechádza zoznamom úloh vo fronte a spúšťa akúkoľvek úlohu, ktorá sa

zmestí do dostupných zdrojov. Toto však účinne zabraňuje tomu, aby sa veľké úlohy vôbec spustili. Aby sa umožnilo spustenie veľkých úloh, tento plánovač implementuje mechanizmus „hladujúcich úloh“. Táto metóda môže fungovať v niektorých situáciách, ale existujú určité okolnosti, kedy tento postup neprináša požadované výsledky. V tomto čase prichádzajú do hry alternatívne plánovače, ktoré možno použiť s PBS. [11]

Výber politiky plánovania práce môže ovplyvniť čas, kedy sa úlohy používateľa začnú vykonávať, a ďalej ovplyvniť, či používatelia vedia použiť výpočtové zdroje spravodlivo a rozumne. V masívne paralelných systémoch spracovania boli algoritmy plánovania úloh kľúčovým faktorom ovplyvňujúcim využitie systémového procesoru. [5]

V súčasnosti sú bežné stratégie plánovania FCFS (First Come First Served), SPT (Shortest Processing Time), First Fit, Best Fit, Greedy a tak ďalej. V rámci tradičných zoskupení je tiež veľká základňa práce na plánovaní pracovných miest. Niektoré z vyspelejších systémov riadenia úloh sú platforma LSF (Load Sharing Facility), PBS Pro, Torque a Slurm. Neskôr sa objavili vylepšené algoritmy založené na backfillingu a jemu podobné. Hoci algoritmus backfillingu môže do určitej miery zlepšiť využitie CPU, môže nastať moment, keď nebudú existovať žiadne úlohy, ktoré by zaplnili voľný priestor CPU, keď parametre úlohy vo fronte nemôžu spĺňať podmienky algoritmu Backfilling. [5]

2.1.2 Problémy a riešenia

Každá úloha vstupuje do systému v určitom čase príchodu tým, že je zaradená do špecifického frontu (v závislosti od volieb používateľa a od charakteristík úlohy). Analýzou existujúcich stôp vykonávania pochádzajúcich z PBS sme určili odhadovaný čas čakania pre každý front, ktorý sa vzťahuje na každú úlohu, ktorú obsahuje. Pri odosielaní úlohy musí používateľ špecifikovať niekoľko informácií, vrátane maximálneho povoleného času vykonania, maximálneho počtu uzlov, ktoré sa majú použiť, a požadovaných zdrojov (jadrá, pamäť, GPU, MIC). Podľa konvencie systém PBS považuje každú úlohu, ako keby bola rozdelená na súbor presne identických „jednotiek úloh“, ktoré majú byť mapované na jednom uzle. Problém dispečingu vyžaduje priradiť čas začiatku každej čakacej úlohe a uzol každej z jej jednotiek. Mali by sa rešpektovať všetky limity kapacity zdrojov a výkonu, pričom sa zohľadní prítomnosť úloh, ktoré sa už vykonávajú. Jednotlivé aktivity nemajú žiadny konečný termín ani čas vydania (t. j. nemusia skončiť v určitom dátume alebo začať po určitom dátume), ani celkový rozsah nie je obmedzený žiadnou hornou hranicou.

Spolu s vyššie uvedenými zdrojmi v tomto probléme uvádzame ďalší obmedzený zdroj, výkon. To umožňuje modelovať a presadzovať obmedzenie výkonu. Úroveň obmedzenia výkonu, t.j. maximálne množstvo energie spotrebovanej systémom v akomkoľvek danom čase, je špecifikované používateľom a predstavuje kapacitu falošného zdroja energie; súčet celého výkonu spotrebovaného bežiacou úlohou plus súčet hodnôt výkonu súvisiacich s nečinnými zdrojmi nesmie nikdy prekročiť daný limit počas vykonávania celého plánu. Ďalšou dôležitou vecou, ktorú si treba všimnúť, je, že „zdroj“ energie nemá lineárne správanie, pokiaľ ide o výpočtovú záťaž: prvá aktivácia jadra v uzle spôsobuje väčšiu spotrebu energie pre zostávajúce jadrá v tomto uzle. Cieľom je skrátiť čakacie doby ako meradlo kvality služieb zaručenej používateľom superpočítačov. [2]

2.2 Virtuálne laboratóriá a časové úseky

Laboratóriá sú nevyhnutné pre výučbu technologických predmetov, kde sú potrebné praktické experimenty na získanie očakávaných kompetencií a odborných zručností. Preto virtu-

álne laboratórium, ktoré poskytne študentovi také isté možnosti ako normálne laboratórium, má veľa výhod. Študenti môžu pracovať doma so svojimi osobnými počítačmi s jednotlivými operačnými systémami, editormi a podobne. Majú prístup k hardvéru laboratória 24 hodín denne. Laboratórium je možné realizovať veľmi kompaktne, čím sa ušetrí veľa miesta a znížia sa aj náklady na personál. Výhody e-learningových systémov nie sú niečím novým a v oblasti strojárstva ho doteraz využíva množstvo iných univerzít. [10, 9] Mnohé z takýchto e-learningových kurzov sú však založené len na teoretickej alebo simulačnej báze. Aby bol zrealizovaný rozumný e-learningový kurz v oblasti strojárstva a boli využité vyššie uvedené dôvody, je potrebné vybudovať online laboratórium s programovateľnými hradlovými poľami (field programmable gate arrays - FPGA) pre vzdialený prístup. Jednotlivé FPGA dosky by boli pripojené k serveru na univerzite a budú programované serverom. Dosky FPGA budú napríklad monitorované pomocou kamier, aby používatelia videli stav dosiek, napr. blikajúce LED diódy alebo informácie na integrovanom displeji. [10]

Na správu online laboratória je potrebný server s nástrojom na správu zdrojov. Hlavnou výzvou je rozhodnúť o prístupe mnohých používateľov k obmedzenému počtu dosiek. Každý študent by mal dostať možnosť používať dosku a po použití musí byť prístup k doske uvoľnený pre ďalšieho študenta. Je veľmi dôležité, že neexistuje možnosť blokovania dosky na dlhší čas. Po pevnom časovom úseku by mal mať používateľ napríklad na krátky čas zakázaný prístup k rezervačnému systému. Rezervačný systém musí byť zabezpečený proti nesprávnemu prístupu a v najlepšom prípade využívať bezpečnostné mechanizmy základného operačného systému. [10]

2.2.1 Rezervácia časových úsekov

Každá implementácia virtuálneho laboratória má svoj špeciálny rezervačný systém, ktorý umožňuje používateľom rezervovať si časový úsek na prácu na zariadení. Používateľovi je buď umožnený prístup v časovom úseku ktorý mu vyhradí inštruktor kurzu počas hodín výučby, čo nie je veľmi efektívne. Alebo si teda používateľ pomocou rezervačného systému vytvorí rezerváciu sám na inštruktorom predom definovaný čas, počas ktorého môže vykonať požadovaný experiment, čo je efektívnejšie. Počas tohto časového úseku má študent prístup k rezervovanému zariadeniu už podľa ďalšej implementácie systému. Je potrebné zaručiť, aby po každom študentovi bolo zariadenie vyčistené a tak nedošlo k prístupu študenta k dátam iného študenta. [9, 8]

Hlavnými objektívmi systému na rezerváciu časových úsekov sú férový pristúp, tak, aby každý používateľ mal prístup k jednotlivým doskám FPGA, efektivita, to znamená čo najnižšie časy, v ktorých sú jednotlivé dosky FPGA nečinné, škálovateľnosť rezervačného systému, aby dokázal zvládnuť narastajúci nápor používateľov, a to bez degradácie výkonnosti a následne poskytnutie jednoduchého a intuitívneho rezervačného systému používateľom. [9]

Existuje niekoľko prístupov k implementácii rezervačného systému.

- Centralizovaný systém správy časových úsekov.
- Systém implementovaný pomocou decentralizovaného zdieľaného registra.
- Hybridný prístup k implementácii rezervačného systému pomocou kombinácie prvých dvoch implementácií.

Pre účely výučby kurzu na univerzitách jednoznačne postačuje centralizovaný systém správy časových úsekov, ktorý beží na serveri na univerzite a študenti sa naň môžu pripájať. [10, 8]

Kapitola 3

Xilinx Zynq a nadstavba PYNQ

Na výučbu predmetu IVH sa na fakulte používajú dosky Zynq, ktoré umožňujú študentom vytvárať vlastné periférie, čo môže byť užitočné pre pochopenie návrhu procesorov, ale aj pre tvorbu vlastných ovládačov, napríklad displejov či hardvérových akcelerátorov.

V tejto kapitole sa preto venujem zariadeniu Xilinx Zynq, jeho architektúre, ako sa dané zariadenie dá programovať a následne jeho nadstavbe PYNQ.

3.1 Xilinx Zynq

Zynq je System on Chip (SoC) – čo znamená, že okrem integrácie väčšiny, ak nie všetkých komponentov počítača do jedného čipu, môžu vývojári využiť aj technológiu FPGA. FPGA, sú zvyčajne samostatné komponenty, ktoré sa používajú na prototypovanie vlastných systémových čipov alebo na navrhovanie hardvéru, ktorý sa neskôr vyvinie do integrovaných obvodov špecifických pre aplikáciu (ASIC). [6]

Zynq má jedinečnú architektúru, ktorá kombinuje systém spracovania (PS) a programovateľnú logiku (PL). PS pozostáva z aplikačných procesorov ARM Cortex-A9 a štandardných komunikačných rozhraní, zatiaľ čo PL je založený na technológii Xilinx Artix-7 alebo Kintex-7 FPGA, čo umožňuje vlastný vývoj hardvéru. [4]

3.1.1 Systém spracovania (PS)

Processing System (PS) je v podstate mozgom Zynq SoC, ktorý je založený na jednom alebo viacerých jadrách ARM Cortex-A9. PS obsahuje nielen centrálnu procesorovú jednotku, ale aj bohatú sadu periférnych rozhraní a pamäťových radičov. Niekoľko kľúčových komponentov PS:

- Jadrá ARM Cortex-A9 v Zynq PS ponúkajú výkonné výpočtové schopnosti vhodné pre operačné systémy ako Linux, úlohy v reálnom čase a aplikačný kód.
- PS obsahuje radiče DDR, ktoré poskytujú priame pripojenie k pamäti DDR3 alebo DDR4 a ponúkajú rýchly a efektívny prístup k dátam.
- Zariadenie obsahuje integrovanú jednotku riadenia hodín, ktorá poskytuje signály hodín pre PS aj PL, čo je nevyhnutné pre časovanie a riadenie. [4]

3.1.2 Programovateľná logika (PL)

Programovateľná logika (PL) ako časť zariadenia Zynq SoC je vysoko flexibilná FPGA oblasť odvodená od Xilinx Artix-7, alebo Kintex-7 rodiny FPGA. Umožňuje implementovať vlastnú hardvérovú logiku priamo na čipe, ktorá môže byť prispôsobená špecifickým potrebám aplikácie. PL možno použiť na rôzne účely:

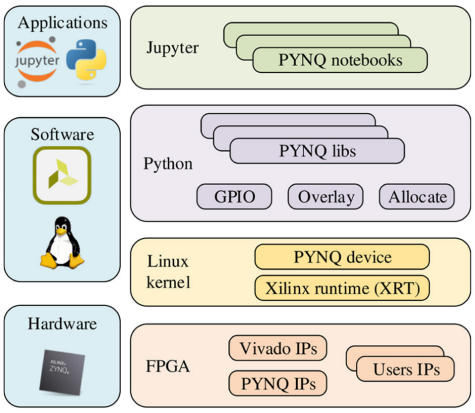
- Vlastné hardvérové funkcie, pomocou ktorých je možné implementovať algoritmy, alebo spracovať úlohy v hardvéri tak, aby sa urýchlili nad rámec toho, čo je možné pomocou softvéru.
- Pomocou prispôsobenia rozhrania je možné vytvoriť nové vlastné rozhrania, alebo podporovať neštandardné I/O protokoly, ktoré nie sú dostupné v rámci PS.
- Je možné využiť FPGA možnosti paralelného spracovania na spracovanie veľkých dátových tokov alebo spracovania úloh v reálnom čase. [4]

3.1.3 Vytvorenie programového súboru

Výhodou dosiek Zynq je, že používateľ si môže vytvárať vlastné periférie pomocou programu Xilinx Vivado, kde používateľ použije `Vivado IP Integrator` na vytvorenie nového blokového dizajnu, do ktorého pridá `Zynq processing system` blok z rozsiahleho IP katalógu programu. Používateľ následne môže tento blok upraviť podľa toho, ako potrebuje tým, že povolí špecifické periférie, ako napríklad univerzálny asynchrónny prijímač/vysielač (universal asynchronous receiver/transmitter - UART). Ďalej si používateľ môže vytvoriť vlastné špeciálne periférie, ktoré môžu slúžiť ako jednoduché rozhranie všeobecného vstupu/výstupu (general-purpose input/output - GPIO), alebo napríklad aj viac zložité komunikačné rozhranie. Pre tieto periférie je dôležité definovať ich rozhrania, cez ktoré budú komunikovať a implementovať ich funkcionalitu napríklad v jazyku VHDL. Následne sa táto periféria dá pridať do blokového dizajnu a pripojiť k bloku `Zynq PS` cez vhodnú zbernicu. Ak používateľ spravil všetko správne, môže pomocou programu Vivado vytvoriť programový súbor, ktorý slúži na naprogramovanie časti FPGA dosiek Zynq SoC. [12, 13, 4]

Dosky Zynq umožňujú ďalšiu interakciu s FPGA časťou, a to pomocou softvéru. Požívateľ po vytvorení programového súboru môže pomocou programu Vivado exportovať dizajn do aplikácie `Xilinx Software Development kit (SDK)`. V `SDK` si používateľ vytvorí nový aplikačný projekt, kde sú už predvolené šablóny ako `‘Hello world‘`, ktoré môže použiť. Po nahratí programového súboru do zariadenia Zynq a tým naprogramovania FPGA môže používateľ vytvoriť ovládače pre ním vytvorené periférie, a to pomocou jazyka C a pamäťového prístupu. Používateľ môže tieto periférie inicializovať pomocou `Xilinx API`, navrhnúť funkcie, ktoré do nich budú zapisovať alebo z nich čítať a následne spracovávať prerušenia. Túto aplikáciu si používateľ preloží v programe `SDK` a následne ju spustí na Zynq doske. Pomocou nástrojov programu `SDK` následne môže používateľ sledovať výstupy alebo ladiť svoj program. [12, 13, 4]

Pre interaktívne vývojové prostredie je možné použiť napríklad nástroje ako jupyter notebook v kombinácii s nadstavbou PYNQ (Python produktivita pre Zynq), ktorá umožňuje vysoko-úrovňový prístup k dizajnom použitím jazyka Python, čím napríklad zjednoduší testovanie alebo vizualizáciu dát. [3]



Obr. 3.1: Celkový rámec používania PYNQ na vývoj Zynq. [7]

3.2 Nadstavba PYNQ

PYNQ je open-source projekt od AMD. Poskytuje rámec založený na Jupyter s Python API na používanie platforiem AMD Xilinx Adaptive Computing. [3]

Na zariadeniach PYNQ je možné spustiť Linuxové programy, ako aj spúšťať programy Python na prepojenie a ovládanie rôznych I/O modulov, ktoré sú naprogramované v FPGA časti. PYNQ umožňuje syntézu hardvéru pomocou jazykov ako je VHLD alebo Verilog, zatiaľ čo ARM časť zariadenia je možné použiť ako Raspberry PI. [3]

3.2.1 Vrstvy v PYNQ

PYNQ pozostáva z troch vrstiev: aplikačná vrstva, softvérová vrstva a hardvérová vrstva. Celkový rámec je znázornený na obrázku 3.1. [7]

Aplikačná vrstva je zvyčajne kód Pythonu vo forme notebooku Jupyter, ktorý beží na ARM CPU vnútri Zynq SoC. Stredná vrstva v rámci PYNQ je softvérová. Táto vrstva obsahuje knižnice Python a interakciu s IP vo vnútri FPGA cez ovládače operačného systému. Niekoľko ovládačov je poskytovaných prostredníctvom knižníc PYNQ na interakciu s IP. Rozhranie sa nazýva prekrytie (overlay) a používa sa na programovanie FPGA a správu IP. Poslednou hardvérovou vrstvou v rámci PYNQ je bitový súbor naprogramovaný do FPGA, dané bitové súbory je možné vytvárať napríklad pomocou programu Vivado a užívateľ ich môže spúšťať a pracovať s nimi pomocou Jupyter Notebooku, ktorý PYNQ hosťuje. Interakcia medzi softvérovou vrstvou a hardvérovou vrstvou prebieha pomocou DMA alebo pamäťovo mapovaných rozhraní. [1]

3.2.2 Jupyter Notebook

Jupyter Notebook je interaktívne počítačové prostredie dostupné cez prehliadač. Je v ňom možné vytvárať dokumenty Jupyter Notebooku, ktoré obsahujú živý kód, interaktívne widgety, grafy, komentáre, rovnice, obrázky a videá. Dosku podporujúcu nadstavbu PYNQ je možné jednoducho naprogramovať a ovládať z Jupyter Notebook pomocou Pythonu.

Notebook je schopný spúšťať kód v širokej škále jazykov. Každý notebook je však spojený s jedným jadrom. Pynq a tento notebook je spojený s jadrom IPython, na ktorom je spustený kód Python. Jupyter Notebook kombinuje tri komponenty:

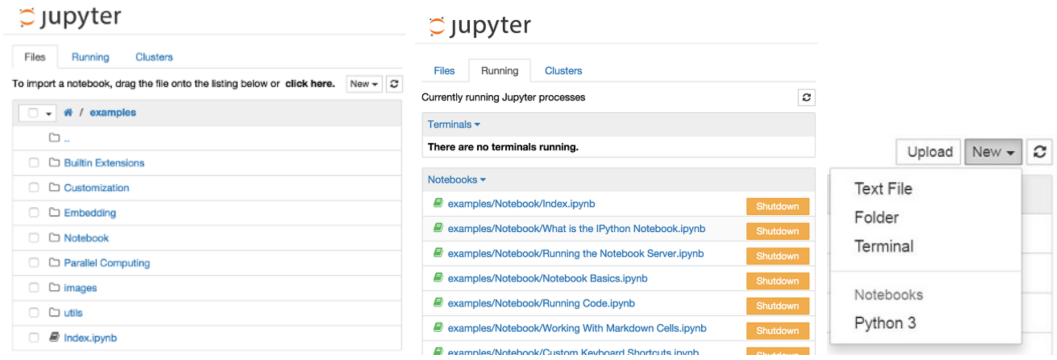
- Webová aplikácia notebooku: Interaktívna webová aplikácia na interaktívne písanie a spúšťanie kódu a vytváranie dokumentov notebooku.
- Jadrá: Samostatné procesy spustené webovou aplikáciou notebooku, ktorá spúšťa kód používateľov v danom jazyku a vracia výstup späť do webovej aplikácie notebooku. Jadro tiež zvláda veci ako výpočty pre interaktívne widgety, vypĺňanie kariet a introspekciu.
- Dokumenty poznámkového bloku: Samostatné dokumenty vo webovej aplikácii, ktoré obsahujú reprezentáciu celého obsahu poznámkového bloku vrátane vstupov a výstupov výpočtov, popisného textu, rovníc, obrázkov a multimediálnych reprezentácií objektov. Každý dokument poznámkového bloku má svoje vlastné jadro.

Používateľské rozhrania ako správa súborov, možnosti vytvorenia nového súboru a prehlad bežiacich dokumentov poznámkového bloku sú znázornené na obrázku 3.2. [3]

3.2.3 Prístupy a práva

K linux shellu sa dá dostať klasicky pomocou seriálového pripojenia, po LAN pripojení a nastavení cez SSH klient, ale aj cez webovú aplikáciu Jupyter Notebook. Na prístup k linuxu sa musí užívateľ verifikovať, je možné mať viacej rôznych používateľov s rôznymi právami a v rôznych skupinách.

Jupyter Notebook aplikácia beží pod používateľom `root`, ale je možné nastaviť, aby pri otvorení napríklad terminálu bol prihlásený dopredu zvolený používateľ. Nastaviť sa dá aj hlavný pracovný priečinok pre aplikáciu Jupyter Notebook. [3]



Obr. 3.2: Používateľské rozhranie Jupyter Notebooku [3]

Kapitola 4

Návrh

V tejto kapitole je podrobne popísaný návrh aplikácie, najprv popisujem vysoko-úrovňový diagram aplikácie a po ňom nasleduje popis jednotlivých komponentov aplikácie. Vzhľadom k tomu, že ide aj o tvorbu používateľského prostredia pre študentov aj lektorov, predstavím diagram použitia. Bude tu navrhnutý databázový model pomocou ER diagramu a v závere sa budem venovať metodike prepojenia jednotlivých kľúčových súčastí systému.

Účelom tejto aplikácie bude poskytnúť študentom online a časovo neobmedzený prístup ku zariadeniam, ktoré by si inak museli vypožičať od školy a sami nastaviť.

Na základe zadania navrhujem aplikáciu, ktorá bude dostupná cez webový prehliadač, pre študenta daného predmetu, kde po prihlásení bude môcť si rezervovať časový úsek, počas ktorého bude môcť na danom zariadení pomocou hosťovaného Jupyter Notebooku spúšťať predom preložené binárne súbory z programu Vivado. Ďalej v tejto webovej aplikácii bude môcť, aj mimo rezervovaného časového úseku, spravovať svoje súbory, s ktorými pracoval na registrovanom zariadení.

Na základe vypracovanej štúdie na tému zdieľania prostriedkov viacerým používateľom som sa rozhodol, že aplikáciu implementujem pomocou rezervácie jednotlivých časových úsekov, čo bude jednoduchšie a pre účely výučby predmetu IVH vhodnejšie ako použitie plánovacích frontov. Takto bude možné každému používateľovi poskytnúť počas rezervovaného časového úseku interaktívne prostredie Jupyter Notebook, pomocou ktorého bude môcť interagovať s rezervovaným koncovým zariadením PYNQ a tak isto so svojimi vlastnými perifériami, ktoré si pre FPGA časť vytvoril, čo by nebolo možné pri implementácii plánovacích frontov.

K tejto aplikácii bude mať prístup aj admin, ktorý po prihlásení sa bude môcť nahrávať nových užívateľov z `csv` súboru, spravovať rezervácie všetkých študentov a blokovať alebo odblokovať jednotlivé PYNQ zariadenia pri poruche alebo údržbe.

Z pohľadu bezpečnosti sa od aplikácie očakáva, aby každý používateľ mal zabezpečený prístup k svojim dátam, ktoré mu iný používateľ nebude môcť zmeniť. Takisto aj prístup na jednotlivé koncové zariadenia PYNQ bude zabezpečený len pre daného používateľa, ktorý si vytvoril rezerváciu a každé zariadenie bude pre tohto používateľa pripravené, a tak nebude možné, aby mal prístup k dátam iného používateľa.

V aplikácii bude možné si rezervovať časový úsek najviac na jeden deň dopredu. V rezervovanom časovom úseku sa užívateľovi objaví tlačidlo, ktorým sa mu z jeho zariadenia vytvorí cez webový server tunel na rezervované zariadenie a zobrazí sa mu na webovej stránke, stránka Jupyter Notebooku hosťovaného na ním rezervovanom zariadení, kde už bude môcť pracovať podľa svojich potrieb.

4.1 Prípady použitia aplikácie

V tejto sekcii popisujem detailne každú interakciu používateľa a aplikácie. Prípady použitia aplikácie sú dôležitou časťou návrhu aplikácie, pomáhajú s pochopením funkčných požiadaviek systému, uľahčujú komunikáciu medzi zainteresovanými stranami a poskytnú základ na vytváranie testovacích prípadov.

V aplikácii budú vystupovať nasledovní používatelia: študent, admin a čas.

- Študent je klasický používateľ, ktorého úlohou v aplikácii je vytvoriť si a manažovať rezervácie, manažovať a pristupovať k svojim súborom a pristupovať k rezervovaným zariadeniam PYNQ.
- Admin bude mať prístup ku všetkým funkciám ako študent, ale navyše bude mať prístup k administratívnemu rozhraniu, kde bude môcť zobrazovať a mazať rezervácie všetkých študentov, zablokovať a odblokovať zariadenia PYNQ podľa potreby, zobrazovať prebiehajúce pripojenia na jednotlivé zariadenia PYNQ a nahrávať nové používateľské účty do aplikácie.
- Úlohou času v aplikácii bude zobrazovanie tlačidla na prístup k rezervovanému zariadeniu PYNQ pre jednotlivých používateľov, ak im aktuálne prebieha rezervovaný časový úsek. Po uplynutí časového úseku bude používateľ mať stále možnosť pokračovať vo svojej práci na teraz už nie rezervovanom zariadení, pokým ďalší používateľ, ktorý má toto zariadenie rezervované, nevytvorí pripojenie na toto zariadenie.

V ďalších sekciách dopodrobna popisujem každý prípad použitia aplikácie zobrazený na obrázku diagramu použitia 4.1.

4.1.1 Zobrazenie používateľových rezervácií

Keďže na vytvorenie alebo zrušenie rezervácie používateľa je potrebné najprv si zobraziť svoje rezervácie, popíšem tento prípad použitia ako prvý. Používateľ bude mať prístup na stránku cez tlačidlo v navigačnej lište, kde sa mu zobrazia všetky časové úseky na ďalší deň. Každý časový úsek bude zobrazovať informáciu, koľko zariadení je už rezervovaných z koľkých voľných. Pre každý časový úsek bude zobrazené tlačidlo rezervovať alebo zrušiť rezerváciu podľa toho, či má študent v danom časovom úseku vytvorenú rezerváciu.

4.1.2 Vytvorenie rezervácie

Po kliknutí na tlačidlo rezervovať bude študentovi vytvorená rezervácia na daný vybraný časový úsek. Klientska časť kontaktuje serverovú aplikáciu, ktorá následne spraví záznam v databáze s používateľovým ID a zvoleným časovým úsekom.

4.1.3 Zmazanie rezervácie

Po kliknutí na tlačidlo zrušiť rezerváciu bude študentovi zrušená rezervácia na daný vybraný časový úsek. Klientska časť kontaktuje serverovú aplikáciu, ktorá následne vymaže záznam v databáze podľa používateľovho ID a zvoleného časového úseku.

4.1.4 Zobrazenie používateľových súborov

Podobne ako pri práci s rezerváciami, ak bude chcieť používateľ pracovať so svojimi súbormi, bude si ich najprv musieť zobraziť. Po kliknutí na tlačidlo v navigačnej lište sa používateľovi

zobrazí stránka so všetkými jeho súbormi, ktoré má uložené na serveri. Používateľovi sa najprv zobrazia všetky priečinky a súbory v nich, zobrazenie týchto priečinkov a súborov bude znázorňovať strom pre lepšiu orientáciu.

4.1.5 Nahrať súbory

Používateľ bude mať možnosť nahrať súbory priamo na stránke, kde budú zobrazené všetky súbory. Na nahratie súboru bude použitý formulár, ktorý kontaktuje serverovú aplikáciu, ktorá súbor spracuje a uloží do používateľovho priečinku.

4.1.6 Stiahnuť/Zmazať súbory

Používateľ si bude môcť každý súbor alebo priečinok stiahnuť alebo vymazať. Každý súbor/priečinok bude mať tlačidlo, vďaka ktorému si ho používateľ bude môcť stiahnuť či vymazať. Pri mazaní priečinkov aplikácia zmaže celý obsah priečinka. Pri sťahovaní priečinka aplikácia vytvorí z vybraného priečinka `zip` súbor, ktorý následne pošle používateľovi. Používateľovi bude tak isto umožnené stiahnuť/zmazať celý obsah jeho osobného priečinka na serveri.

4.1.7 Pristúpiť k rezervovanému zariadeniu

Na pristúpenie k rezervovanému zariadeniu si bude musieť používateľ rezervovať časový úsek; keď tento časový úsek nastane, aplikácia zobrazí používateľovi tlačidlo v navigačnej lište. Po kliknutí na toto tlačidlo serverová aplikácia vytvorí pomocou démona spojenie medzi klientom a zariadením PYNQ. Tak isto toto PYNQ pripraví na prácu nového používateľa, a to tým, že reštartuje službu Jupyter Notebook, odpojí pracovný adresár predošlého používateľa a pripojí pracovný adresár aktuálneho používateľa, následne bude používateľ presmerovaný na stránku aplikácie, kde bude už vložená hosťovaná stránka Jupyter Notebooku.

4.1.8 Prihlásenie/Odhlásenie

Používateľ sa bude môcť prihlásiť a odhlásiť z aplikácie pomocou prihlasovacích údajov, ktoré mu budú poskytnuté adminom aplikácie. Prihlasovacie mená a zahašované heslá používateľov budú uložené v databáze.

4.1.9 Nahranie nových používateľov

Admin bude mať možnosť nahrať nových používateľov v administračnom rozhraní. Admin nahrá `csv` súbor s používateľskými menami pomocou formulára, serverová aplikácia tieto mená spracuje, každému používateľovi vygeneruje heslo, uloží jeho prihlasovacie údaje do databázy a následne vytvorí súbor s používateľskými menami a heslami, ktorý uloží do adminovho správcu súborov.

4.1.10 Zobraziť rezervácie všetkých používateľov

Admin si bude môcť v administračnom rozhraní zobraziť rezervácie všetkých používateľov. Na stránke bude zobrazený zoznam všetkých časových úsekov na najbližší deň a pod každým časovým úsekom bude zoznam všetkých používateľských rezervácií, každá rezervácia bude mať tlačidlo na zmazanie.

4.1.11 Zrušiť rezerváciu používateľovi

Admin bude môcť zrušiť akúkoľvek rezerváciu používateľovi pomocou tlačidla zrušiť vedľa danej rezervácie. Po stlačení tlačidla serverová aplikácia bude zavolaná a následne rezerváciu vymaže z databázy. Admin následne dostane notifikáciu o úspešnom alebo neúspešnom zmazaní.

4.1.12 Zablokovať/Odblokovať zariadenie

Admin bude vidieť stav každého zariadenia PYNQ v systéme a bude mať možnosť toto zariadenie zablokovať alebo odblokovať podla potreby. Keď admin zariadenie zablokuje a bude na ňom aktuálne prebiehať komunikácia medzi zariadením a klientom, bude táto komunikácia pomocou démona zrušená a zariadenie vyčistené. Následne zariadenie bude prepnuté do stavu "DISABLED" a všetky rezervácie v budúcnosti, ktoré by teraz prevyšovali maximálny počet zariadení, budú zrušené. Adminovi príde potvrdenie o tom, že zariadenie bolo zablokované a údaje o zrušenej komunikácii a rezerváciách sa mu uložia do správcu súborov.

4.2 Prehľad aplikácie na vysokej úrovni

Aplikácia virtuálne laboratórium je navrhnutá na zefektívnenie procesu výučby jazyka VHDL na zariadeniach PYNQ. Platforma bude pozostávať z komponentov: klientske rozhranie, serverová aplikácia, databáza a jednotlivé cieľové zariadenia PYNQ.

Prepojenie medzi týmito komponentmi je znázornené na obrázku 4.2. Používateľ sa bude môcť pripojiť k webovému serveru, kde bude bežať webová aplikácia s klientskym rozhraním, ktoré umožní užívateľovi rezervovať si časový úsek na prácu so zariadením PYNQ. Na serveri bude bežať démon, ktorý bude komunikovať so serverovou aplikáciou, pomocou REST API a na základe tejto komunikácie bude poskytovať informácie alebo vykonávať jednotlivé príkazy, ako napríklad vytvorenie tunela medzi PYNQ zariadením a užívateľom, vyčistenie zariadenia po užívateľovi a príprava zariadenia pre ďalšieho užívateľa.

Zariadenia PYNQ budú pripojené na lokálnej sieti s webovým serverom, takže sa k nim užívateľ bez vytvorenia tunela nebude môcť dostať. Každé zariadenie PYNQ bude mať na jeho IP adresu a port, na ktorom hostuje Jupyter Notebook, priradený port na strane webového serveru v predsmerovacej NAT tabuľke. V posmerovacej NAT tabuľke bude na webovom serveri nastavená maškaráda na všetky zariadenia. Následne podľa potreby bude démon vytvárať pravidlá v presmerovacej tabuľke pre jednotlivých používateľov, ktorí majú aktuálne prebiehajúci registrovaný časový úsek.

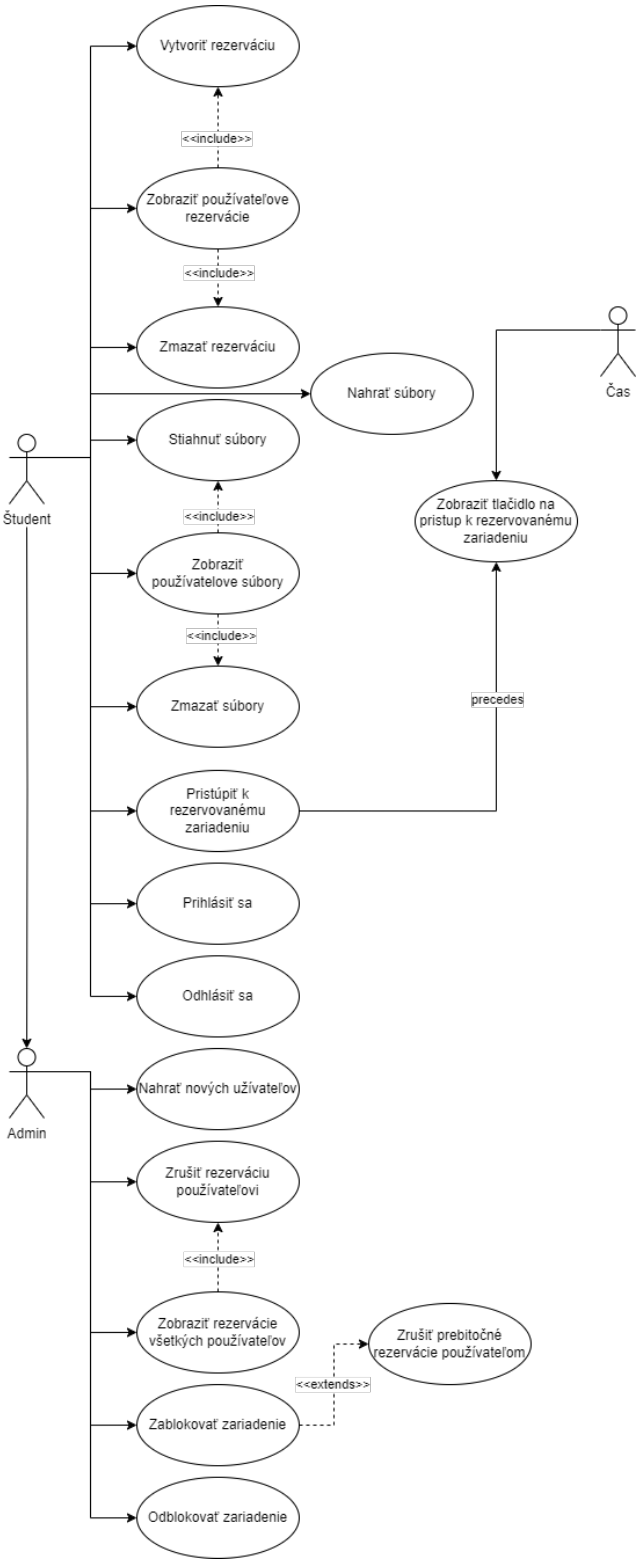
Webový server bude komunikovať aj s jednoduchou databázou, ktorá bude slúžiť na ukladanie prihlasovacích údajov používateľov a jednotlivých registrácií časových úsekov používateľov.

4.3 Komponenty aplikácie

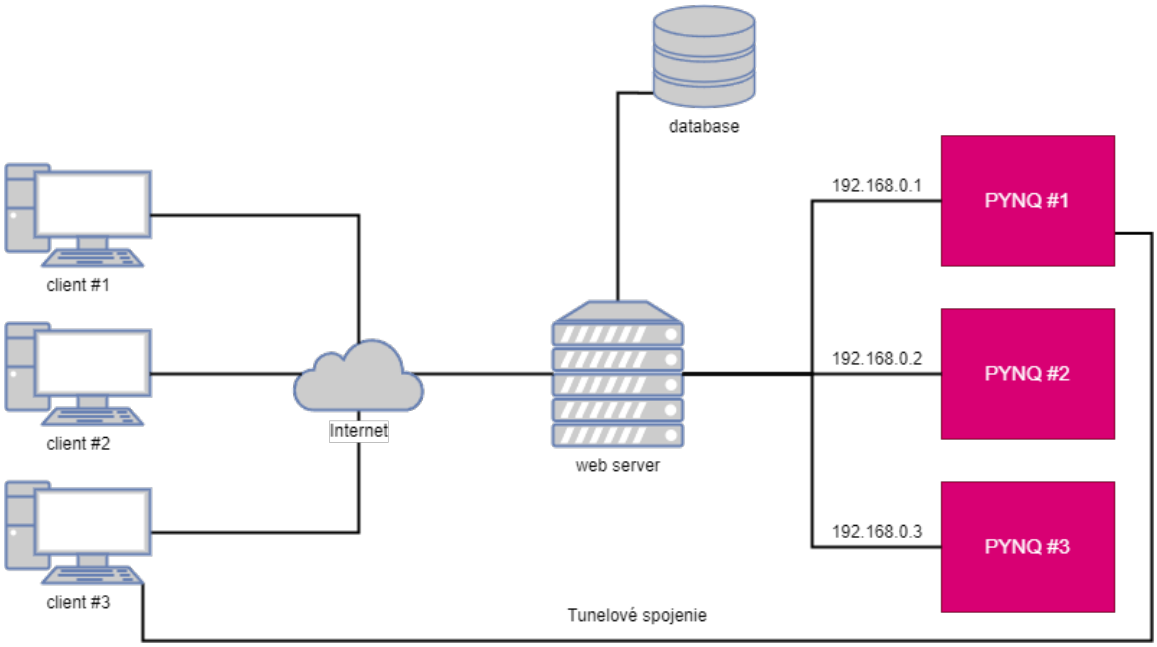
Aplikácia bude pozostávať z troch hlavných komponentov: klientske rozhranie, serverové rozhranie a jednotlivé zariadenia PYNQ.

4.3.1 Klientske rozhranie

Klientske rozhranie, kam budú mať používatelia prístup, umožní používateľom registrovať si jednotlivé časové úseky pre prácu na zariadeniach PYNQ. Tak isto aj spravovať svoje



Obr. 4.1: Diagram použitia



Obr. 4.2: Prehľad na vysokej úrovni

registrácie. Pomocou tohto rozhrania im bude umožnená práca na jednotlivých koncových zariadeniach PYNQ pomocou vloženia hosťovanej Jupyter Notebook web stránky z registrovaného zariadenia PYNQ na hlavnú stránku webovej aplikácie. Pomocou tejto Jupyter Notebook stránky bude užívateľ nahrávať a spúšťať pred-kompilované binárne súbory, vytvárať Jupyter Notebook dokumenty, môcť ovládať zariadenie pomocou spomínaných Jupyter Notebook dokumentov použitím programovacieho jazyka Python, alebo dokonca spustiť si terminál s Linux shell prostredím a spúšťať príkazy priamo tam.

Klientske rozhranie umožní používateľom spravovať svoje pracovné súbory aj mimo registrovaný časový úsek, a to pomocou súborového manažéra, ktorý umožní používateľom mazať, nahrávať a sťahovať súbory, ktoré budú dostupné aj na zaregistrovanom zariadení PYNQ.

Klientske rozhranie používateľom neumožní vytvoriť si účet, a to kvôli bezpečnosti. Používateľské účty bude môcť vytvárať iba admin tejto aplikácie, ktorý v aplikácii v administračnom rozhraní nahrá `csv` súbor s používateľskými menami a serverová aplikácia, každému používateľovi vygeneruje náhodne heslo a vytvorí mu účet. Admin potom vo svojom súborovom manažéri v aplikácii bude mať `csv` súbor, ktorý bude obsahovať používateľské mená s vygenerovanými heslami a vlajkou úspechu vytvorenia používateľského účtu pre každého používateľa.

Admin si tak isto bude môcť zobraziť všetky rezervácie používateľov a ak bude niekto schválne blokovať ostatných užívateľov, tak mu tieto rezervácie bude môcť zrušiť. Admin uvidí stav každého zariadenia PYNQ. Ak k nemu v danom momente bude mať niekto vytvorený tunel, admin uvidí podrobnosti tohto tunela, ako je IP adresa používateľa, jeho používateľské meno a ID v databáze. Admin bude môcť tieto zariadenia spravovať tak, že každé z nich bude môcť zablokovať alebo odblokovať podľa potreby, bez nutnosti zastavenia celej aplikácie, ak napríklad nastane porucha na niektorom zo zariadení, alebo sa bude jednať len o údržbu.

4.3.2 Serverové rozhranie

Serverová aplikácia sa postará o celú obchodnú logiku aplikácie, ako vytváranie rezervácií, autentifikácia používateľov, komunikácia s databázou, komunikácia s démonom, pomocou ktorého bude spravovať všetky zariadenia PYNQ, vytvárať a odstraňovať pravidlá v smerovacích tabuľkách, pre pripojenie používateľa k jednotlivým zariadeniam PYNQ. Databáza bude ukladať všetky rezervácie používateľov a prihlasovacie dáta používateľov.

Démon bude spravovať zariadenia PYNQ pomocou SSH klienta a upravovať pravidlá IP tabuliek. Ďalej bude komunikovať so serverovou aplikáciou pomocou REST API endpointu. Serverová aplikácia bude pomocou démona spravovať všetky pravidlá v IP tabuľkách tak, aby na každé zariadenie PYNQ mal prístup len jeden používateľ, ktorý si to zariadenie registroval na daný časový úsek. Na spravovanie týchto pravidiel v IP tabuľkách bude démon vytvárať smerovacie pravidlá, čím sa zaistí presmerovanie komunikácie pomocou tunela cez hosťovské zariadenie medzi zariadením PYNQ a používateľom.

4.3.3 PYNQ zariadenia

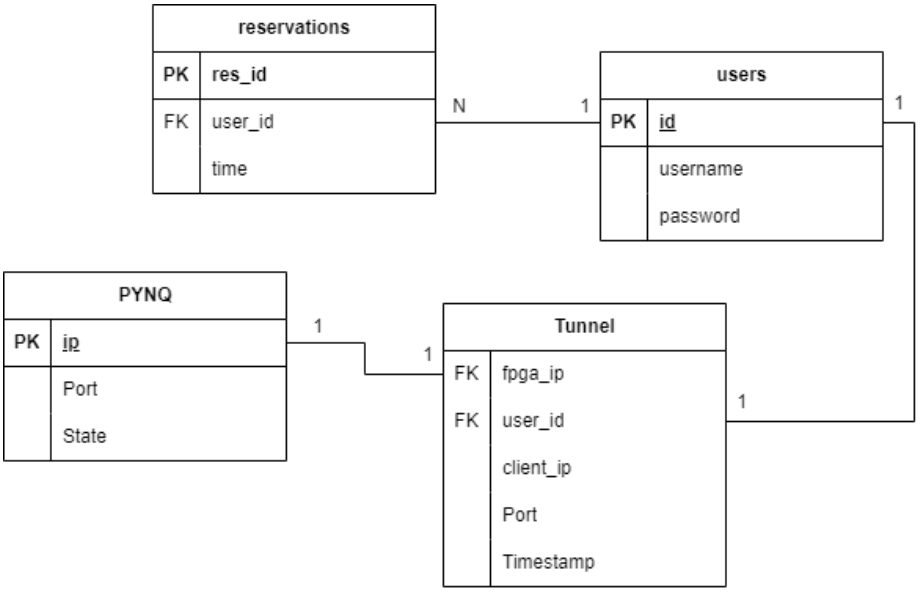
Posledným komponentom aplikácie sú zariadenia PYNQ, ktoré budú slúžiť na spúšťanie dopredu skompilovaných binárnych súborov. Binárne súbory si používateľ dopredu naprogramuje v ním preferovanom prostredí, väčšinou to je program Vivado. Binárne súbory si používateľ bude môcť nahrať na server aj predtým, ako sa aktivuje jeho rezervovaný časový úsek. Tieto súbory, ktoré používateľ nahrá na server, sa mu automaticky sprístupnia na ním rezervovanom zariadení. V rezervovanom časovom úseku bude mať používateľ prístup priamo na stránke aplikácie k Jupyter Notebooku, ktorý je hosťovaný ním rezervovaným zariadením. Vďaka tomuto Jupyter Notebooku bude môcť na zariadení spúšťať a testovať svoje binárne súbory, vytvárať alebo nahrávať nové súbory, spúšťať Jupyter Notebook dokumenty, alebo dokonca pristúpiť k terminálu daného zariadenia, odkiaľ bude môcť vykonávať rôzne príkazy podla svojej potreby.

4.4 Diagram vzťahu entít (ER)

V tejto sekcii budem popisovať diagram vzťahu entít, ktorý je zobrazený na obrázku 4.3 Táto aplikácia bude potrebovať len dve entity, ktoré budú musieť byť uložené v databáze: rezervácie a používateľov.

- Rezervácia: Obsahuje informácie o rezervácii vykonanej používateľom pre daný časový úsek, ktorý bude uložený ako atribút tejto entity. Táto entita bude mať tri atribúty. Atribút `res_id`, ktorý predstavuje identifikátor a hlavný kľúč tejto entity. Atribút `user_id` je cudzí kľúč z entity user a odkazuje na používateľa, ktorý túto rezerváciu vytvoril. Posledný atribút je `time`, ktorý predstavuje časový úsek, na ktorý bola táto rezervácia vytvorená a je uložený v podobe časovej značky.

- Používatelia: Obsahuje prihlasovacie údaje používateľov a bude pozostávať z troch atribútov. Atribút `id` bude predstavovať identifikátor používateľa v databáze a je primárnym kľúčom tejto entity, tento atribút bude tiež cudzím kľúčom v entite rezervácia. Atribút `username` je prihlasovacie meno používateľa a zároveň bude unikátnym indexom v databáze, aby sa zamedzilo používateľom s rovnakým prihlasovacím menom. Posledným atribútom bude `password`, ktorý bude predstavovať zahašované heslo používateľa.



Obr. 4.3: Diagram vzťahu entít

- PYNQ: Každé zariadenie PYNQ bude v systéme figurovať ako entita, ktorá bude uložená v pamäti démona. Tieto zariadenia si démon načíta z konfiguračného `json` súboru pri spustení. Počet možných rezervácií na každý časový úsek sa bude odvíjať od počtu dostupných PYNQ zariadení. Rezervácia nebude viazaná na žiadne konkrétne zariadenie; keď nastane používateľov rezervovaný časový úsek, aplikácia vyberie jedno z dostupných zariadení PYNQ, na ktorom aktuálne neprebieha rezervovaný časový úsek iného používateľa. Primárnym kľúčom bude IP adresa zariadenia, pod ktorou je zariadenie pripojené do súkromnej siete so serverovou aplikáciou. Atribút port bude ukladať informáciu o tom, na ktorom porte na zariadení, kde beží serverová aplikácia, je namapovaná IP adresa zariadenia PYNQ, a port, kde zariadenie PYNQ hostuje webovú stránku Jupyter Notebooku. Posledný atribút bude predstavovať stav zariadenia PYNQ.
- Tunel: Ide o relačnú entitu medzi používateľom a ním aktuálne rezervovaným zariadením PYNQ. Táto entita bude predstavovať informácie o vytvorenom tuneli z používateľovho zariadenia na zariadenie PYNQ. Táto entita bude uložená v pamäti démona. Tunel bude mať dva cudzie kľúče, a to IP adresu zariadenia PYNQ, na ktorú je vytvorený, a ID používateľa, ktorý ma dané zariadenie rezervované. Ďalej bude ukladať informáciu o IP adrese používateľovho zariadenia, z ktorého komunikuje so zariadením PYNQ. Atribút port znovu bude ukladať informáciu o tom, na ktorom porte je namapovaná IP adresa a port zariadenia PYNQ na ktorej bude dostupná hosťovaná stránka Jupyter Notebooku na zariadení, na ktorom bude bežať serverová aplikácia.

4.5 Prepojenie medzi komponentmi

Pre správne fungovanie aplikácie budú musieť komponenty medzi sebou komunikovať. Na túto komunikáciu budú využité rôzne technológie, ktoré sú dostupné v jednotlivých častiach aplikácie.

Klientske rozhranie bude komunikovať so serverovou aplikáciou pomocou volania API endpointov serverovej aplikácie, ktoré mu buď rovno odovzdajú informácie, alebo klientske rozhranie pomocou týchto endpointov pošle metódou POST dáta serverovej aplikácii.

Serverová aplikácia bude, ako už som popisoval, komunikovať s klientskym rozhraním pomocou API endpointov a taktiež bude komunikovať s démonom pomocou jeho REST API endpointu, a takisto od neho bude získavať buď informácie, alebo mu ich posielať metódou POST a vo formáte json.

Démon bude komunikovať so serverovou aplikáciou a aj s jednotlivými zariadeniami PYNQ, a to pomocou pripojenia cez SSH klienta, ktorým bude pod používateľom root spúšťať skripty na vyčistenie a pripravenie zariadenia pre ďalšieho používateľa.

Zariadenia PYNQ budú komunikovať s klientskym rozhraním pomocou vloženej Jupyter Notebook stránky, ku ktorej bude mať používateľ umožnený prístup pomocou vytvoreného tunela.

Kapitola 5

Implementácia

Táto kapitola sa venuje implementácii aplikácie, pričom detailne popisuje použité technológie a postupy, ktoré boli nevyhnutné pre vývoj aplikácie.

5.1 Architektúra

Sekcia sa venuje architektúre aplikácie, ktorá je znázornená na blokovej schéme v obrázku 5.1. Aplikácia sa skladá z troch hlavných častí: klientskej časti, serverovej časti a jednotlivých koncových zariadení PYNQ.

Klientska časť obsahuje používateľské rozhranie, ktoré je implementované v jazykoch HTML, CSS a JavaScript. Táto časť komunikuje so serverovou časťou pomocou protokolov HTTP a HTTPS, pričom reaguje na rôzne požiadavky používateľa – buď zobrazuje webové stránky aplikácie, alebo odosiela žiadosti serverovej časti na vykonanie akcií. Klientska časť tiež komunikuje so zariadeniami PYNQ prostredníctvom protokolov HTTP a HTTPS. Komunikácia prebieha cez vložený Jupyter Notebook, ktorý je prístupný vytvorením tunela z používateľovho zariadenia na zariadenie PYNQ cez webový server.

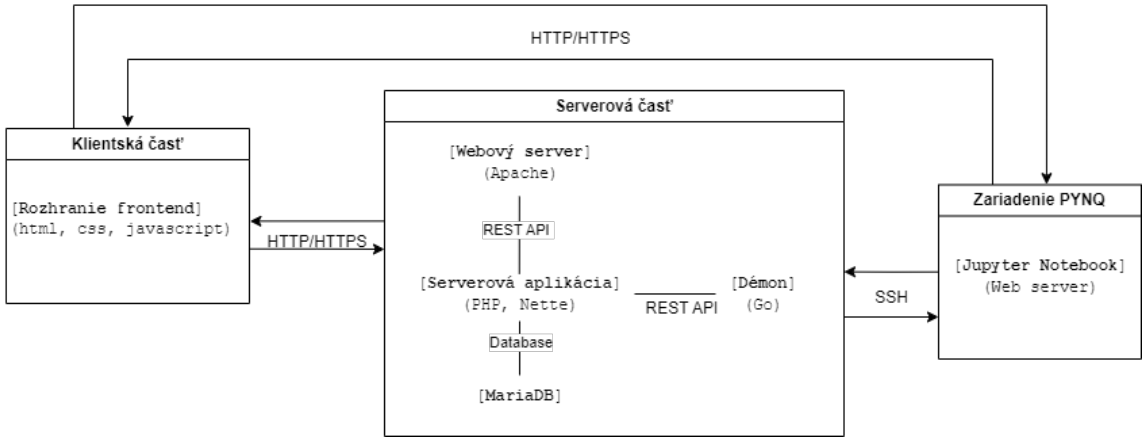
Serverová časť pozostáva z webového servera Apache, v ktorom beží aplikácia napísaná v jazyku PHP a rámca Nette, z databázy MariaDB a z démona. Táto časť aplikácie obsluhuje požiadavky pomocou REST API. Na základe požiadavky môže zobraziť webovú stránku, poslať dáta vo formáte json, alebo spracovať požiadavku na vykonanie určitej akcie. Démon bežiaci na serveri je napísaný v jazyku Go a komunikuje so serverovou aplikáciou cez REST API. Serverová aplikácia môže cez toto API poslať požiadavku na získanie dát o zariadeniach PYNQ, tuneloch, alebo na zmenu stavu zariadenia PYNQ.

Každé zariadenie PYNQ hosťuje webovú stránku s Jupyter Notebookom, ktorá umožňuje používateľovi ovládať zariadenie alebo spúšťať preložené súbory.

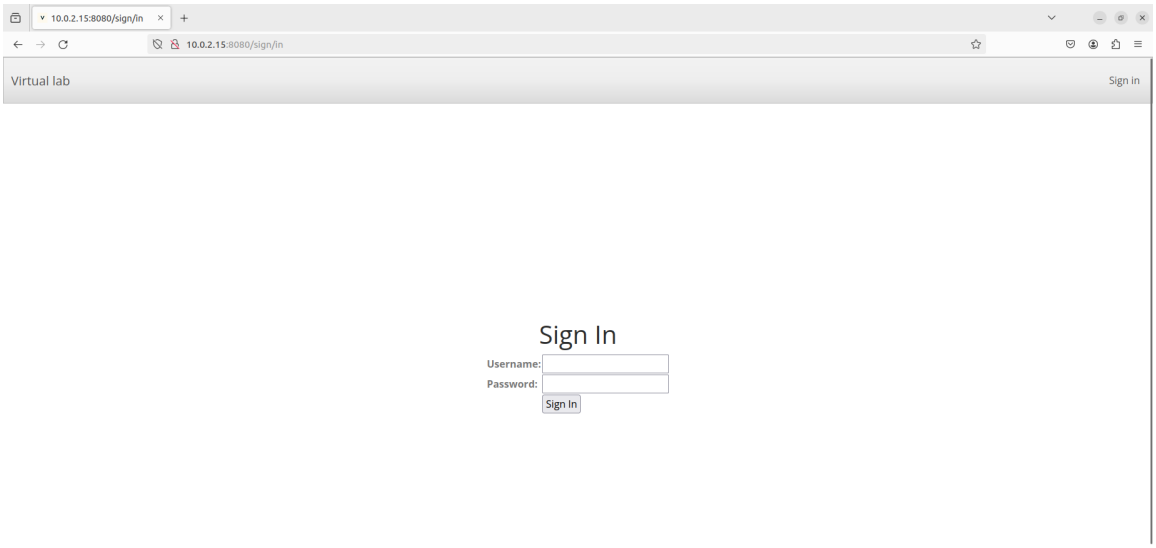
5.2 Používateľské rozhranie

Cieľom používateľského rozhrania je umožniť jednoduchú a efektívnu interakciu používateľa s aplikáciou. Rozhranie, napísané v HTML, CSS a JavaScript, je navrhnuté tak, aby bolo intuitívne a prístupné z akéhokoľvek zariadenia.

Rozhranie obsahuje navigačnú lištu, ktorá je konzistentne prítomná na vrchu každej stránky. Táto lišta umožňuje používateľom rýchlo prechádzať medzi rôznymi časťami aplikácie a poskytuje informácie o stave prihlásenia.



Obr. 5.1: Bloková schéma architektúry



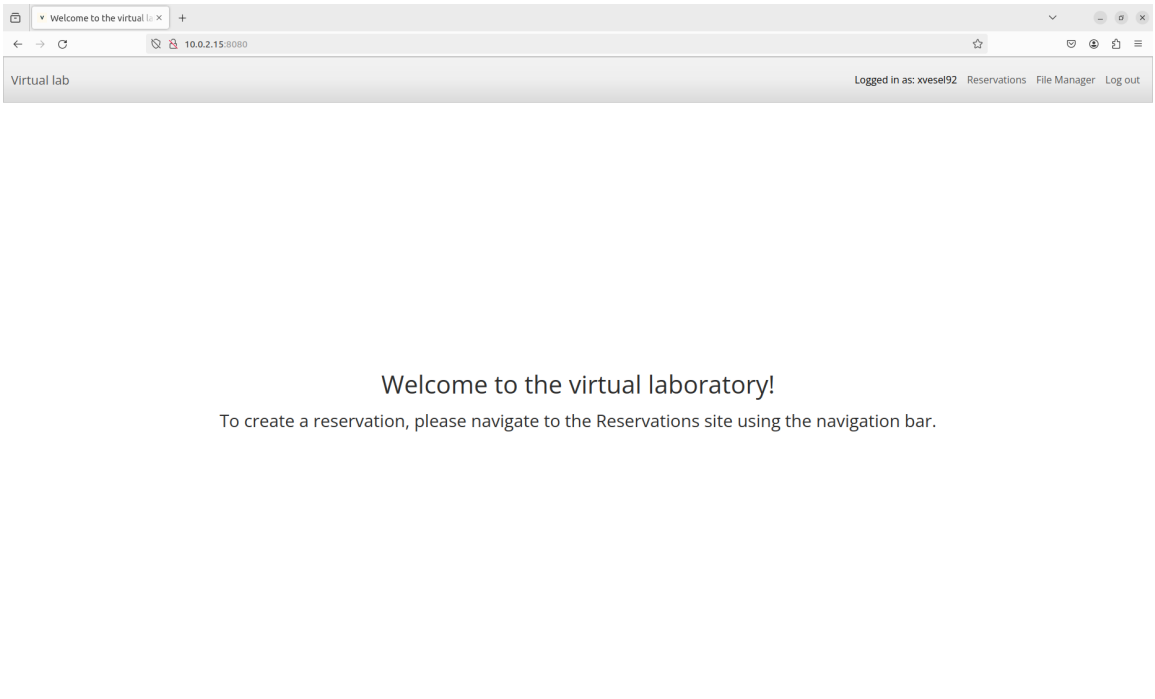
Obr. 5.2: Stránka prihlásenia

5.2.1 Prihlasovacie rozhranie

Prihlasovacie rozhranie, zobrazené na obrázku 5.2, umožňuje používateľom zadávať svoje prihlasovacie údaje prostredníctvom formulára, ktorý sa postará o komunikáciu so serverovou časťou aplikácie. Táto stránka je prvým bodom interakcie používateľa s aplikáciou, zabezpečujúc tak prístup do systému.

5.2.2 Uvítacie rozhranie

Po prihlásení sa používateľ dostáva do uvítacieho rozhrania zobrazeného na obrázku 5.3, ktoré poskytne základné informácie používateľovi a nasmeruje ho k ďalšej interakcii s aplikáciou.



Obr. 5.3: Uvítacia stránka

5.2.3 Rezervačné rozhranie

V rezervačnom rozhraní, ktoré je zobrazené na obrázku 5.4, si môžu používatelia rezervovať alebo odrezervovať časové úseky podľa aktuálnej dostupnosti. Táto stránka dynamicky zobrazuje dostupné časové úseky a ich stav na základe aktuálnej dopytovej situácie. Tak isto dynamicky zobrazuje aj počet aktuálnych rezervácií prihláseného používateľa.

5.2.4 Rozhranie na správu súborov

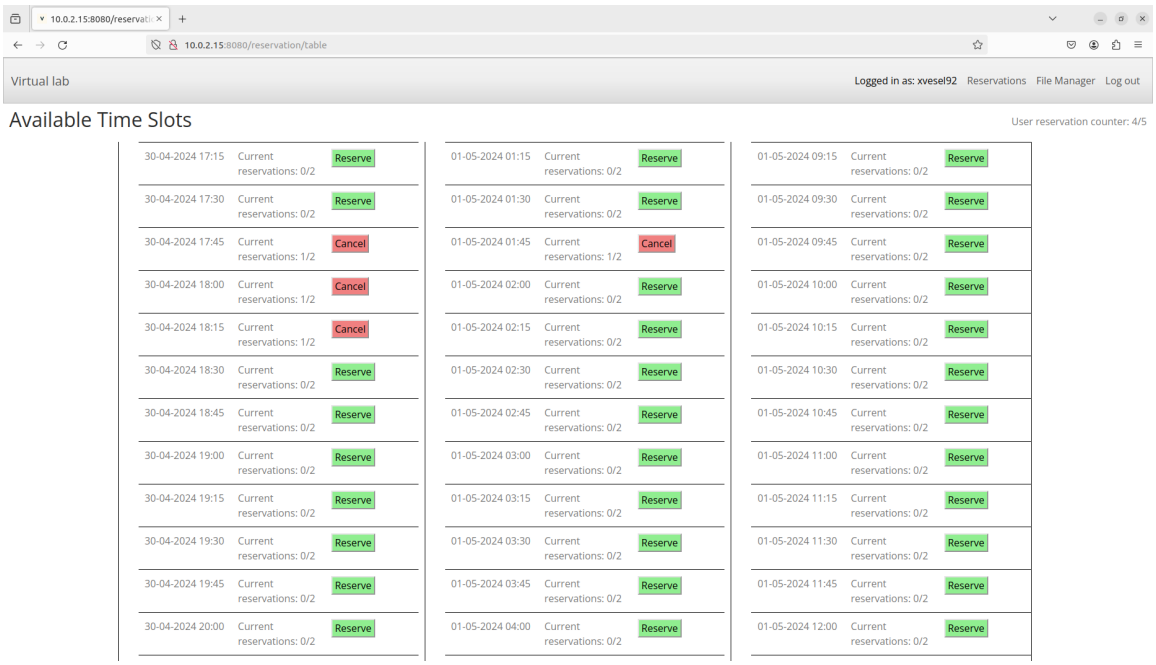
Rozhranie na správu súborov, zobrazené na obrázku 5.5, umožňuje používateľom nahrávať a spravovať súbory v ich osobnom pracovnom priečinku na serveri. Používatelia môžu súbory jednoducho sťahovať alebo odstraňovať a majú tak prístup k organizácii svojich dát v hierarchickej štruktúre aj mimo rezervovaného časového úseku.

5.2.5 Rozhranie aktívneho pripojenia

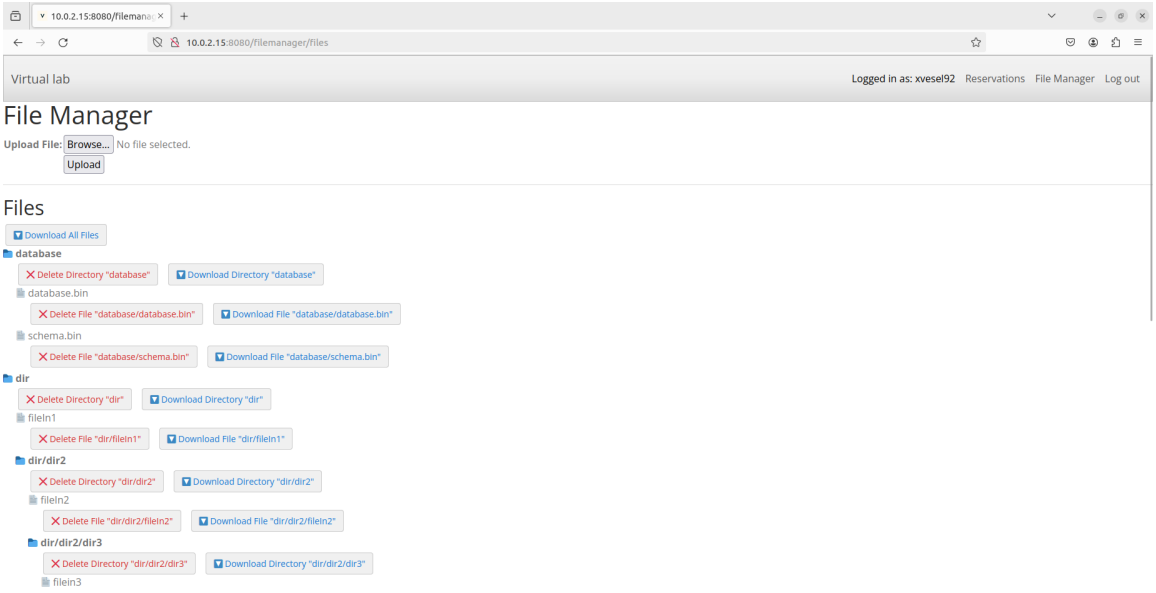
Rozhranie aktívneho pripojenia k zariadeniu, zobrazené na obrázku 5.6, umožňuje používateľom interagovať priamo so zariadením PYNQ cez integrovaný Jupyter Notebook. Táto stránka je dostupná iba vtedy, ak má používateľ aktívnu rezerváciu alebo vytvorené tunelové spojenie na jedno zo zariadení PYNQ.

5.2.6 Administračné rozhranie

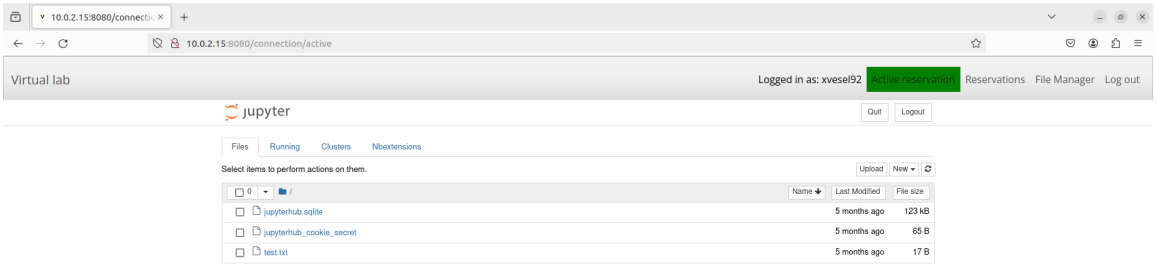
Administračné rozhranie, zobrazené na obrázku 5.7, poskytuje administrátorom nástroje na správu používateľov a jednotlivých zariadení PYNQ. Administrátori môžu pridávať nových používateľov, spravovať rezervácie časových úsekov a monitorovať a meniť stav jednotlivých zariadení PYNQ.



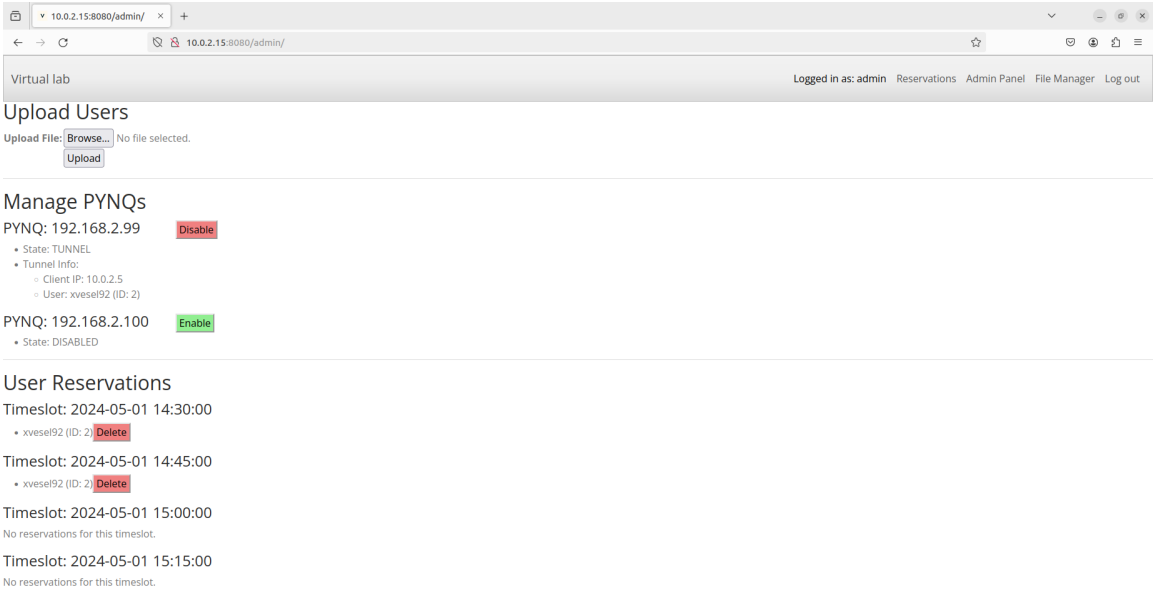
Obr. 5.4: Stránka pre rezervácie časových úsekov



Obr. 5.5: Stránka správcu súborov



Obr. 5.6: Stránka pripojenia k zariadeniu



Obr. 5.7: Stránka admin panelu

5.3 Serverová aplikácia

Serverová aplikácia zodpovedá za spracovanie požiadaviek od používateľov a poskytuje dáta používateľskému rozhraniu. Je implementovaná v jazyku PHP s využitím rámca Nette, ktorý uprednostňuje architektúru založenú na prezentéroch namiesto tradičného modelu `MVC`.

Rámec Nette je v serverovej aplikácii využitý pre svoju robustnosť a schopnosť efektívne spracovať viaceré typy požiadaviek cez REST API. Prezentéry v Nette môže obsluhovať viaceré používateľské rozhrania a poskytovať rôzne endpointy pre operácie ako sú zápis do databázy alebo autentifikáciu používateľov.

5.3.1 Moduly a služby serverovej aplikácie

Pre správne fungovanie aplikácie je potrebné, aby aplikácia mohla komunikovať s databázou, rozhraním pre správu a komunikáciu s PYNQ jednotkami, alebo mohla verifikovať a spravovať používateľov či manipulovať so súbormi na serveri, toto jej umožňujú moduly, ktoré následne používajú jednotlivé prezentéry.

ApiService a ApiFacade

Služba `ApiService` zabezpečuje komunikáciu s rozhraním pre správu a komunikáciu s jednotkami PYNQ, vrátane získavania dát o aktívnych tuneloch a zariadenia PYNQ, a to pomocou REST API. V prezentéroch je už dostupný modul `ApiFacade`, ktorý umožňuje volanie týchto funkcií.

ReservationFacade

Modul `ReservationFacade` sa používa na prácu s databázou, pomocou ktorého prezentér môže získavať, vkladať alebo mazať informácie o rezerváciách alebo používateľoch. Tento modul umožňuje efektívne spracovanie dát spojených s používateľskými rezerváciami.

Authenticator a FileManager

Modul `Authenticator` sa stará o autentifikáciu používateľov, ich verifikáciu a správu používateľských rolí. `FileManager` je použitý na manipuláciu so súbormi v používateľských pracovných priečinkoch, čo zahŕňa vypísanie, nahrávanie, sťahovanie a mazanie súborov.

5.3.2 Funkcionalita

Funkcionalita serverovej aplikácie je úzko spätá s prezentérmi, ktoré komunikujú s používateľským rozhraním, ale aj s ostatnými komponentami v serverovom rozhraní a to pomocou vyššie popísaných modulov. Preto táto sekcia popisuje funkcionalitu každého prezentéru a tým aj funkcionalitu serverovej aplikácie.

DefaultPresenter

`DefaultPresenter` v rámci serverovej aplikácie plní kľúčovú úlohu v riadení prístupu a poskytovaní dát používateľom. Hlavnými funkcionalitami prezetéra sú autentifikácia a poskytnutie mena aktuálne prihláseného používateľa do navigačnej lišty. Pri každom volaní

prezentér zabezpečuje kontrolu, či je používateľ prihlásený. Ak nie je prihlásený, automaticky ho presmeruje na prihlasovaciu stránku. Ak je používateľ prihlásený, jeho meno je poslané do šablóny, ktorá sa používa na zobrazenie navigačnej lišty, čo umožňuje aplikácii personalizovať užívateľské rozhranie. Prezentér poskytuje aj niekoľko endpointov:

- `get-live-reservation`: Zistí, či má používateľ aktuálne prebiehajúcu rezerváciu zariadenia, alebo aktívne tunelové spojenie na zariadenie PYNQ. Tento endpoint je dôležitý pre dynamické zobrazenie tlačidla na prístup do rozhrania aktívneho pripojenia.
- `buttons`: Endpoint generuje `JSON` odpoveď s dátami pre jednotlivé tlačidlá, ktoré reprezentujú časové úseky.
- `allreservations`: Poskytuje počet všetkých rezervácií pre každý časový úsek v systéme, čo umožňuje používateľom získať prehľad o obsadenosti jednotlivých časových úsekov.
- `get-fpga-count`: Vracia počet všetkých dostupných zariadení PYNQ v systéme.
- `get-user-reservation-count`: Informuje o celkovom počte rezervácií, ktoré má aktuálne prihlásený používateľ.
- `reservation`: Tento endpoint umožňuje vytvorenie alebo zrušenie rezervácie na základe poskytnutých údajov cez POST metódu. Endpoint skontroluje, či je rezervácia možná bez prekročenia kapacitných limitov a vykoná požadovanú akciu, či už pridanie alebo odstránenie rezervácie.

LandingPagePresenter

`LandingPagePresenter` v serverovej aplikácii plní jedinú úlohu, a to je vykreslenie uvítacej stránky aplikácie, ktorej poskytuje statickú správu do šablóny, ktorá je následne vykreslená.

ReservationPresenter

`ReservationPresenter` je prezentér ktorý v rámci serverovej aplikácie zohráva významnú rolu pri správe a zobrazení časových úsekov na rezerváciu. Jeho primárnou funkciou je zobraziť používateľom dostupné časové úseky na rezerváciu v prehľadnej a organizovanej forme.

Prezentér generuje časové úseky v 15-minútových intervaloch na nasledujúci deň. Tieto časové úseky distribuuje do troch stĺpcov, čím uľahčí vizuálnu orientáciu a tým pomáha používateľom rýchlo identifikovať dostupné časové úseky.

SignPresenter

`SignPresenter` je zodpovedný za spracovanie prihlásenia do aplikácie. Jeho hlavnou úlohou je zobraziť a spracovať prihlasovací formulár, čo je nevyhnutný krok na overenie identity používateľov. Vykreslený formulár je základným rozhraním, cez ktoré používatelia zadávajú svoje prihlasovacie údaje. Formulár pozostáva z dvoch hlavných polí, jedno pre prihlasovacie meno a druhé pre heslo, okrem toho obsahuje aj tlačidlo na potvrdenie údajov, ktoré spúšťa proces overenia.

Na overenie používateľa prezentér používa autentifikačný systém aplikácie, ktorý je implementovaný cez modul `Authenticator`. Autentifikačný proces zahŕňa overenie, či zadané prihlasovacie meno a heslo zodpovedajú údajom uloženým v databáze.

Po úspešnom prihlásení je používateľ presmerovaný na hlavnú stránku aplikácie, kde sú mu sprístupnené funkcie podľa jeho oprávnení. Ak ale bolo prihlásenie neúspešné, je o tom používateľ informovaný a môže sa opätovne pokúsiť zadať svoje prihlasovacie údaje.

FileManagerPresenter

`FileManagerPresenter` sa zaoberá správou súborov a priečinkov v používateľovom pracovnom priečinku na serveri. Medzi jeho hlavné funkcionality patrí nahrávanie, sťahovanie, mazanie a zobrazovanie súborov a priečinkov. Na operácie so súbormi a priečinkami používa prezentér modul `FileManager`.

ConnectionPresenter

`ConnectionPresenter` je kľúčovým prezentérom v aplikácii, ktorý zabezpečuje riedenie tunelových spojení so zariadeniami PYNQ. Tento prezentér zohráva zásadnú úlohu v udržiavaní spoľahlivosti a kontinuity komunikácie medzi používateľmi a zariadeniami.

Prezentér vykresľuje rozhranie aktívneho pripojenia kde šablóne poskytuje informáciu o porte na serveri, na ktorom je nastavené presmerovanie komunikácie na rezervované zariadenie PYNQ.

Pred vykreslením rozhrania prezentér zistí, či používateľ má aktívnu rezerváciu a či už existuje tunelové spojenie na zariadenie. Ak používateľ má rezerváciu, ale spojenie ešte nie je vytvorené, prezentér iniciuje vytvorenie tunela. Ak používateľ nemá aktívnu rezerváciu a ani spojenie, je presmerovaný na hlavnú stránku s informáciou, že nemá žiadnu rezerváciu.

Prezentér monitoruje stav tunelových spojení a aktívnych rezervácií. Ak zistí pri vytváraní nového tunelového spojenia pre používateľa s rezerváciou, že existujúce spojenie už nie je spojené s aktívnou rezerváciou (t.j. rezervácia vypršala alebo bola zrušená) a nie je voľné žiadne iné zariadenie PYNQ, spojenie zruší, čím uvoľní zariadenie pre nového používateľa.

AdminPresenter

`AdminPresenter` je zameraný na administratívne úlohy, správu zariadení a používateľov. Jeho hlavnou úlohou je poskytovať administračné rozhranie a spravovať rôzne operácie, ktoré sú kľúčové pre správu systému.

Prezentér pred vykreslením administračného rozhrania skontroluje, či aktuálne prihlásený používateľ má administrátorské práva. Ak nie, je presmerovaný na domovskú stránku so správou o jeho nedostatočných právach, čím sa zabezpečí, že administratívne funkcie sú prístupné len oprávneným používateľom.

Podobne ako `ReservationPresenter`, aj `AdminPresenter` generuje časové úseky a získava informácie o rezerváciách pre tieto úseky. Tieto informácie sú kľúčové pre monitorovanie a spravovanie rezervácií jednotlivých používateľov.

`AdminPresenter` umožňuje adminovi nahrávať nových používateľov prostredníctvom `csv` súboru. Nette formulár skontroluje formát súboru a po úspešnom nahratí je každému používateľovi vygenerované náhodné heslo, a používateľ je pridaný do systému.

Prezentér poskytuje funkcie na aktiváciu a deaktiváciu zariadení PYNQ. Tieto funkcie umožňujú adminovi spravovať stav zariadení v závislosti od ich potrieb alebo technických

problémov. Pri deaktivácii zariadenia prezentér kontroluje a ruší všetky aktívne rezervácie a tunelové spojenia, aby zabezpečil konzistenciu dát a správne fungovanie systému.

5.4 Štruktúra databázy

Štruktúra implementovanej databázy je zobrazená na obrázku 5.8. Ostatné entity z ER diagramu zobrazeného na obrázku 4.3 v kapitole Návrh sú uložené v pamäti démona, kde sú bližšie popísané.

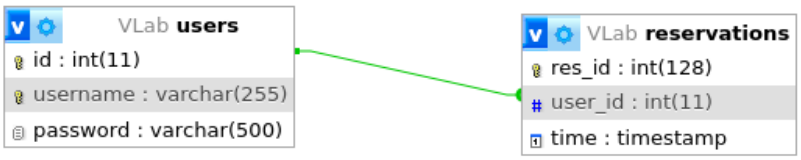
5.4.1 Popis tabuliek

Tabuľka `VLab users` uchováva informácie o používateľoch systému `VLab` a zahŕňa nasledujúce polia:

- `id`: Primárny kľúč typu celého čísla (`int(11)`), ktorý jednoznačne identifikuje každého používateľa. Tento atribút je nastavený tak, aby automaticky narastal, čo znamená, že sa automaticky generuje jedinečná hodnota pre každý nový záznam.
- `username`: Pole reťazca (`varchar(255)`), ktoré obsahuje používateľské meno používateľa `VLab`. Dĺžka 255 znakov umožňuje široké spektrum používateľských mien. Používateľ sa týmto používateľským menom prihlasuje do systému a tento atribút je v databáze nastavený, aby bol unikátnym indexom, čo znamená, že v danej tabuľke nemôže byť viacej záznamov s rovnakým používateľským menom.
- `password`: Pole reťazca (`varchar(500)`), ukladá heslo pre každého používateľa. Limit 500 znakov je kvôli tomu, že heslá sú uložené v hašovanej forme, čo je nevyhnutné pre bezpečnosť.

Tabuľka `VLab reservations` spravuje rezervácie vytvorené používateľmi v rámci aplikácie a zahŕňa nasledujúce polia:

- `res_id`: Primárny kľúč typu celého čísla (`int(11)`), ktorý jednoznačne identifikuje každú rezerváciu. Tento atribút je nastavený tak, aby automaticky narastal, čo znamená, že sa automaticky generuje jedinečná hodnota pre každý nový záznam.



Obr. 5.8: Štruktúra databázy

- `user_id`: Cudzí kľúč (`int(11)`) spájajúci každú rezerváciu s používateľom z tabuľky `VLab users`. Týmto je vytvorený vzťah označujúci, ktorému používateľovi patria jednotlivé rezervácie.
- `time`: Pole časovej pečiatky (`timestamp`), ktorá značí začiatok časového úseku, pre ktorý je rezervácia vytvorená.

5.4.2 Systém správy databáz

Databáza využíva `MariaDB`, populárny open-sourcový systém na správu relačných databáz. Je to `fork MySQL` a je známy pre svoj výkon, spoľahlivosť a jednoduché použitie. `MariaDB` podporuje pokročilé funkcie jazyka `SQL` a je plne kompatibilná s `MySQL`, čo znamená, že dokáže spúšťať aj aplikácie napísané pre `MySQL`, a to bez úprav.

Systém `MariaDB` na správu databázy je dobrou voľbou pre webové aplikácie vďaka svojmu robustnému výkonu. Jeho škálovateľnosť a pokročilé možnosti klastrovania sú predurčené pre aplikácie, ktoré očakávajú veľké zaťaženie alebo vyžadujú vysokú dostupnosť.

5.4.3 Implementácia vzťahov

Vzťah medzi `VLab users` a `VLab reservations` je implementovaný prostredníctvom cudzieho kľúča (`user_id`) v tabuľke `VLab reservations`, ktorý odkazuje na pole `id` v tabuľke `VLab users`. Ide o vzťah typu jedného k viacerým (`one-to-many`), kde každý používateľ môže mať viacero rezervácií, ale každá rezervácia je spojená iba s jedným používateľom.

`MariaDB` vynucuje referenčnú integritu pomocou obmedzení cudzích kľúčov. To zaisťuje, že všetky hodnoty `user_id` v tabuľke `VLab reservations` zodpovedajú existujúcim hodnotám `id` v tabuľke `VLab users`, čím sa zabráni osamoteným záznamom a zachová sa konzistencia údajov.

5.5 Rozhranie pre správu a komunikáciu s PYNQ jednotkami

Démon je kľúčovou súčasťou systému, zodpovednou za správu komunikácie medzi serverovou aplikáciou a koncovými zariadeniami PYNQ, vytvára teda medzivrstvu medzi serverovou aplikáciou a nastavením hosťovského a serverového systému. Hlavné funkcie démona zahŕňajú:

- Získavanie a poskytovanie informácií o zariadeniach PYNQ a vytvorených tuneloch. Démon tieto informácie ukladá v pamäti a sú dostupné prostredníctvom REST API endpointov.
- Komunikácia s PYNQ zariadeniami na ich prípravu pre používateľov, čo zahŕňa montovanie používateľských pracovných priečinkov a konfiguráciu tunelových spojení.

Démon komunikuje so serverovou aplikáciou pomocou `REST API endpointov` a s PYNQ zariadeniami pomocou `SSH klienta`.

Na implementáciu démona bol z dôvodu týchto požiadaviek vybraný programovací jazyk Go. Démon potrebuje mať napríklad prístup k sieťovým rozhraniam, čo mu jazyk Go umožňuje. Toto rozhodnutie bolo tiež motivované z dôvodu jednoduchosti jazyka, robustnej štandardnej knižnice a rozsiahlej podpory vytvárania REST API serverov a manipulácie s IP tabuľkami.

5.5.1 Štruktúra démona

Démon je navrhnutý ako monolitická aplikácia v jazyku Go, pozostávajúca z jedného balíka `main`, ktorý obsahuje všetky kľúčové súbory potrebné na jeho fungovanie. Jednotlivé súbory sú špecificky určené na rôzne úlohy a funkcionalitu démona:

- `main.go`: Tento súbor je vstupným bodom programu. Inicializuje a spustí HTTP server a zabezpečuje správne načítanie základnej konfigurácie.
- `apilib.go`: Definuje štruktúry dát, ktoré démon používa. Tieto štruktúry sú kľúčové pre ukladanie dát o zariadeniach PYNQ, tunelových spojeniach a konfiguračných parametroch.
- `boot.go`: Obsahuje funkcie, ktoré sa automaticky spustia pri štarte démona. Tieto funkcie zahŕňajú nastavenie IP tabuliek a načítanie dát o zariadeniach PYNQ.
- `get.go`: Obsahuje implementáciu funkcií, ktoré reagujú na HTTP GET požiadavky od klientov. Tieto funkcie sú zavolané cez definované REST API endpointy a slúžia na získavanie dát o zariadeniach PYNQ a tunelových spojeniach.
- `post.go`: Podobne ako `get.go`, ale tento súbor obsahuje funkcie reagujúce na HTTP POST požiadavky. Tieto funkcie vytvárajú a mažú tunelové spojenia a aktualizujú stav zariadení PYNQ.
- `sshClient.go`: Zahrňuje implementáciu SSH klienta, ktorý je použitý na komunikáciu so zariadeniami PYNQ.

5.5.2 Hlavné rozhranie démona

Súbor `main.go` predstavuje vstupný bod programu, ktorý koordinuje jeho operácie. Tento súbor importuje nevyhnutné balíky a knižnice jazyka Go, deklaruje globálne premenné, a zodpovedá za inicializáciu komponentov a nastavenie HTTP servera. Jeho funkcionalita je rozdelená medzi inicializáciu a konfiguráciu servera.

Na začiatku súboru sú importované základné balíčky jazyka Go a knižnice pre vytvorenie HTTP servera, prácu s IP tabuľkami a SSH klientom. Súbor následne definuje globálne premenné, ktoré ukladajú informácie o jednotlivých zariadeniach PYNQ, tunelových spojeniach, konfigurácii a SSH klientovi.

Pri spustení hlavnej funkcie programu sa najprv inicializuje `iptables` klient a overí sa jeho funkčnosť. Následne sa načítajú konfiguračné dáta zo súboru, správne nastaví hosťovské prostredie a inicializuje HTTP server s definovanými cestami pre REST API endpointy. Tento server je nakoniec spustený na porte 20000 a počúva prichádzajúce žiadosti, ktoré spracováva podľa priradených definovaných obsluhujúcich funkcií. Na tento server sa nepripojuje klientska časť priamo, ale vždy pristupuje cez serverovú časť webového prostredia, čo zjednodušuje požiadavky na démona.

5.5.3 Dátové štruktúry

Súbor `apilib.go` definuje základné dátové štruktúry, ktoré slúžia na ukladanie dát o zariadeniach PYNQ, tunelových spojeniach, inštrukcií používateľov a komunikáciu v rámci aplikácie. Tieto štruktúry sú serializované do formátu JSON pre sieťový prenos a sú základom pre aplikačné programové rozhranie (API). Definovaných je šesť dátových štruktúr:

- `FPGA`: Ukladá informácie o jednotlivých zariadeniach PYNQ. Zaznamenáva podstatné informácie ako IP adresu zariadenia, port, ktorý špecifikuje komunikačný kanál a stav zariadenia.
- `Tunnel`: Reprezentuje informácie o tunelových spojeniach medzi klientom a PYNQ zariadeniami. Uchováva údaje ako IP adresy zariadenia a klienta, ID používateľa a port používaný pre tunel.
- `Instruction`: Definuje formát príkazov posielaných cez API na ovládanie tunelových spojení. Umožňuje špecifikovať akciu (vytvorenie alebo zrušenie), cieľové zariadenie a používateľa.
- `State`: Slúži na zmenu stavu PYNQ zariadenia cez API. Umožňuje dynamicky meniť stav zariadení. Zahrňuje IP adresu zariadenia a požadovaný stav.
- `ErrorInternal`: Táto štruktúra je používaná pre interné chybové hlásenia v systéme.
- `Commons`: Obsahuje konfiguračné nastavenia, ktoré sú globálne pre celú aplikáciu. Zahrňuje nastavenia sieťových rozhraní a štartovací port pre maškarádu, ktoré sú potrebné pre správne fungovanie démona a jeho schopnosť komunikovať s externými zariadeniami.

Dáta v týchto štruktúrach sú ukladané priamo v RAM pamäti démona s tým, že dáta o PYNQ zariadeniach sú načítané z konfiguračného JSON súboru.

5.5.4 Konfiguračný súbor

Máme konfiguračný JSON súbor, ktorý obsahuje dáta o jednotlivých zariadeniach PYNQ, tento súbor je načítaný pri spustení aplikácie a dáta z neho sú serializované do štruktúry `FPGA` a uložené v pamäti RAM démona.

5.5.5 Práca so sieťovým rozhraním a tunelmi

Démon spravuje a riadi sieťovú komunikáciu medzi klientmi a zariadeniami PYNQ. Hlavné funkcie zabezpečujú, že komunikácia je efektívne presmerovaná a bezpečne spravovaná.

Počiatočné nastavenie hosťovského prostredia

Démon obsahuje funkcie na nastavenie a správu sieťových pravidiel pomocou IP tabuliek, ktoré sú potrebné na riadenie prichádzajúcej a odchádzajúcej sieťovej komunikácie s PYNQ zariadeniami. Funkcia `masqueradeCreate()` nastavuje pravidlá pre maškarádu, čo umožňuje, aby odchádzajúca komunikácia zo zariadení PYNQ vyzerala, akoby prichádzala priamo zo servera, na ktorom je démon spustený. Táto funkcia taktiež inicializuje pravidlá na presmerovanie portov, čo umožňuje presnú distribúciu sieťovej komunikácie na špecifické zariadenia pomocou komunikačných kanálov.

Funkcia `tablesFlush()` zase zabezpečuje, že pri spustení aplikácie sa všetky predchádzajúce nastavenia IP tabuliek vymažú. Týmto sa predchádza akumulácii zastaraných alebo konfliktných pravidiel, ktoré by mohli ovplyvniť výkon alebo bezpečnosť aplikácie.

Práca s tunelmi

Tunely sú vytvárané a spravované pomocou dvoch kľúčových funkcií: `instructionCreate()` a `instructionDelete()`, ktoré sú volané podľa typu prijatej inštrukcie cez endpoint. Tieto funkcie riadia vytváranie a rušenie tunelov, ktoré umožňujú pripojenie medzi klientmi a PYNQ zariadeniami.

Funkcia `instructionCreate()` spracúva žiadosti o vytvorenie tunela. To zahŕňa nastavenie príslušných pravidiel v IP tabuľkách, vytvorenie komunikačného kanálu na porte na serveri pre konkrétneho klienta, a inicializáciu SSH spojenia pre správu zariadenia PYNQ. Vytvorené tunely sú potom uložené v globálne dostupnom zozname, čo umožňuje ich ďalšie spracovanie a poskytovanie dát tunelov pre serverovú aplikáciu.

Na druhej strane, funkcia `instructionDelete()` sa zaoberá rušením tunelov. Tento proces zahŕňa odstránenie pravidiel IP tabuliek, inicializáciu SSH spojenia a vyčistenie zariadenia PYNQ. Výsledkom je, že zariadenie je pripravené na ďalšie použitie bez dát predošlého používateľa, alebo zanechaných tunelových spojení ostatných používateľov.

Pri vytváraní alebo rušení tunelového spojenia je potrebné, aby sa démon pripojil k zariadeniu PYNQ a toto zariadenie pripravil pre používateľa. Na akcie, ktoré je potrebné pri vytváraní alebo rušení tunelového spojenia vykonať, sú vytvorené skripty, ktoré tieto akcie vykonajú a démon ich len pomocou spojenia spustí. Toto spojenie medzi démonom a zariadením sa vytvorí cez SSH kanál, ktorý nie je pre používateľov normálne dostupný. Autentifikácia prebieha pomocou RSA kľúča, ktorý je uložený jedine na hosťovskom serveri. Pri eskalácii práv na PYNQ zariadeniach tak nehrozí to, že by niekto dostal prístup ku všetkým zariadeniam.

5.6 Systémové požiadavky

Táto sekcia popisuje nevyhnutné systémové požiadavky na spustenie a správne fungovanie aplikácie. Aplikácia beží na operačnom systéme `Ubuntu 22.04` so systémom správy databáz `MariaDB`. Webová stránka beží pomocou aplikácie `Apache` a jazykom PHP, pre ktorý je potrebné rozšírenie `zip`. Démon beží pomocou jazyka `Go`, ktorý má svoje závislosti uložené pomocou balíčkov. Webová aplikácia má závislosti spravované pomocou nástroja `composer`, ktorý ich nainštaluje. Aplikácia potrebuje mať zapnutý a nastavený `nfs` server.

Pred spustením aplikácie je potrebné, aby server mal vygenerované kľúče pre SSH komunikáciu a admin nastavil cestu k týmto kľúčom v súbore `main.go` v démonovi a tak isto nastavil cestu k priečinku, v ktorom budú uložené pracovné priečinky používateľov. Túto cestu je potrebné nastaviť jednak v súbore `main.go` v démonovi, jednak v module `FileManager.php` vo funkcii `buildDir()`. Je potrebné túto cestu k priečinku povoliť pre `nfs` server pre sieť, na ktorej sú pripojené zariadenia PYNQ.

Pre správne fungovanie aplikácie je potrebné, aby na serveri a zariadeniach PYNQ bol vytvorený používateľ `student` s takým istým `UID` na oboch zariadeniach. A je potrebné, aby bola pod týmto používateľom spustená serverová aplikácia.

5.7 PYNQ nastavenia

Zariadenie PYNQ je potrebné nastaviť tak, aby správne fungovalo s aplikáciou. Je potrebné, aby každé zariadenie PYNQ malo pridaný SSH kľúč servera, na ktorom beží aplikácia a malo povolené, aby server pod týmto kľúčom mohol byť prihlásený bez zadávania hesla ako používateľ `root`, ktorý potom spustí skripty.

Je potrebné, aby skripty (`mount.sh` a `unmount.sh`), ktoré sa démon pomocou `SSH` spojenia pokúsi, spustiť boli v domovskom priečinku používateľa `root`.

V konfiguračnom súbore `jupyter_notebook_config.py`, ktorý sa nachádza v domovskom priečinku používateľa `root`, je potrebné pridať niekoľko nastavení pre správne fungovanie s aplikáciou.

- Nastavenie nootebookového priečinku na priečinok `/home/student`, na tento priečinok sa budú pripájať pracovné priečinky zo servera.
- Nastavenie terminálu v Jupyter Notebooku tak, aby pri vytvorení nového terminálu bol prihlásený používateľ `student`.
- Nastavenie hesla do Jupyter Notebooku, pomocou ktorého sa prípadní používatelia budú môcť prihlásiť.
- Nastavenie čísla portu, na ktorom Jupyter Notebook beží.
- Nastavenie cookies `Tornado` web serveru tak, aby stránka Jupyter Notebooku povoľovala `cross-site scripting (XSS)` a tým umožnila byť vložená do stránok webovej aplikácie.

Kapitola 6

Vyhodnotenie vlastností systému na vzorke užívateľov

Táto kapitola sa zameriava na vyhodnotenie kľúčových vlastností systému na vzorke reálnych užívateľov. Pochopenie toho, ako koncoví používatelia interagujú so systémom a aké majú z neho pocity, je dôležité pre ďalšie smerovanie a vývoj produktu. Cieľom tohto hodnotenia nie je len overiť funkčnosť a bezpečnosť systému, ale aj zároveň poskytnúť pohľad na použiteľnosť a používateľské pohodlie, ktoré systém prináša.

Na začiatku kapitoly je definovaný spôsob, akým bol systém testovaný, ako aj kritériá, podľa ktorých boli vybraní účastníci testovania. Ďalšie sekcie sa venujú rozboru spätnej väzby účastníkov, rozboru výsledkov a ich implikáciám pre budúce verzie systému. V závere kapitoly sumarizujem kľúčové zistenia a formulujem odporúčania pre ďalší vývoj a zlepšenie systému.

Cieľom tejto kapitoly je poskytnúť komplexný prehľad o tom, ako je systém vnímaný jeho koncovými užívateľmi a ako je na tom systém z bezpečnostnej a funkčnej stránky.

6.1 Ciele testovania

Sekcia si kladie za cieľ detailne definovať a vymedziť hlavné zámery a očakávania spojené s testovaním systému na vybranej vzorke používateľov. Tieto ciele nám pomôžu lepšie pochopiť, ako systém slúži koncovým používateľom, a sú základom pre interpretáciu získanej spätnej väzby. Hlavné ciele testovania sú:

- `Použiteľnosť`: Otestovať, ako intuitívne a jednoducho môžu používatelia interagovať so systémom pri vykonávaní bežných úloh. Zameriam sa na to, ako rýchlo a efektívne používatelia dosahujú svoje ciele, a identifikujem prípadné bariéry, ktoré im v tom bránia.
- `Spokojnosť používateľov`: Zistiť, do akej miery sú používatelia spokojní s celkovou funkčnosťou a klientskym rozhraním aplikácie.
- `Bezpečnosť`: Ohodnotiť, ako efektívne aplikácia chráni používateľské dáta a zabezpečuje prevádzku proti potenciálnym bezpečnostným hrozbám.
- `Adaptabilita`: Posúdiť, ako ľahko je možné systém prispôsobiť zmenám v požiadavkách na rôzne potreby používateľov.

Výsledky získané v rámci týchto cieľov umožňujú identifikovať silné a slabé stránky systému, čo je nevyhnutné pre jeho ďalší vývoj, zdokonalenie a vyhodnotenie bezpečnosti systému.

6.2 Postupy použité na testovanie systému

Vybraná vzorka používateľov zahŕňa troch aktuálnych študentov predmetu IVH a piatich úspešných absolventov tohto predmetu. Tým je zaručené, že používatelia majú dostatočné technické zručnosti na ohodnotenie implementovaného systému. Používatelia systém testovali pod dohľadom pre bližšie pochopenie pracovného toku používateľa, ktorý používa systém. S používateľmi bol následne po otestovaní vykonaný rozhovor, v ktorom som zisťoval odpovede na jednotlivé ciele testovania.

6.3 Výsledky testovania

Na základe pozorovania používateľov pri práci so systémom a následnými rozhovormi s používateľmi bolo zistené, že väčšina cieľov testovania bola pozitívne naplnená. Každý používateľ si pochvaľoval interaktívne používateľské rozhranie a funkcie systému, ktoré mu umožňujú správu súborov aj mimo rezervovaného časového úseku, alebo jednoduchý a prehľadný systém na rezerváciu časových úsekov. Používatelia sa snažili použiť `SQL injection` na narušenie systému pri prihlasovaní, alebo pomocou Jupyter Notebooku narušiť zariadenie PYNQ, pomocou ktorého chceli rozbiť celý systém.

Vrámci spätnej väzby používatelia požadovali niekoľko zmien na vylepšenie práce s aplikáciou. Požadovali pridanie počítadla času na navigačnú lištu, ktoré by ukazovalo zostávajúci čas do najbližšej rezervácie, alebo čas do konca aktuálnej rezervácie. Ďalej navrhli pridať potvrdzovací formulár pri mazaní neprázdnych priečinkov v rozhraní na správu súborov. A následne aj možnosť si zaregistrovať aktuálne prebiehajúci časový úsek, ak je v ňom voľné zariadenie PYNQ. Tieto požiadavky som už do aplikácie nestihol implementovať.

6.4 Diskusia o výsledkoch

V kontexte cieľov testovania bola použiteľnosť a adaptabilita naplnená. Systém možno na základe spätnej väzby veľmi jednoducho a intuitívne používať. Systém je možné na základe požiadaviek používateľov zo spätej väzby vďaka rámcu Nette a JavaScriptu jednoducho rozšíriť o požadované požiadavky v ďalších verziách.

Systém nebolo možné narušiť pomocou `SQL injection` vo formulári na prihlásenie, lebo o komunikáciu s databázou sa stará rámec Nette, ktorý správne pracuje so zaslanými dátami vo formulári, takže pri komunikácii s databázou nedôjde k `SQL injection`.

Používateľovi sa podarilo narušiť zariadenie PYNQ pomocou Jupyter Notebooku, ktorý beží na zariadení PYNQ pod eskalovanými právami, takže každý Jupyter dokument, ktorý je ním spustený, sa vykonáva s eskalovanými právami. Toto používateľovi umožnilo narušiť súborový systém zariadenia a tým vyradiť zariadenie z prevádzky, ale k vyradeniu viacerých zariadení pomocou tohto zariadenia sa nedostal, vďaka bezpečnostným opatreniam. Tento problém je možné riešiť logovaním každej akcie používateľa na zariadení a následným prípadným zamedzením prístupu do systému, čo je možné, keďže systém bude používaný na akademickej pôde, kde používateľmi sú študenti daného predmetu.

6.5 Zhrnutie výsledkov

V rámci testovania systému boli získané pozitívne ohlasy od používateľov, ktorí si cenili interaktívne užívateľské rozhranie a pokročilé funkcie systému. Používatelia vyskúšali rôzne bezpečnostné testy, vrátane pokusov o `SQL injection`, no systém sa ukázal ako odolný vďaka použitiu rámca Nette, ktorý efektívne zamedzil týmto hrozbám. Avšak, bol identifikovaný problém s eskalovanými právami v Jupyter Notebooku na zariadeniach PYNQ, ktoré umožnili narušenie súborového systému.

Spätná väzba od užívateľov poukázala na niekoľko oblastí na zlepšenie, vrátane pridania časového počítadla, potvrdzovacieho formulára na mazanie neprázdnych priečinkov a možnosti rezervácie prebiehajúceho časového úseku, čo zvýši užívateľský komfort pri práci s aplikáciou. Systém sa javí ako dobre adaptabilný a ľahko rozšíriteľný na základe užívateľských požiadaviek, čo predstavuje silnú východiskovú pozíciu pre ďalšie vývojové cykly. Celkovo, hoci boli identifikované určité rizikové aspekty, výsledky testovania potvrdili, že systém má solídny základ pre akademické využitie a ponúka robustné riešenie pre použitie vo výučbe predmetu IVH.

Kapitola 7

Záver

Cieľom práce bolo vytvoriť aplikáciu virtuálneho laboratória, ktorá umožňuje viacerým používateľom pristupovať k zdieľaným zariadeniam PYNQ, pomocou tunelových spojení na jednotlivé zariadenia. Pre splnenie tohto cieľa bolo potrebné použiť niekoľko rôznych technológií a postupov. Tento cieľ bol splnený.

Teoretické vedomosti ohľadom možností zdieľania prostriedkov viacerým používateľom a vedomosti o doskách Zynq s nadstavbou PYNQ som naštudoval a opísal v kapitolách 2 a 3. Návrh aplikácie je popísaný v kapitole 4 a implementácia aplikácie v kapitole 5. A napokon, vyhodnotenie vlastností implementovaného systému na vzorke používateľov je opísané v kapitole 6.

Aplikácia dokáže obsluhovať požiadavky používateľov, vytvárať rezervácie pre časové úseky, pripravovať zariadenia PYNQ pre používateľov, spravovať používateľské súbory uložené na serveri, vytvárať tunelové spojenia na jednotlivé koncové zariadenia PYNQ, spravovať dané zariadenia PYNQ a rezervácie používateľov pomocou administračného rozhrania.

Hlavným prínosom tejto práce je zjednodušenie prístupu k zariadeniam PYNQ pomocou webovej aplikácie na výučbu predmetu IVH.

V rámci tejto práce som sa stretol s niekoľkými obmedzeniami, ktoré môžu systém ovplyvniť. Po prvé, príprava jedného zariadenia PYNQ pre ďalšieho používateľa trvá približne 25 sekúnd kvôli nutnosti reštartovania aplikácie Jupyter Notebook, čo môže negatívne ovplyvniť pracovný tok používateľa. Pravda, systém bol testovaný len v kontrolovanom prostredí, čo môže skresliť výsledky v porovnaní s jeho použitím v reálnej prevádzke.

Ako rozšírenie práce by bolo možné implementovať jednotlivé požiadavky používateľov získané pomocou spätnej väzby. Aplikáciu by bolo možné rozšíriť pridaním funkcie pre automatické rozpoznávanie a konfiguráciu nových PYNQ zariadení na sieťovom rozhraní, čo by zjednodušilo prácu pri rozširovaní počtu dostupných zariadení v prípadoch, keď dopyt po aplikácii náhle vzrastie.

Z bezpečnostného hľadiska by k aplikácii bolo možné vytvoriť démona, ktorý by bežal na jednotlivých zariadenia PYNQ a sledoval aktivitu používateľov na zariadeniach. Ak by došlo k podozrivej aktivite, napríklad ku vypnutiu tohto démona, alebo k narušeniu súborového systému, nahlásil by túto aktivitu aministrátorovi, a to už buď pomocou rozhrania na serveri, alebo emailovou komunikáciou.

Literatúra

[1] Bjerge, K.; Schougaard, J. H. a Larsen, D. E. A scalable and efficient convolutional neural network accelerator using HLS for a system-on-chip design. *Microprocessors and microsystems*. Kidlington: Elsevier B.V, 2021, zv. 87, s. 104363. ISSN 0141-9331.

[2] Borghesi, A.; Collina, F.; Lombardi, M.; Milano, M. a Benini, L. Power Capping in High Performance Computing Systems. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Cham: Springer International Publishing, 2015, sv. 9255, s. 524–540. Lecture Notes in Computer Science. ISBN 9783319232188.

[3] Devices, A. M. *PYNQ: Python productivity for Adaptive Computing platforms* online. 2022. Dostupné z: https://pynq.readthedocs.io/en/v3.0.0/. [cit. 2024-4-22].

[4] Devices, A. M. *Zynq 7000 SoC Technical Reference Manual* [online]. Rev. 2024. Dostupné z: https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM/Zynq-7000-SoC-Technical-Reference-Manual. [cit. 2024-5-7].

[5] Han, L. Resource Scheduling Algorithm for Heterogeneous High-Performance Clusters. In: *Intelligent Networked Things*. Singapore: Springer Nature Singapore, s. 307–321. Communications in Computer and Information Science. ISBN 9811989141.

[6] Horn, D. *What is Zynq* [online]. Rev. 2023. Dostupné z: https://digilent.com/blog/what-is-zynq/. [cit. 2024-4-22].

[7] Li, H.; Wan, B.; Fang, Y.; Li, Q.; Liu, J. K. et al. An FPGA implementation of Bayesian inference with spiking neural networks. *Frontiers in neuroscience*. Switzerland: Frontiers Research Foundation, 2024, zv. 17, s. 1291051–1291051. ISSN 1662-4548.

[8] Mohsen, A. E.-R.; GadAlrab, M. Y.; Mahmoud, Z. elhaya; Alshaer, G.; Asy, M. et al. Remote FPGA lab for ZYNQ and virtex-7 kits. In: IEEE. *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2019, s. 185–188.

[9] Monzo, C.; Cobo, G.; Morán, J. A.; Santamaría, E. a García Solórzano, D. Remote laboratory for online engineering education: The rlab-uoc-fpga case study. *Electronics*. MDPI, 2021, zv. 10, č. 9, s. 1072.

[10] Reichenbach, M.; Schmidt, M.; Pfundt, B. a Fey, D. A new virtual hardware laboratory for remote FPGA experiments on real hardware. In: Citeseer. *Proceedings*

of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE). 2011, s. 1.

[11] Sherwani, J.; Ali, N.; Lotia, N.; Hayat, Z. a Buyya, R. Libra: a computational economy-based job scheduling system for clusters. *Software, practice & experience*. Chichester, UK: John Wiley & Sons, Ltd, 2004, zv. 34, č. 6, s. 573–590. ISSN 0038-0644.

[12] sleibso. *Bringing up the Avnet MicroZed with Vivado* [online]. Rev. 2021-10-14. Dostupné z: https://support.xilinx.com/s/article/Bringing-up-the-Avnet-MicroZed-with-Vivado?language=en_US. [cit. 2024-5-7].

[13] sleibso. *Adam Taylor's MicroZed Chronicles: Setting the SW Scene* [online]. Rev. 2021-11-08. Dostupné z: https://support.xilinx.com/s/article/364777?language=en_US. [cit. 2024-5-7].

Príloha A

Obsah priloženého pamäťového média

Príloha obsahuje dátovú štruktúru priloženého média a popis jednotlivých priečinkov a súborov.

- `xvesel92.pdf` - Verzia bakalárskej práce vo formáte PDF.
- `xvesel92/` - Priečinok obsahujúci zdrojové kódy pre textovú časť bakalárskej práce.
- `daemon/` - Priečinok obsahujúci zdrojové kódy démona.
- `Scripts/` - Priečinok obsahujúci skripty, ktoré sa používajú na zariadeniach PYNQ.
- `web/` - Priečinok obsahujúci zdrojové kódy webovej aplikácie.
- `VLab.sql` - Zdrojový kód generujúci databázu.
- `README.md` - Informácie o aplikácii, inštalácie a spustení.