ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra radioelektroniky

Pivní kotel s automatickým řízením

Dominik Roman
Elektronika a komunikace

Rok: 2023
Vedúci práce: doc. Ing. Stanislav Vítek, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Roman** Jméno: **Dominik** Osobní číslo: **491972**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Pivní kotel s automatickým řízením

Název bakalářské práce anglicky:

Brewery Boiler with Automatic Control

Pokyny pro vypracování:

Cílem práce je návrh a implementace sladového kotle pro domácí nebo komunitní vaření piva. Proces vaření piva bude řízen mikrokontrolérem, který bude vytvářet rozhraní (API) pro vzdálenou správu kotle. Součástí řešení je uživatelské rozhraní, které umožňuje registraci a přihlášení uživatele, tvorbu receptů vaření piva a jejich sdílení mezi uživateli.

Seznam doporučené literatury:

[1] MÅRTENSSON, Daniel. Principles of an automatic brewing machine: By using CodeSys with Raspberry Pi. 2017.
[2] WEEKS, Michael. Arduino controlled brewing. In: SoutheastCon 2015. IEEE, 2015. p. 1-5.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Stanislav Vítek, Ph.D. katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

doc. Ing. Stanislav Vítek, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Stanislav Vítek, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poďakovanie / Prehlásenie

Ďakujem svojmu vedúcemu práce doc. Ing. Stanislav Vítek, Ph.D. za neoceniteľné rady a pomoc pri tvorbe bakalárskej práce.

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržovaní etických princípov pri príprave záverečných prací.

V Prahe dňa 25. 05. 2023

.....

Dominik Roman

Abstrakt / Abstract

Táto bakalárska práca sa zaoberá návrhom a implementáciou pivného kotla s automatickým riadením, ktorý umožňuje domácim pivovarníkom dosiahnuť konzistentné výsledky pri varení piva. Systém sa skladá z viacerých komponentov, ako je mikročip, obrazovka, relé, teplomer, tepelné teleso a prevodník napätia. Implementovaná serverová časť využíva REST API a databázový systém Postgresql, zatiaľ čo klientská časť je postavená na technológiách Vue.js, Tailwind a PrimeVue. Cieľom práce je poskytnúť užívateľom jednoduché a efektívne riešenie pre dosiahnutie konzistentných výsledkov a zlepšenie kvality domáceho piva.

Kľúčové slová: pivný kotol, automatické riadenie, systém na varenie, domáci pivovar, mikročip, serverová časť, klientská časť

This bachelor's thesis focuses on the design and implementation of an automated brewing system, specifically a beer kettle with automatic control, which allows home brewers to achieve consistent results in beer brewing. The system consists of various components such as a microchip, screen, relay, thermometer, heating element, and voltage converter. The implemented server-side utilizes a REST API and the Postgresql database system, while the client-side is built using Vue.js, Tailwind, and PrimeVue technologies. The objective of this work is to provide users with a simple and efficient solution for achieving consistent results and improving the quality of homebrewed beer.

Keywords: beer kettle, automatic control, brewing system, home brewing, microchip, server-side, client-side

Title translation: Brewery Boiler with Automatic Control

Obsah /

Tabuľky / Obrázky

Kapitola 1
Úvod

V dnešnej dobe je výroba piva populárnym koníčkom mnohých ľudí. Domáce varenie piva sa stalo obľúbenou aktivitou a zdrojom kreativity a radosti z vlastnej výroby vynikajúceho piva. S rastúcim záujmom o domáce varenie piva však prichádzajú aj nové výzvy. Jednou z nich je potreba zabezpečovať konzistentné a kvalitné výsledky bez stáleho dohľadu a manuálneho riadenia procesu varenia.

Cieľom tejto bakalárskej práce je navrhnúť a implementovať systém, ktorý umožní automatizáciu procesu varenia piva doma. Systém by mal zahŕňať špeciálny pivný kotol vybavený riadiacimi prvkami a senzormi, ktoré umožnia monitorovanie a riadenie teploty, času varenia a ďalších dôležitých parametrov. Kombináciou moderných technológií a softvérového riešenia bude možné ovládať tento kotol z webového rozhrania, ktoré poskytne užívateľom jednoduché a intuitívne prostredie na nastavenie a sledovanie procesu varenia.

V priebehu práce si podrobnejšie preskúmané motivácie pre výrobu piva doma, vrátane jeho obľúbenosti a historického vývoja. Ďalej sa pozrieme na rôzne metódy varenia piva a súčasný stav domáceho varenia, vrátane ponuky na trhu a identifikácie problémov, ktoré sa môžu vyskytnúť pri manuálnom varení. Na základe analýzy navrhneme vhodné technológie pre automatizáciu procesu varenia a implementáciu systému. Výsledný systém bude obsahovať jednak hardvérovú časť, ako je mikročip pre riadenie kotla a senzory pre monitorovanie teploty, a jednak softvérovú časť, vrátane serverovej a klientskej časti, ktoré umožnia vzdialené ovládanie a sledovanie procesu varenia.

V nasledujúcich kapitolách detailnejšie popíšeme využité technológie, vrátane Internetu vecí (IoT), ktorý umožní komunikáciu medzi kotlom a webovým rozhraním, a ďalej zanalyzujeme jednotlivé komponenty systému, ako je mikročip, obrazovka, relé, teplomer, tepelné teleso a prevodník napätia. Popíšeme aj serverovú časť, ktorá zahŕňa REST API pre komunikáciu medzi klienskou časťou a kotlom, a využíva databázový systém Postgresql. Klientsku časť systému postavíme na technológiách Vue.js, Tailwind a PrimeVue, ktoré poskytnú užívateľsky prívetivé a interaktívne rozhranie pre ovládanie a monitorovanie procesu varenia.

Po detailnom preskúmaní využitých technológií a komponentov systému nasleduje kapitola venovaná implementácii. Tu popíšeme podrobnosti implementácie, vrátane zapojenia jednotlivých častí kotle, návrhu komunikácie medzi nimi a návrhu databázovej schémy. Ďalej opíšeme architektúru serverovej časti vrátane API koncových bodov, servisov, zabezpečenia a datových modelov. Klientskú časť odprezentujeme z hľadiska architektúry, vylepšení, stránok a komponentov, navigačných ciest, štýlovania pomocou Tailwind a používania knižnice PrimeVue. Implementácia pivného kotla bude zahŕňať aj funkcionalitu pripojenia na Wi-Fi, prevod čísel a zobrazenie na obrazovke, časovač a nastavenie času, ako aj algoritmy pre varenie s a bez Wi-Fi pripojenia.

V závere práce zhodnotíme využitia a možné vylepšenia systému. Odprezentujeme príklady použitia systému bez Wi-Fi pripojenia a s Wi-Fi pripojením prostredníctvom ukážky webového rozhrania. Okrem toho otvoríme diskusiu pre možné vylepšenia systému, ktoré by mohli byť implementované v budúcej práci.

Celkovo bude táto bakalárska práca prínosom pre domácich pivovarníkov, ktorí chcú mať plnú kontrolu a automatizáciu pri varení piva. Systém pivného kotla s automatickým riadením poskytne užívateľom jednoduché a efektívne riešenie pre dosiahnutie konzistentných výsledkov a zlepšenie kvality ich domáceho piva.

Kapitola 2
Motivácia

2.1 Obľúbenosť piva

Na svete pravdepodobne neexistuje človek, ktorý by nepoznal pivo. Podľa štatistík z roku 2022, malo pivo veľký podiel v svetovej ekonómii. Pivo je najkonzumovanejší alkoholický nápoj a celkovo tretí najkonzumovanejší nápoj na svete, po vode a čaji. Jedna zo stodesiatich práci je spojená s pivom, buď priamo, nepriamo, alebo má priamy dopad na pivný sektor. Pivný sektor podporuje približne 23,1 milóna práci na svete [1].

2.2 História piva

V súčasnosti sa historici domnievajú, že prvé fermentované nápoje existovali už asi pred 12 000 rokmi v čase, kedy ľudia začali pestovať plody. Existujú aj vecné dôkazy, ktoré históriu potvrdzujú, hovoríme napríklad o sumerskej tabuli, ktorá je asi 6000 rokov stará, alebo o básni, ktorá obsahuje najstarší známy recept a je stará asi 3900 rokov. Recepty sa najčastejšie šírili cez básne, alebo piesne [2].

Obrázok 2.1. Ukážka starého receptu vyrytého na kameni [3].

2.3 História domovarníctva

O počiatku domovarníctva všeobecná zhoda nie je. Môžeme ho napríklad položiť do stredoveku, kedy varili pivo mnísi v kláštore. Nevieme ale odlíšiť, či to boli začiatky domovarníctva, alebo pivného priemyslu. Medzi prvými domovarníkmi boli väčšinou ženy, ktoré varili pivo ako chutný nápoj v domácnostiach. Hoci to v tej dobe nevedeli, ale varením piva si nápoj dezinfikovali a tým predchádzali rôznym ochoreniam. Pivo pre nich bolo dokonca nie len formou hydratácie, ale aj zdrojom nutričných hodnôt. Počas obdobia objavovania nových území si ľudia so sebou prenášali svoje pivá a zásoby na ich výrobu. Aj napriek tomu, že pivarské fabriky vlastnili len bohatší ľudia, aj tí chudobnejší si vedeli nájsť spôsob ako si pivo uvariť doma, keďže to bol stále pre nich najbezpečnejší

nápoj. O tom ako bolo pivo a jeho varenie obľúbené značí aj príhoda pútnikov, ktorý museli neplánovane zakotviť v anglickom meste Plymouth, len preto, že im pivo na lodi došlo [2].

2.4 Vývoj vo varení

Pivovarníctvo prešlo asi najväčším technologickým vývojom počas priemyselnej revolúcie. Začiatkom 60. rokov 18. storočia boli vymyslené nástroje, ktoré su do dnes základom pri varení piva. Bavíme sa napríklad o teplomere, hustomere, či bubnovom pekáči, ktoré pomohli zlepšiť efektivitu pri varení a chuť samotného nápoja. V tom isto čase študoval francúzsky vedec Louis Pasteur mikroorganizmy a objavil úlohu kvasiniek v procese fermentácie. O tieto poznatky sa podelil s pivovarníkmi a tie im pomohli doladiť chuť a zabrániť kysnutiu a kazeniu piva.

V neposlednom rade priemyselná revolúcia priniesla inováciu v technológii plnenia fliaš. Výsledkom boli prvé hnedé pivné fľaše, ktoré zabránili preniknutiu UV lúčov a následnému tzv. skunkingu, čo je jav, pri ktorom svetlo, najmä ultrafialové žiarenie, pôsobí na chmeľové zlúčeniny obsiahnuté v pive a spôsobuje ich rozklad. Tento rozklad produkuje nepríjemnú vôňu, pripomínajúcu zápach skunka [4]. Ich zavedením sa pivo stalo vhodnejším na prepravu na väčšie vzdialenosti, pričom si uchovalo kvalitu [5].

2.5 Spôsoby varenia piva

Varenie piva sa dá rozdeliť na 3 spôsoby. Varenie z mladinového koncentrátu, ktoré je najjednoduchšou možnosťou. Proces prebieha tak, že sládek si zakúpi mladinový koncentrát, ktorý vo vode uvarí a následne ho dá kvasiť. Druhou možnosťou je varenie zo sladových výťažkov, pri ktorom sa namiesto sladu varí sladový koncentrát. A treťou možnosťou je varenie zo základných surovín. Poslednou možnosť sa budeme v práci podrobnejšie zaoberať a naše riešenie postavíme na nej [6].

2.6 Súčasný stav domovarníctva

2.6.1 Ponuka na trhu

V súčasnosti na našom trhu existuje niekoľko spoločností, ktoré ponúkajú na predaj kotle na domáce varenie piva. Ich ceny sa pohybujú od 150 eur vyššie. Najlacnejšie varianty ponúkajú nerezové telo, zväčša s dvojitým dnom a odpustom vody, možnosť nastavenia teploty pomocou potenciometra, kontrolku v podobe červenej diódy, ktorá signalizuje, či kotol práve ohrieva tekutinu. Ohrev je zabezpečený rovnakým princípom ako pri rychlovarnej kanvici, to znamená, že kanvica je vybavená bimetalovým pásikom alebo termostatom, ktorý po dosiahnutí nami zadanej teploty, zvyčajne s nejakou odchylkou pár stupňov, odpojí tepelné teleso od prívodu elektrickej energie. Drahšie varianty ponúkajú napríklad skladacie nádoby, do ktorých vieme umiestniť slad pri rmutovaní, cirkuláciu vody z dna kotla pomocou kompresoru a pridanej trubky, led obrazovku na zobrazenie teploty a časovaču, predvolené recepty a možnosť si predprogramovať recept. Cena sa samozrejme odlišuje aj objemom kotla. Objem kotla sa pohybuje zväčša v rozmedzí 30 až 70 litrov.

2.6.2 Problematika

Hlavným problémom komerčných kotlov určených pre domovarníctvo je ohrev vody. Ako je spomenuté v predošlej sekcii, ohrievanie funguje na rovnakom princípe ako rýchlovarná kanvica. Pri sladovare, aj pri užití sladového koša, dopadajú čiastočky sladu na dno hrnca, kde sa následne pripália. Potom vzniká pripálená vrstva na dne kotla, kotol to nevie správne vyhodnotiť, keďže aj teplomer je umiestnení na dne a tak sa začne samovoľne vypínať. Tento problém je iba čiastočne vyriešený cirkuláciou vody, ktorá bola spomenutá v predošlej sekcii. Ďalším problémom je umiestnenie teplomera na dne kotla. Teplota v kotli by mala byť na každom mieste rovnaká. Umiestnenie spôsobuje vysoké nezhody teplot medzi dnom a povrchom kotla. Problémom je aj nemožnosť byť notifikovaný o priebehu varu, či nemožnosť si predprogramovať recept prostredníctvom webovej, alebo mobilnej aplikácie. Recept následne vedieť zdieľať s ostatnými domácimi sladkami pomocou jednotnej platformy, ktorá by vedela ovládať pivný kotol. Azda najväčším problémom je cenová nedostupnosť, keďže pri lepších kotloch sa cena pohybuje približne okolo 500 eur.

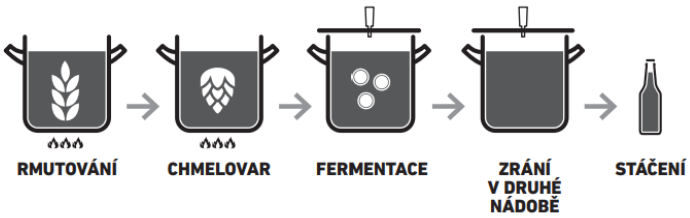
2.7 Návrh riešenia

Jedným z možných riešení je vytvorenie portálu, na ktorý sa budú môcť domáci sladkovia prihlásiť, vytvoriť si recept a nazdieľať ho ostatným. Cez portál by bolo možné napríklad spústiť varenie, sledovať stav kotla a zastaviť varenie. Kotol by bol pripojený na internetovú sieť a zasielal požiadavky na server, ktorý by ich vyhodnocoval, spracovával a odpovedal na ne. V kotli by boli aspoň 2 senzory teploty, ktoré by teplotu napríklad priemerovali. Zmenil by sa aj spôsob zohrievania kvapaliny a využilo by sa napríklad tyčové tepelné teleso.

Kapitola 3
Varenie zo základných surovín

3.1 Obsah receptu a ako sa varí

Telo receptu vieme rozdeliť do 4 základných častí medzi suroviny, rmutovanie, chmelovar a fermentáciu [6]. Jednotlivé časti si popíšeme a vysvetlíme si pri nich ako sa varí pivo.



Obrázok 3.1. Ukážka postupu varenia [7].

3.1.1 Suroviny

Táto časť slúži ako súhrn všetkého, čo je pre varenie potrebné. Nájdeme v nej koľko rmutovacej a koľko vysladzovacej vody potrebujeme. Množstvo a typ sladov, ktoré potrebujeme. Množstvo a typ chmeľu. Aké kvasinky budeme používať, alebo prídané veci pri fermentácii a samozrejme cukor, s ktorým sa má pivo fľašovať [6].

SUROVINY

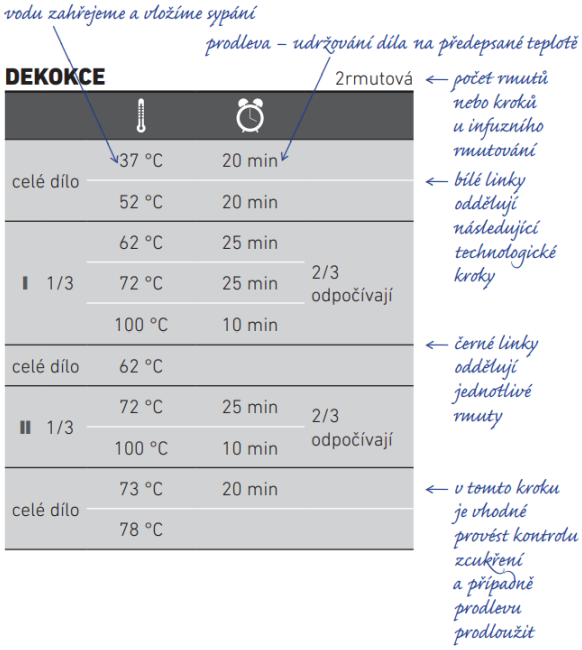
	Rmutovací		17,0 l
	Vyslazovací		18,0 l
	Plzeňský	4,34 kg	(87,3 %)
	Mnichovský	0,54 kg	(10,9 %)
	Karamelový	0,09 kg	(1,8 %)
	Žatecký červeňák		24 g
	Premiant		13 g
	Spodní z pivovaru – Saflager W-34/70		150 ml

Obrázok 3.2. Ukážka časti suroviny v recepte [7].

3.1.2 Rmutovanie

Pri rmutovaní sa voda zahreje na prvú potrebnú teplotu, vysype sa do nej slad a udržuje sa určitý čas na danej teplote. Zahrievanie by malo byť plynulé a nemalo by prekročiť 1°C/min. Rmutovanie vieme rozdeliť na 2 formy a to na dekokčnú a infúznu. Dekokcia môže byť niekoľko rmutová. To znamená, že ak je napríklad dvojrmutová, tak sa celé

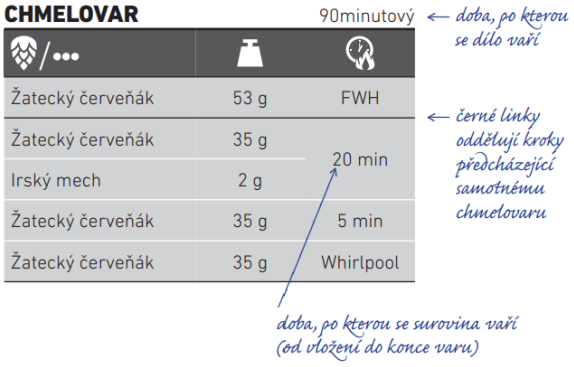
dielo varí pri zadaných teplotách a počas rmutov, teda v tomto prípade počas dvoch rmutov, sa dielo rozdelí na viac častí a každá časť sa varí zvlášť. Pri infúznej forme sa dielo nerozdeľuje, ale vždy sa varí spolu. Po dokončení rmutovania dielo necháme vychladnúť, prečerpáme ho a slad vyplachneme vodou s určitou teplotou [6].



Obrázok 3.3. Ukážka časti rmutovanie v recepte s popisom [7].

3.1.3 Chmeľovar

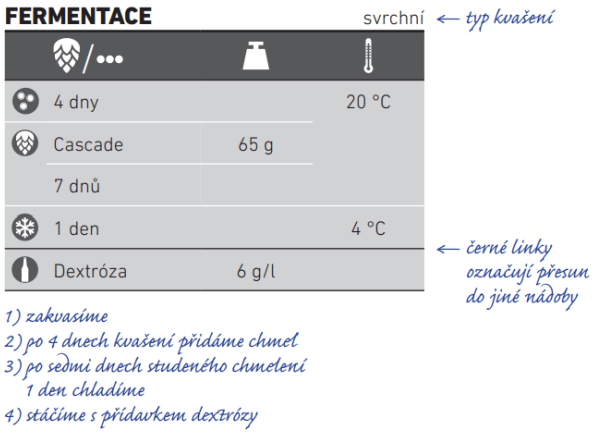
Chmeľovar je sekcia pri ktorej sa varí chmeľ najčastejšie pri teplote 100°C. Vždy je zadaný čas, ktorý popisuje ako dlho daný krok trvá. V krokoch sú potom zadané druhy, hmotnosti a čas jednotlivých druhov chmeľu, ktoré sa budú variť. Čas, ktorý je zadaný znamená dobu ako dlho sa má chmeľ variť. To znamená, že ak máme zadaný chmeľ x a 10 minút, tak chmeľ x dáme variť na 10 minút pred koncom [6].



Obrázok 3.4. Ukážka časti chmeľovar v recepte s popisom [7].

3.1.4 Fermentácia

Fermentáciu vieme taktiež rozdeliť do viacerých častí, keďže pivo zreje aj po dobe, keď je dielo odobrané z fermentačnej nádoby. Prvá časť fermentácie je tá, pri ktorej dielo fermentuje vo fermentačnej nádobe. Ako druhú časť môžeme považovať časť po umiestnení piva do kegu, alebo fliaš, buď so zbytkovým, alebo prídavným cukrom, kde následne pivo zreje [6].



Obrázok 3.5. Ukážka časti fermentácia v recepte s popisom [7].

Kapitola 4
Využité technológie

4.1 Cieľ

Cieľom práce bude vytvoriť inteligentný kotol na varenie piva. Kotol bude komunikovať so serverom, s ktorým si bude vymieňať dáta. Následne sa data spracujú, vyhodnotia a uložia do databázy. V klientskej časti bude vytvorené užívateľské rozhranie alebo portál, v ktorom si uživatelia budú môcť vytvárať vlastné recepty, zdieľať ich, alebo ich dať variť. Rovnako ako z kotla, tak aj z uživateľského rozhrania budú data spracovávané, vyhodnocované a ukládané do databázy.

4.2 IoT

4.2.1 Čo pojem znamená

IoT (Internet of Things) je technologický koncept, ktorý umožňuje sieťové prepojenie fyzických zariadení a ich vzájomnú komunikáciu prostredníctvom internetu. Využitie IoT technológií v priemysle a v domácnostiach je v súčasnosti veľmi rozšírené a denne sa zvyšuje počet zariadení, ktoré sú schopné byť pripojené na internet a interagovať s inými zariadeniami [8].

4.2.2 Čím sa zaoberá

IoT má široké uplatnenie v rôznych oblastiach, vrátane priemyslu, energetiky, dopravy, zdravotníctva a aj v domácnostiach. V priemysle sa využíva na monitorovanie a riadenie výrobných liniek a zariadení, čím sa zvyšuje efektivita a produktivita. V energetike umožňuje monitorovanie spotreby energie a riadenie spotreby v reálnom čase, čím sa znižujú náklady a zvyšuje efektivita. V doprave umožňuje monitorovanie vozidiel a premávky, čím sa zlepšuje bezpečnosť a znižujú sa časy čakania. V zdravotníctve sa využíva na sledovanie zdravia pacientov, monitorovanie liekov a zariadení, čím sa zvyšuje kvalita starostlivosti. V domácnostiach sa využíva na automatizáciu a riadenie rôznych zariadení, ako sú napríklad osvetlenie, kúrenie, klimatizácia, bezpečnostné kamery, čím sa zvyšuje pohodlie a efektivita [8].

4.3 Kotol

4.3.1 Modul

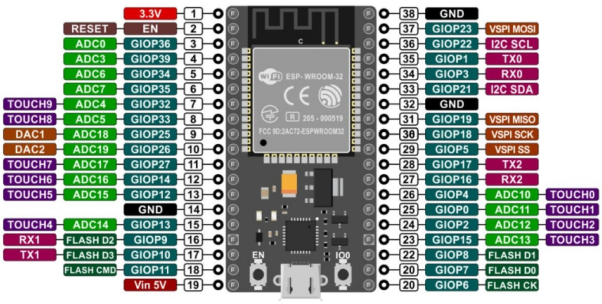
Riadiacim a najdôležitejším prvkom časti kotol bude modul ESP-WROOM-32.

ESP32-WROOM-32 je vysoko výkonný modul pre WiFi a Bluetooth komunikáciu, navrhnutý spoločnosťou Espressif Systems. Modul je založený na čipe ESP32, ktorý je vybavený dvojicou procesorov s frekvenciou 240 MHz a podporou WiFi a Bluetooth komunikácie.

Má niekoľko výhod oproti iným modulom na trhu. Jednou z týchto výhod je vysoká rýchlosť procesora, ktorá umožňuje vyššiu rýchlosť prenosu dát v sieťach Wi-Fi a Bluetooth. Modul tiež podporuje rôzne režimy spotreby energie, čo umožňuje jeho použitie v batériovo napájaných zariadeniach.

Je vybavený rôznymi perifériami, ktoré zahŕňajú 34 GPIO pinov, 3 UART rozhrania, 3 SPI rozhrania, 2 I2C rozhrania, 18 kanálov PWM a 10-bitový ADC. Modul tiež podporuje rôzne vstupné a výstupné signály, ako napríklad GPIO, I2S, I2C, PWM, UART, SPI a ADC.

Modul sa používa v rôznych aplikáciách, ako napríklad v priemysle, IoT aplikáciách, robotike, inteligentných domácnostiach a mnohých ďalších oblastiach. Vďaka svojej vysokorýchlostnej komunikácii, nízkej spotrebe energie a rôznym perifériám, umožňuje taktiež vývojárom a koncovým používateľom vytvárať vysoko výkonné zariadenia a aplikácie [9].



Obrázok 4.1. Ukážka modulu ESP32-WROOM-32 [10].

4.3.2 Obrazovka

Ako obrazovku použijeme Waveshare 1,9 palcový 91 segmentový e-Paper displej. Je to zariadenie, ktoré umožňuje zobrazenie informácií pomocou e-Paper technológie. Má veľkosť 1,9 palca a je navrhnutý tak, aby sa dal jednoducho použiť s rôznymi mikrokontrolérmi a platformami.

Jednou z najväčších výhod e-Paper displejov je ich nízka spotreba energie. Displej nepotrebuje energiu na udržanie zobrazenia, čo znamená, že spotrebuje energiu len počas aktualizácie obrazovky.

Displej sa pripája k mikrokontroléru pomocou I2C rozhrania. Je vybavený interným kontrolérom, ktorý umožňuje zobrazovať texty, obrázky a iné grafické prvky. Podporuje 91 segmentov, čo umožňuje zobrazenie veľkého množstva informácií na malom displeji.

Má rozlíšenie 128 x 32 pixelov a zobrazuje čierne a biele farby. Jeho nízka spotreba energie umožňuje, aby bol napájaný napríklad pomocou batérií. Displej je tiež odolný voči mechanickým nárazom a vibráciám, čo z neho robí vhodnú voľbu pre aplikácie, ktoré vyžadujú robustné a spoľahlivé zariadenia.

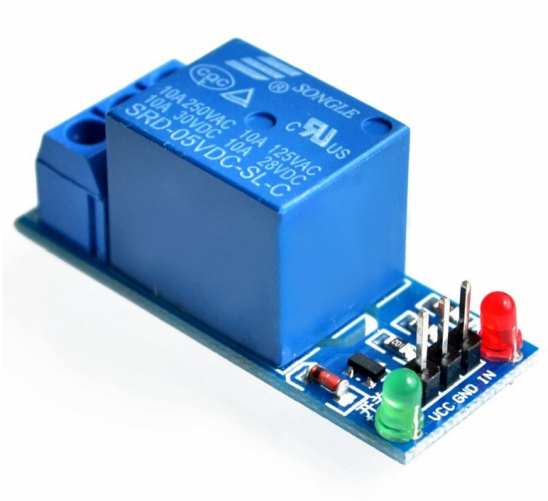
Môže byť použitý v rôznych aplikáciách, ako napríklad v IoT aplikáciách, vývoji embedded zariadení, náramkových zariadeniach a mnohých ďalších. Jeho malá veľkosť a nízka spotreba energie umožňujú vytvárať zariadenia, ktoré sú malé, ľahké a majú dlhú výdrž batérie [11].

Obrázok 4.2. Obrazovka Waveshare [12].

4.3.3 Relé

1-kanálový 5V relé modul je elektronické zariadenie používané na prepínanie vysokých napätí a prúdov pomocou nízkonapäťového signálu. Tento konkrétny modul má jeden kanál, čo znamená, že môže prepínať len jeden obvod alebo zariadenie. Vstupné napätie je 5V a jeho max. záťaž je 250V striedavého napätia a 10A.

Tento modul má taktiež tzv. `low-level trigger` (taktiež známy ako `active-low`), čo znamená, že sa spína pomocou nízkej úrovne signálu. Pri vstupe 0 sa relé zapne a pri vstupe 1 sa relé vypne. To znamená, že pri návrhu a použití tohto modulu musíme brať do úvahy, že sme nútení poskytnúť nízkonapäťový signál na vstupe pre riadenie relé [13].



Obrázok 4.3. 1-kanálový 5V relé modul [14].

4.3.4 Teplomer

DS18B20 je digitálny teplomer s jedným pripojením a komunikačným protokolom `1-Wire`, ktorý umožňuje komunikáciu s mikrokontrolérmi a ďalšími zariadeniami. Tento senzor teploty sa vyznačuje vysokou presnosťou a stabilnosťou a je často používaný v rôznych aplikáciách, vrátane merania teploty v interiéroch, priemyselných procesoch a v poľnohospodárstve. Senzor môže merať teplotu v rozsahu -55°C až +125°C s presnosťou ±0,5°C v rozmedzí teploty od -10°C do +85°C. To znamená, že tento senzor teploty je schopný merať teplotu v širokom rozmedzí.

Pripojenie senzora k mikrokontroléru je možné pomocou 3 vodičov (červený, čierny a biely). Červený a čierny vodič slúžia na napájanie senzora, zatiaľ čo biely vodič slúži na prenos dát medzi senzorom a mikrokontrolérom [15].

Obrázok 4.4. Digitálny teplomer DS18B20 [16].

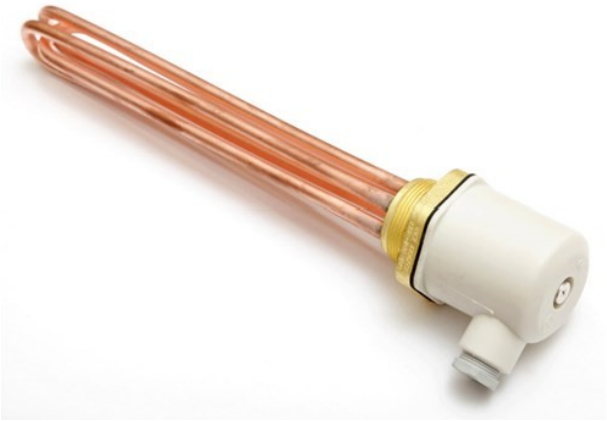
4.3.5 Tepelné teleso

Tepelné teleso využijeme medené s výkonom 1500W, napájané 230V, ktoré je vhodné pre elektrokotle, alebo bojlery. Je vytvorené z medi. Niektorí sládkovia veria, že meď je najlepší materiál pre varenie piva, pretože zaujímavo reaguje spolu s cukrom a spolu to môže vytvoriť príjemnú karamelovú chuť piva [17]. Meď taktiež výborne vedie teplo. Výkon topného telesa sme spočítali pomocou vzťahu:

$$t.v. = 4,2 \cdot m \cdot \frac{\Delta T}{\Delta t} + str \tag{1}$$

kde t.v. je tepelný výkon (kW), 4,1 je tepelná kapacita vody (kJ/kg/°C), m hmotnosť ohrievanej vody v kilogramoch a ΔT/Δt je žiadaná rýchlosť ohrevu (°C/s), str sú predpokladané straty v percentách. Dielo je tvorené väčšinou vodou, tak pri tomto výpočte môžme počítať s tepelnou kapacitou vody.

Po dosadení 25 kilogramov s rýchlosťou ohrevu 1°C za minútu a pri optimistickom predpoklade nulových strát dostávame výsledok 1,8kW. Volíme tepelné teleso s výkonom 1,5kW. Ak by sme zvolili teleso s výkonom nad 1,8kW, ohrev by bol rýchlejší ako 1°C za minútu. To by mohlo spôsobiť zničenie časti enzýmov a zvýšiť riziko pripálenia diela [6].



Obrázok 4.5. Tepelné teleso [18].

4.3.6 Prevodník napätia

Ako prevodník napätia využijeme sieťový adaptér 5V/3A. Vstup do adaptéru je 110V-240V 50/60Hz. Výstup je 3A 5V s konektorom microUSB. Adaptér bude napájať modul ESP32 4.3.1.

Obrázok 4.6. Sieťový adaptér 5V/3A [19].

Ako ďalšie súčiastky využijeme tlačidlá, ktoré budú slúžiť na prepínanie stavov v kotli, rezistory, led na signalizáciu stavu, spínač na zapínanie, alebo vypínanie kotla a flexišnúru.

4.4 Serverová časť

4.4.1 REST API

REST API (Representational State Transfer Application Programming Interface) je populárny spôsob, ako navrhnúť a implementovať API (Application Programming Interface) na prenos dát medzi webovými aplikáciami. REST API je navrhnuté na základe architektúry klient-server, kde klient a server komunikujú prostredníctvom štandardných HTTP požiadaviek a odpovedí.

Používa základné HTTP metódy, ako sú GET, POST, PUT, DELETE a PATCH, na manipuláciu s dátami. Každý z týchto metód má určený význam a použitie:

- **GET:** slúži na získanie dát z webovej aplikácie.
- **POST:** slúži na vytvorenie nových dát v webovej aplikácii.
- **PUT:** slúži na aktualizáciu existujúcich dát v webovej aplikácii.
- **DELETE:** slúži na odstránenie dát z webovej aplikácie.
- **PATCH:** slúži na aktualizáciu časti existujúcich dát v webovej aplikácii.

Zvyčajne používa JSON formát na prenos dát medzi klientom a serverom. JSON (JavaScript Object Notation) je textový formát, ktorý umožňuje zápis štruktúrovaných dát [20].

Je návrhovým vzorom, ktorý umožňuje jednoduché a efektívne navrhovanie a implementovanie API. REST API má niekoľko výhod, ako sú jednoduchá implementácia, zvýšená škálovateľnosť, znovupoužiteľnosť kódu a jednoduchá testovateľnosť. REST API sa používa vo veľkom množstve webových aplikácií, vrátane sociálnych médií, elektronického obchodu a správy obsahu [21].

4.4.2 Postgresql

PostgreSQL je open-source relačná databáza, ktorá sa používa v rôznych oblastiach, vrátane webových aplikácií, podnikových informačných systémov a vedeckých aplikácií. Je to jeden z najrozšírenejších relačných databázových systémov a získal si popularitu pre svoje pokročilé funkcie a možnosti.

Používa architektúru klient-server, kde klienti sa pripájajú na server, kde beží databázový server PostgreSQL. Server spracováva požiadavky od klientov a vykonáva operácie s dátami. Jednou z výhod tejto architektúry je možnosť centralizácie dát a správy prístupových práv používateľov.

Podporuje množstvo dátových typov, vrátane klasických dátových typov, ako sú čísla, reťazce a dátumy, ale aj pokročilých typov, ako sú polia, JSON, XML a rôzne geometrické typy. Navyše, používatelia môžu vytvárať vlastné dátové typy na základe svojich potrieb [22].

4.4.3 FastAPI a SQLAlchemy

FastAPI je webový rámec, ktorý bol vytvorený pre jednoduchú a efektívnu tvorbu API s použitím jazyka Python. Tento rámec je postavený na Starlette, ktorý je ľahký a rýchly rámec pre webové aplikácie, a na Pydantic, ktorý je modul pre serializáciu a validáciu dát. S použitím týchto modulov, FastAPI umožňuje rýchle vytvorenie a dokumentáciu API.

FastAPI ponúka množstvo funkcií, ktoré sú využiteľné pri tvorbe API. Medzi ne patrí automatická dokumentácia API pomocou OpenAPI a JSON Schema, podpora pre `async/await`[1], validácia dát, `dependency injection`[2] a mnoho ďalších [25].

SQLAlchemy je vysokoúrovňová ORM knižnica, ktorá umožňuje pracovať s relačnými databázami pomocou objektovo-orientovaného prístupu. SQLAlchemy poskytuje výkonné a flexibilné rozhranie pre prácu s databázami, ktoré umožňuje vytváranie, čítanie, aktualizovanie a mazanie (CRUD) dát v databáze. SQLAlchemy podporuje množstvo relačných databázových systémov, vrátane MySQL, PostgreSQL, Oracle, SQLite a mnoho ďalších [26].

V kombinácii s FastAPI, SQLAlchemy umožňuje jednoduchú a efektívnu prácu s databázou. FastAPI umožňuje rýchle vytvorenie API a SQLAlchemy poskytuje výkonné rozhranie pre prácu s databázou. Kombinácia týchto dvoch nástrojov umožňuje programátorom vytvárať efektívne a škálovateľné webové aplikácie s použitím programovacieho jazyka Python [27].

4.5 Klientská časť

4.5.1 Vue.js

Vue.js je moderný JavaScriptové rozšírenie určené pre tvorbu webových aplikácií. Jeho návrh sa zameriava na jednoduchosť, flexibilitu a rýchlosť vývoja, čo ho robí veľmi populárnym u vývojárov po celom svete.

Vue.js má veľmi malú veľkosť a jednoduchú syntax, čo umožňuje rýchlejší vývoj a jednoduchú údržbu kódu. Jeho základom sú tzv. komponenty, ktoré sa dajú jednoducho vytvárať, distribuovať a opätovne používať v rôznych častiach aplikácie. Vue.js taktiež umožňuje jednoduchú integráciu s inými knižnicami a rozšíreniami.

Medzi jeho výhody patrí aj jeho dobrá dokumentácia, veľké množstvo komunitných modulov, rozšírení a silná podpora od spoločnosti, ktorá ho vyvinula.

Umožňuje vývoj webových aplikácií v reálnom čase, čo znamená, že zmeny sa okamžite zobrazia bez nutnosti manuálneho obnovenia stránky. To uľahčuje testovanie a zlepšenie pre používateľov [28].

[1] umožňuje jednoduché a čisté programovanie asynchrónnych operácií v jazykoch, ktoré ho podporujú [23].
[2] je návrhový vzor, kde závislosti objektu sú poskytnuté externým zdrojom namiesto toho, aby boli vytvárané samotným objektom [24].

4.5.2 Tailwind

Tailwind je moderný CSS rámec, ktorý zjednodušuje tvorbu webových aplikácií pomocou preddefinovaných tried CSS. Poskytuje viac ako 300 tried, ktoré sa dajú použiť pri tvorbe webovej stránky. Tieto triedy zjednodušujú tvorbu, údržbu kódu a umožňujú vývojárom efektívne pracovať.

Triedy Tailwind sa zvyčajne používajú priamo v HTML kóde a slúžia na definovanie rôznych štýlov. Napríklad trieda `text-blue-500` nastaví farbu textu na modrú. Triedy sú založené na filozofii názvoslovia `utility-first`, čo znamená, že sú zamerané na funkčnosť a použitie, a nie na estetický vzhľad.

Okrem toho, poskytuje aj vlastné utility triedy pre responzívne dizajn, ktoré sa dajú použiť pre rôzne obrazovky a zariadenia. Tailwind taktiež umožňuje jednoduché prispôsobenie vlastným potrebám prostredníctvom súbora konfigurácie, ktorý umožňuje zmeniť farby, veľkosti a mnoho ďalších vlastností tried.

Výhody Tailwindu sú rýchlosť vývoja, zjednodušenie údržby kódu a škálovateľnosť. Vďaka použitiu preddefinovaných tried sa minimalizuje nutnosť písania vlastného CSS a umožňuje rýchlejšie vytváranie a úpravy štýlov [29].

4.5.3 PrimeVue

PrimeVue je open-source knižnica pre Vue.js, ktorá poskytuje komponenty pre tvorbu moderných webových aplikácií. Táto knižnica obsahuje širokú škálu komponentov, vrátane tabuliek, formulárov, dialogov, grafických prvkov a mnoho ďalších.

Jednou z veľkých výhod PrimeVue je, že komponenty sú veľmi dobre navrhnuté a flexibilné. To znamená, že môžeme ľahko prispôsobiť vzhľad a správanie týchto komponentov podľa našich potrieb. PrimeVue tiež poskytuje mnoho možností, ako prispôsobiť tieto komponenty tak, aby sa ľahko integrovali s inými knižnicami a rozšíreniami.

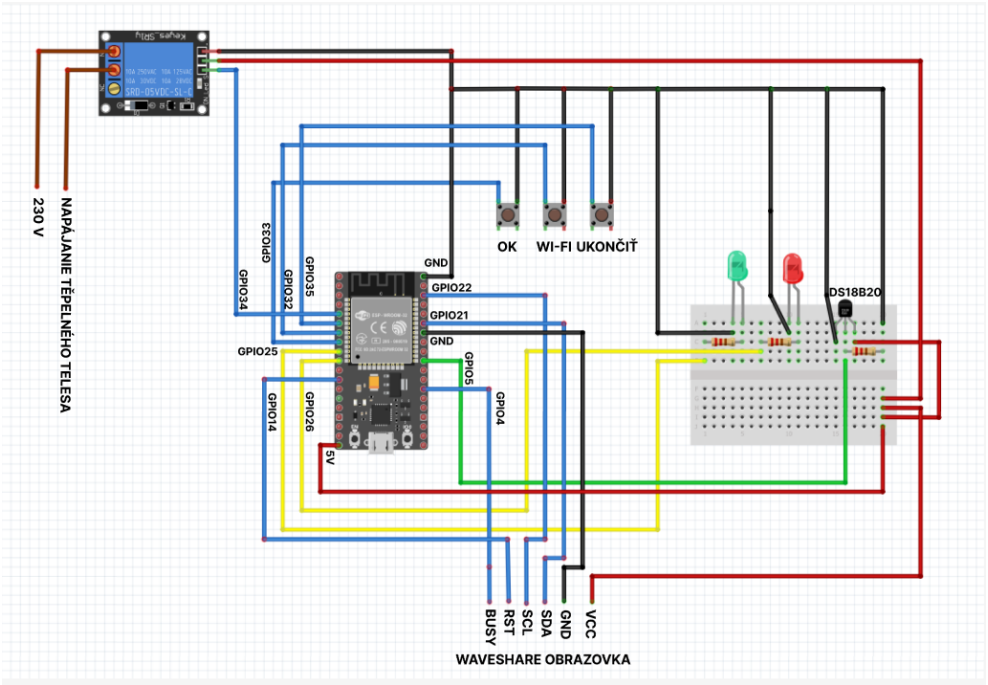
Ďalšou výhodou PrimeVue je, že je veľmi dobre zdokumentovaná a má aktívnu komunitu. To znamená, že v prípade otázok alebo problémob, existuje veľká šanca, že niekto v komunite bude schopný pomôcť. Okrem toho je knižnica aktualizovaná na pravidelnej báze, takže môžeme očakávať, že webové aplikácie budú vždy v súlade s najnovšími technológiami a trendmi [30].

Kapitola 5
Implementácia

5.1 Úvod do implementácie

5.1.1 Zapojenie časti kotol

Ako je z obrázka 5.1 vidno, schéma sa skladá zo súčiastok, ktoré boli popísané v kapitole 4.3. Srdce nášho obvodu bude tvoriť modul ESP-32 4.3.1, ktorý bude mať na starosti ovládanie celého obvodu a zasielanie požiadavok na server. Bude napájaný z adaptéru 4.3.1. Do relé 4.3.1 bude vstupovať sieťové napätie 230V fázovým vodičom do pinu `NC` (normaly closed). Na fázovom vodiči zo sieťového napätia pridáme ešte spínač, ktorý bude normovaný na 230V a bude slúžiť na zapínanie a vypínanie kotla. Z pinu `CO` (common) vyvedieme vodič do napájania tepelného telesa 4.3.1. Cez GPIO piny prepojíme tlačidlá `OK`, `WI-FI` a `UKONČIŤ`, LED svetlá slúžiace na indikovanie varenia, či pripojenia na WiFi a senzor teploty DS18B20 4.3.1.



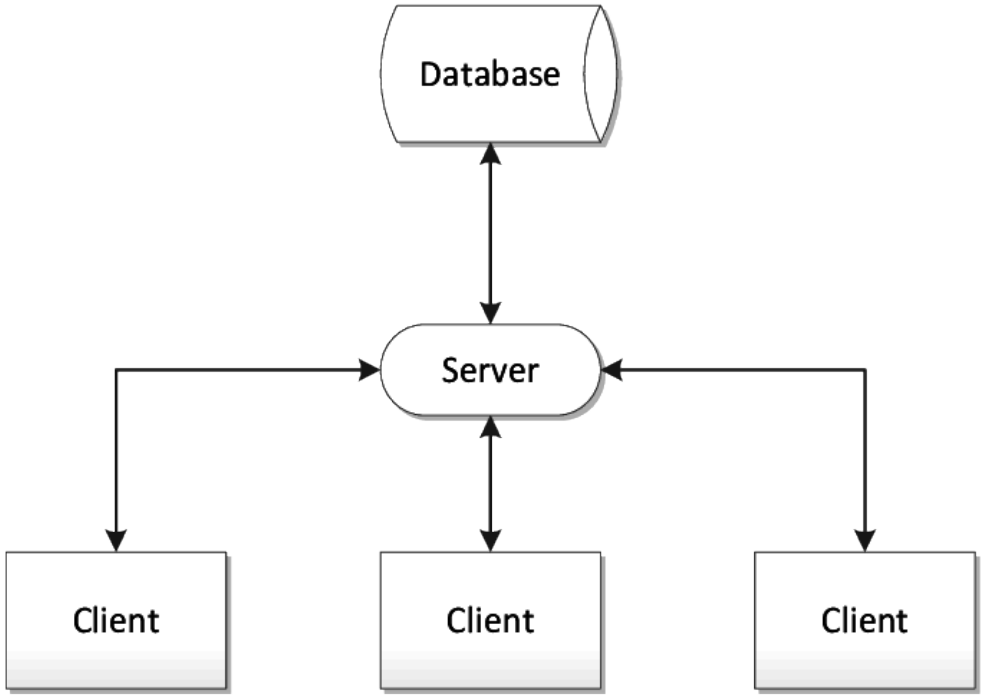
Obrázok 5.1. Schéma zapojenia.

O vyobrazenie aktuálnej teploty a času vareni sa nám postará obrazovka 4.3.1. Jej prepojenie je následovne:

- **SDA (Serial Data) -> GPIO21:** SDA pin slúži na prenos sériových dát medzi ESP32 a displejom pomocou I2C protokolu. GPIO21 je schopný fungovať ako I2C datový pin [31].
- **SCL (Serial Clock) -> GPIO22:** SCL pin je zodpovedný za generovanie hodín pre I2C komunikáciu medzi ESP32 a displejom. GPIO22 bol vybraný ako SCL pin kvôli jeho podpore I2C komunikácie [31].
- **RST (Reset) -> GPIO14:** BUSY pin na displeji indikuje, či je displej momentálne zaneprázdnený alebo vykonáva nejakú operáciu. Pripojenie k GPIO4 je vybrané pre jednoduchú správu a monitorovanie stavu displeja [32].
- **VCC -> 5V**
- **GND -> GND**

5.1.2 Návrh komunikácie

Tak ako v predošlej sekcii 5.1.1 bol srdcom modul 4.3.1, tak v tejto sekcii bude srdcom server. Server môžeme považovať ako srdce celej tejto práce. Bude mať na starosti komunikáciu s databázou, klientskou časťou a komunikáciu s modulom 4.3.1.

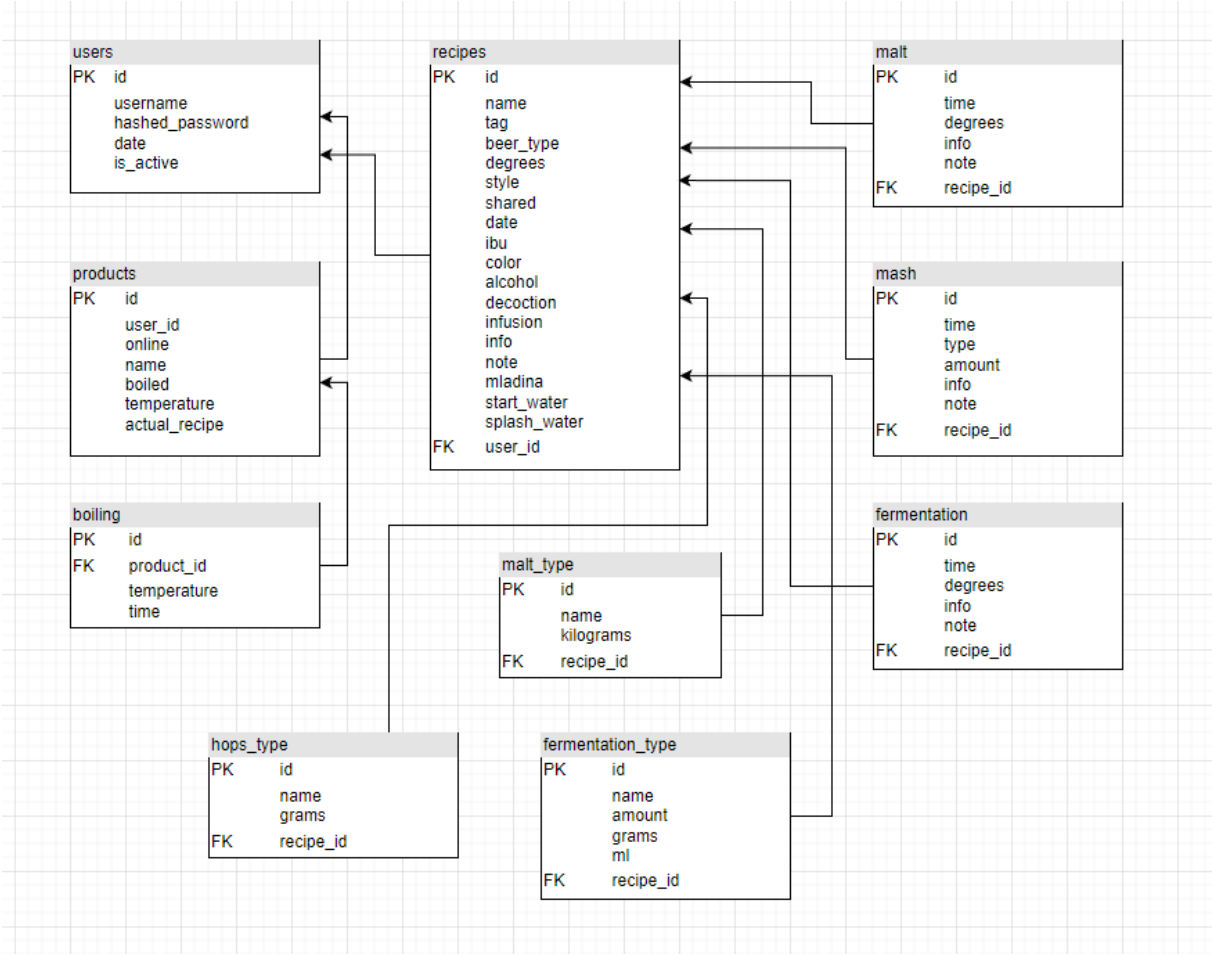


Obrázok 5.2. Schéma komunikácie [33].

Ako na ilustračnom obrázku 5.2 vidno, tak celý koncept bude fungovať tak, že budeme mať v našom prípade v 3. úrovni 2 časti Client. Jedná bude náš modul 4.3.1 a tá druhá bude klientská časť 4.3.1. Obe tieto časti budú komunikovať so serverom prostredníctvom požiadavok skrz HTTP metódy 4.3.1. Server bude tieto požiadavky spracovávať a následne im na ne odpovedať. Rovnako tak bude komunikovať s databázou.

5.1.3 Návrh databázovej schémy

Aby sme mohli spracovávať data do databázy, tak musíme mať vytvorenú databázovú schému.



Obrázok 5.3. Databázová schéma.

Ako na obrázku 5.3 môžeme vidieť, tak máme 10 tabuliek. Každá tabuľka má ako svoj primárny kľúč `id`. Tabuľky v schéme:

- **users:** tabuľka, do ktorej budeme ukládať uživateľov portálu. Má tieto atribúty:
  - **username:** textový reťazec, označuje užívateľské meno.
  - **hashed_password:** textový reťazec, označuje zakódované heslo.
  - **date:** dátový typ `Date`, označuje dátum vytvorenia účtu.
  - **is_active:** `boolean`, označuje či je účet aktívny.

Je prepojená vzťahom 1:N s tabuľkou `recipes` a 1:N s tabuľkou `products`.

- **products:** tabuľka, do ktorej budeme vkladať používateľové produkty, respektíve zariadenia. Má tieto atribúty:
  - **user_id:** cudzí kľúč odkazujúci na id užívateľa tabuľky `users`.
  - **online:** `boolean`, označuje či je produkt pripojený na WiFi.
  - **name:** textový reťazec, označuje názov produktu.
  - **boiled:** `boolean`, označuje či sa produkt práve používa.
  - **temperature:** textový reťazec, označuje aktuálnu teplotu kvapaliny v produkte.
  - **actual_recipe:** textový reťazec, označuje názov receptu, ktorý sa aktuálne varí.

Je prepojená s tabuľkou `users` vzťahom N:1 a a s tabuľkou `boiling` vzťahom 1:N.

- **boiling:** tabuľka, do ktorej budeme vkladať teploty z receptu, na ktorých sa varí aktuálny recept. Má tieto atribúty:
  - **product_id:** cudzí kľúč odkazujúci na tabuľku `products`.
  - **temperature:** textový reťazec, potrebná teplota varenia.
  - **time:** textový reťazec, potrebný čas varenia.

Je prepojená s tabuľkou `products` vzťahom N:1.

- **recipes:** tabuľka, do ktorej sa ukládajú vytvorené recepty užívateľom. Má tieto atribúty:
  - **name:** textový reťazec, označuje názov receptu.
  - **tag:** textový reťazec, označuje označenie receptu autorom.
  - **beer_type:** textový reťazec, označuje druh piva.
  - **degrees:** textový reťazec, označuje stupňovitosť piva.
  - **style:** textový reťazec, označuje štýl piva.
  - **shared:** boolean, označuje, či pivo má byť viditeľné aj ostatným užívateľom.
  - **date:** dátový typ `Date`, označuje dátum vytvorenia receptu.
  - **ibu:** textový reťazec, označuje hodnotu IBU piva.
  - **color:** textový reťazec, označuje farbu piva.
  - **alcohol:** textový reťazec, označuje alkohol v percentách piva.
  - **decoction:** `boolean`, označuje akým spôsobom sa bude variť rmut, dekokciou.
  - **infusion:** `boolean`, označuje akým spôsobom sa bude variť rmut, infúziou.
  - **info:** textový reťazec, označuje dodatočné informácie o pive,
  - **note:** textový reťazec, označuje poznámku užívateľa k pivu.
  - **mladina:** textový reťazec, označuje na koľko litrov vody je recept urobený.
  - **start_water:** textový reťazec, označuje s koľko litrami vody začína var.
  - **splash_water:** textový reťazec, označuje koľko litrov vody je potrebných na preplachnutie sladu.
  - **user_id:** cudzí kľúč, odkazuje na tabuľku `users`.

Je prepojená s tabuľkami `malt`, `mash`, `fermentation`, `mash_type`, `hops_type`, `fermentation_type` vzťahom 1:N a s tabuľkou `users` vzťahom N:1.

- **malt:** tabuľka, do ktorej budeme ukládať časť rmutovanie z receptu. Má tieto atribúty:
  - **time:** textový reťazec, označuje čas ako dlho bude časť trvať.
  - **degrees:** textový reťazec, označuje na koľkych stupňoch sa bude časť variť.
  - **info:** textový reťazec, označuje dodatočné informácie o časti.
  - **note:** textový reťazec, označuje poznámku užívateľa k časti.
  - **recipe_id:** cudzí kĺúč, odkazujúci na tabuľku `recipes`.

Je prepojená vzťahom N:1 s tabuľkou `recipes`.

- **mash:** tabuľka, do ktorej budeme ukládať časť chmeľovar z receptu. Má tieto atribúty:
  - **time:** textový reťazec, označuje čas ako dlho bude časť trvať.
  - **type:** textový reťazec, označuje druh chmeľu.
  - **amount:** textový reťazec, označuje množstvo chmeľu.
  - **info:** textový reťazec, označuje dodatočné informácie o časti.
  - **note:** textový reťazec, označuje poznámku užívateľa k časti.
  - **recipe_id:** cudzí kĺúč, odkazujúci na tabuľku `recipes`.

Je prepojená vzťahom N:1 s tabuľkou `recipes`.

- **fermentation:** tabuľka, do ktorej budeme ukládať časť fermentácia z receptu. Má tieto atribúty:
  - **time:** textový reťazec, označuje čas ako dlho bude časť trvať.
  - **degrees:** textový reťazec, označuje na koľkych stupňoch bude fermentovať.
  - **info:** textový reťazec, označuje dodatočné informácie o časti.
  - **note:** textový reťazec, označuje poznámku užívateľa k časti.
  - **recipe_id:** cudzí kĺúč, odkazujúci na tabuľku `recipes`.

Je prepojená vzťahom N:1 s tabuľkou `recipes`.

- **malt_type:** tabuľka, do ktorej budeme ukládať časť suroviny pre rmutovanie z receptu. Má tieto atribúty:
  - **name:** textový reťazec, označuje názov sladu.
  - **kilograms:** textový reťazec, označuje množstvo sladu v kilogramoch.
  - **recipe_id:** cudzí kĺúč, odkazujúci na tabuľku `recipes`.

Je prepojená vzťahom N:1 s tabuľkou `recipes`.

- **hops_type:** tabuľka, do ktorej budeme ukládať časť suroviny pre chmeľovar z receptu. Má tieto atribúty:
  - **name:** textový reťazec, označuje názov chmeľu.
  - **grams:** textový reťazec, označuje množstvo chmeľu v gramoch.
  - **recipe_id:** cudzí kĺúč, odkazujúci na tabuľku `recipes`.

Je prepojená vzťahom N:1 s tabuľkou `recipes`.

- **fermentation_type:** tabuľka, do ktorej budeme ukládať časť suroviny pre fermentáciu z receptu. Má tieto atribúty:
  - **name:** textový reťazec, označuje názov kvasiniek, poprípade produktu pre podporu fermentácie.
  - **amount:** textový reťazec, označuje množstvo.
  - **grams:** `boolean`, označuje či je množstvo v gramoch.
  - **ml:** `boolean`, označuje či je množstvo v mililitroch.
  - **recipe_id:** cudzí kĺúč, odkazujúci na tabuľku `recipes`.

Je prepojená vzťahom N:1 s tabuľkou `recipes`.

5.2 Serverová časť

5.2.1 Architektúra

Jednotlivé časti budeme kvôli poriadku pri vývoji rozdeľovať do adresárov. Do adresára `api` umiestníme koncové body REST API 5.1.1. V adresári `routes` sa budú nachádzať cesty k jednotlivým koncovým bodom. Do adresára `schema` umiestníme vstupné, výstupné, alebo pomocné objekty. Do adresára `service` umiestníme servisové služby pre jednotlivé koncové body, ktoré budú manipulovať s datami. V súbore `database.py` vytvoríme reláciu s databázou. Súbor `main.py` bude slúžiť ako hlavný súbor. V `models.py` vytvoríme objekty, ktoré sa budú vkládať do databázy.

5.2.2 API koncové body

Na začiatku každého každého súboru obsahujúceho API koncové body v adresári api zadefinujeme smerovač kvôli lepšiemu logickému usporiadaniu koncových bodov.

```
router = APIRouter(
    tags=["Recipe"],
    responses={404: {"description": "Not found"}},
)
```

V ukážke kódu vyššie môžeme vidieť definovaný smerovač pomocou triedy `APIRouter` z modulu `fastapi`. Tento objekt predstavuje smerovač pre definovanie tras a operácii v rámci API súvisiacich s receptami. Parametre `tags` a `responses` budú vytvárať metadata pre tento smerovač. Parameter `tags` slúži pre vytvorenie logickej skupiny. Parameter `responses` definuje rôzne odpovede, ktoré môže smerovač vrátiť. V príklade je definovaná odpoveď pre prípad, že zdroj nie je nájdený (404 - Not found).

Je nutné to potom zapísať ako cestu do súboru `routes.py` následovne:

```
from fastapi import APIRouter
from api import oAuth2Api, productApi, recipeAPi, \
    fermentationApi, favoriteRecipesApi, boilingApi

router = APIRouter()
router.include_router(oAuth2Api.router)
router.include_router(productApi.router)
router.include_router(recipeAPi.router)
router.include_router(fermentationApi.router)
router.include_router(favoriteRecipesApi.router)
router.include_router(boilingApi.router)
```

Ako koncové body API využijeme HTTP metódy spomenuté vyššie. Nad funkciou vytvoríme anotáciu smerovača a skrz ňu budeme volať príslušnú HTTP metódu. Do argumentu HTTP metódy vložíme cestu vo forme textového reťazca, ktorá slúži ako označenie príslušného koncového bodu API. V ceste sa môže vyskytovať aj parameter cesty, ktorý sa vkláda do kučeravých zátvoriek a zapisuje do argumentu funkcie, nad ktorou sa anotácia nachádza. Do argumentu funkcie pridávame buď požadovaný parameter z cesty, požadované telo vo forme objektu, alebo napríklad závislosť na databáze ako je vidno v kóde nižšie, poprípade bezpečnosť 5.2.4. Až na niektoré výnimky, v tele funkcie vraciame len volanie funkcie, ktorá je implementovaná v súboroch v priečinku service [34].

```
@router.get("/recipe/all")
async def get_recipes_list(db: Session = Depends(get_db)):
    return list_of_all_recipes(db)
```

Dostupné koncové body API si rozdelíme do logických skupín na koncové body pre autorizáciu, koncové body pre produkt a koncové body pre recepty.

	HTTP metóda	Cesta
1.	POST	/token
2.	POST	/register
3.	GET	/users/me

Tabuľka 5.1. Používané koncové body API pre autorizáciu.

V tabuľke 5.1 **1. koncový bod** slúži na získanie prístupového tokenu. Používateľ odosiela svoje prihlasovacie údaje na túto cestu, a ak sú údaje správne, server vráti prístupový token, ktorý sa následne používa na autentifikáciu pri volaní ďalších koncových bodov. **2. koncový bod** slúži na registráciu nových používateľov. Používateľ odosiela svoje registračné údaje (používateľské meno, heslo) na túto cestu a server vytvorí nový používateľský účet. **3. koncový bod** slúži na získanie informácií o prihlásenom používateľovi. Používateľ odosiela prístupový token ako súčasť požiadavky a server vráti informácie o používateľovi, jeho používateľské meno, e-mail a iné relevantné údaje.

	HTTP metóda	Cesta /product
1.	GET	/all/`{user_id}`
2.	POST	/add
3.	DELETE	/delete/`{product_id}`
4.	POST	/online/`{product_id}`
5.	POST	/boiling
6.	POST	/actual_temperature
7.	GET	/actual_temperature/`{product_id}`
8.	GET	/get_boil_status/`{product_id}`
9.	POST	/finish/`{product_id}`
10.	GET	/get_heat_temperature/`{product_id}`
11.	POST	/stop_boiling/`{product_id}`

Tabuľka 5.2. Používané koncové body API pre produkt.

V tabuľke 5.2 **1. koncový bod** slúži na získanie všetkých produktov pre daného používateľa. Používateľské ID je súčasťou cesty a server vráti zoznam všetkých produktov patriacich danému používateľovi. **2. koncový bod** slúži na pridanie nového produktu. Používateľ odosiela informácie o produkte v rámci požiadavky POST a server pridá tento produkt do databázy. **3. koncový bod** slúži na odstránenie produktu. Používateľ odosiela ID produktu ako súčasť cesty a server odstráni daný produkt z databázy. **4. koncový bod** slúži na označenie produktu ako online. Používateľ odosiela ID produktu ako súčasť cesty a server nastaví príznak online pre daný produkt. **5. koncový bod** slúži na označenie začatia varu. **6. koncový bod** slúži na aktualizáciu teploty. Kotol odosiela požiadavku POST s aktuálnou teplotou a server aktualizuje teplotu v kotli. **7. koncový bod** slúži na získanie aktuálnej teploty v kotli. Používateľ odosiela ID produktu ako súčasť cesty a server vráti aktuálnu teplotu. **8. koncový bod** slúži na získanie stavu varu daného produktu. Používateľ odosiela ID produktu ako súčasť cesty a server vráti informácie o stave varu. **9. koncový bod** slúži na ukončenie časti procesu varu daného produktu. Používateľ odosiela ID produktu ako súčasť cesty. **10. koncový bod** slúži na získanie teploty ohrevu daného produktu. Používateľ odosiela ID produktu ako súčasť cesty a server vráti o teplotu ohrevu. **11. koncový bod** slúži na zastavenie varu. Používateľ odosiela ID produktu ako súčasť cesty.

	HTTP metóda	Cesta /recipe
1.	GET	/all/`{user_id}`
2.	GET	/all
3.	GET	/get/`{recipe_id}`
4.	POST	/add
5.	DELETE	/delete/`{recipe_id}`

Tabuľka 5.3. Používané koncové body API pre recepty.

V tabuľke 5.3 **1. koncový bod** slúži na získanie všetkých receptov pre konkrétneho používateľa. Používateľské ID je súčasťou cesty a server vráti zoznam všetkých receptov patriacich danému používateľovi. **2. koncový bod** slúži na získanie všetkých receptov. Používateľ odosiela GET požiadavku a server vráti zoznam všetkých dostupných receptov. **3. koncový bod** slúži na získanie konkrétneho receptu. Používateľ odosiela ID receptu ako súčasť cesty a server vráti informácie o danom recepte. **4. koncový bod** slúži na pridanie nového receptu. Používateľ odosiela informácie o recepte (napríklad názov, ingrediencie, postup) v rámci požiadavky POST a server pridá tento recept do databázy. **5. koncový bod** slúži na odstránenie receptu. Používateľ odosiela ID receptu ako súčasť cesty a server odstráni daný recept z databázy.

5.2.3 Servisy

Pod adresár `service` vytvoríme servisy, ktoré slúžia na ukládanie, premapovanie, vyberanie a vyhodnocovanie dat.

```
def list_of_author_recipes(db: Session, user_id):
    user_db = db.query(models.User)
        .filter(models.User.id == user_id).first()
    recipes_list = []
    for recipe in user_db.recipes:
        recipes_list.append(
            recipesListSchema.ListSchema(recipe_uuid=str(recipe.id),
                                         recipe_name=recipe.name,
                                         author=user_db.username,
                                         tag=recipe.tag, alcohol=recipe.alcohol,
                                         ibu=recipe.ibu))
    return recipes_list
```

Napríklad do tohto servisu žiadame v argumente databázovú reláciu a ID užívateľa. V databáze sa nám urobí výber, ktorý vyselektuje všetke recepty užívateľa a vloží ich do premennej `user_db`. Vytvorí sa nová premenná typu `list`. Iteráciou cez recepty v objekte `user_db` sa premapujú data do nového formátu a vložia do nového listu `recipes_list`. Ten sa následne vráti z funkcie.

5.2.4 Bezpečnosť

Na bezpečnosť využívame knižnicu OAuth2. Koncové body sú opísané v sekcii 5.2.2. Prebieha to následovne:

- Používateľ sa prihlási na koncový bod `/token` pomocou POST požiadavky a odošle svoje prihlasovacie údaje (meno a heslo) vo formáte `OAuth2PasswordRequestForm`.
- Server zavolaním funkcie `authenticate_user` overí, či zadané prihlasovacie údaje sú správne. Tá skontroluje, či používateľ existuje v databáze a či sa zadané heslo zhoduje s uloženým zakódovaným heslom.
- Ak sú prihlasovacie údaje platné, server vygeneruje prístupový token volaním funkcie `create_access_token`. Token obsahuje informácie o prihlásenom používateľovi a je podpísaný pomocou tajného kľúča `SECRET_KEY` a algoritmu `ALGORITHM`.
- Server vráti vygenerovaný prístupový token ako odpoveď na požiadavku na koncovom bode `/token`.
- Používateľ môže použiť tento prístupový token pre autentifikáciu na chránených koncových bodoch, ktoré vyžadujú autentifikáciu. Token je posielaný ako `Authorization` hlavička s hodnotou `Bearer` ⟨*access_token*⟩.
- Pri každej požiadavke na chránený koncový bod sa server najprv overí platnosť a integritu prístupového tokenu. Dekóduje token a skontroluje jeho podpis pomocou tajného kľúča `SECRET_KEY` a algoritmu `ALGORITHM`.
- Ak je token platný, server získa používateľa zo zakódovaných informácií v tokene volaním funkcie `get_current_user`.
- Ak používateľ existuje a je aktívny, server povolí prístup k chránenému koncovému bodu. V opačnom prípade server vráti chybovú odpoveď.
- Používateľ môže ďalej vykonávať operácie v rámci chráneného koncového bodu.

Celý proces zabezpečuje, že iba oprávnení používatelia s platným a správnym prístupovým tokenom majú prístup k chráneným častiam aplikácie [35].

```
@router.get("/users/me/")
async def read_users_me(user: UserInDB = Depends(get_current_user)):
    return {"user": user}
```

5.2.5 Modely

Pri modeloch využívame prístup ORM (Object-Relational Mapping). ORM je prístup, ktorý umožňuje mapovať objekty z objektovo orientovaného programovania na relačnú databázu a umožňuje jednoduchú manipuláciu s dátami medzi objektovým modelom a databázovým modelom.

ORM technológia zjednodušuje prácu s databázou tým, že eliminuje potrebu priameho písania SQL dotazov a namiesto toho poskytuje abstrakciu nad databázovými operáciami. Namiesto ručného vytvárania a vykonávania dotazov sa používajú objektovo orientované metódy a vlastnosti, ktoré sú viazané na modely definované v aplikácii.

ORM umožňuje definovať štruktúru a vzťahy medzi objektami pomocou tried a atribútov a následne tieto objekty a vzťahy ukladať, aktualizovať a načítavať z databázy bez potreby priamej práce s SQL dotazmi [36].

Popíšeme si model užívateľa, ktorý môžeme vidieť v kóde nižšie, ostatné modely sa implementujú obdobne.

Trieda `User` je model pre reprezentáciu používateľa v databáze. Obsahuje niekoľko atribútov, ktoré definujú štruktúru a vlastnosti používateľa [37].

- **__tablename__:** Tento atribút určuje názov tabuľky v databáze, do ktorej sa budú používatelia ukladať.
- **id:** Atribút `id` je primárny kľúč tabuľky a slúži na jednoznačnú identifikáciu používateľa.
- **username:** Atribút `username` je reprezentácia používateľského mena. Je nastavený ako unikátny, čo znamená, že každý používateľ má unikátne meno.
- **hashed_password:** Atribút `hashed_password` uchováva zašifrované heslo používateľa. Heslo sa ukladá v zašifrovanom formáte, čo zabezpečuje bezpečnosť a ochranu používateľských údajov.
- **date:** Atribút `date` je dátum vytvorenia používateľa. Ukladá sa vo formáte reťazca.
- **is_active:** Atribút `is_active` indikuje, či je používateľ aktívny alebo nie. Je nastavený na `boolean` hodnotu (`True` alebo `False`).
- **recipes:** Atribút `recipes` je vzťah k modelu `Recipe` a zabezpečuje vzájomnú asociáciu medzi používateľom a jeho receptami.
- **products:** Atribút `products` je vzťah k modelu `Product` a zabezpečuje vzájomnú asociáciu medzi používateľom a jeho produktami.
- **favorite_recipe:** Atribút `favorite_recipe` je vzťah k modelu Recipe prostredníctvom tabuľky `favorite_recipes`. Slúži na uchovávanie informácií o obľúbených receptoch používateľa.

```
class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String, unique=True)
    hashed_password = Column(String)
    date = Column(String)
    is_active = Column(Boolean, default=True)
    recipes = relationship("Recipe", back_populates="user")
    products = relationship("Product", back_populates="user")
    favorite_recipe = relationship("Recipe",
        secondary=favorite_recipes, back_populates="user_favorite")
```

5.2.6 Pripojenie do databázy

Po importovaní potrebných knižníc sa definuje premenná `DATABASE_URL`, ktorá obsahuje URL adresu databázy. Vytvára sa inštancia `Engine` pomocou funkcie `create_engine()`, kde sa ako parameter poskytuje `DATABASE_URL`. `Engine` slúži na pripojenie k databáze a vykonávanie dotazov. Vytvára sa trieda `SessionLocal` pomocou funkcie `sessionmaker()`, ktorá definuje, ako budú vytvárané relácie s databázou. V tejto konfigurácii sa nastavuje, aby relácie neboli automaticky potvrdené (`autocommit=False`) a aby sa zmeny neodosielali automaticky (`autoflush=False`). Vytvorenú reláciu k `Engine` objektu pripája `bind=engine`.

Celý kód vytvára základné prostredie pre pripojenie k databáze pomocou SQLAlchemy. Pomocou `Engine` a `SessionLocal` je možné vytvárať relácie s databázou a vykonávať dotazy na databázové tabuľky. Následne je možné definovať ORM modely pre tabuľky a pracovať s nimi v rámci aplikácie [27].

```
DATABASE_URL = "url"
engine = _sql.create_engine(DATABASE_URL)

SessionLocal = _orm.sessionmaker(autocommit=False,
                    autoflush=False, bind=engine)

Base = _declarative.declarative_base()
```

5.3 Klientská časť

5.3.1 Architektúra

Jednotlivé súbory pod adresárom `src` si rozložíme do viacerých adresárov z hľadiska logického rozloženia. Pod adresárom `assets` budeme pridávať multimediálne súbory ako obrázky, vektory a podobne. Pod adresárom `components` budeme vytvárať menšie časti, tzv. komponenty užívateľského rozhrania, ktoré budeme následne skládať. Pod adresárom `router` budeme vytvárať cesty k jednotlivým stránkam. A pod adresárom `views` budeme vytvárať jednotlivé stránky nášho portálu.

5.3.2 Vloženie vylepšení

Po vygenerovaní projektu si ako prvé pridáme CSS rozšírenie Tailwind a knižnicu PrimeVue do hlavného JavaScript súboru projektu, pod adresárom css. Pridanie týchto súborov nám uľahčí prácu v tom, že nebude potrebné si písať vlastné štýly a vytvárať niektoré komponenty. Pôjde ich jednoducho z knižníc použiť. Pridáme si aj knižnicu `router`, ktorá nám pomôže určovať cesty jednotlivých stránok [38].

```
import { createApp } from 'vue'
import App from './App.vue'
import PrimeVue from "primevue/config";
import router from "@/router/router";
import "./index.css"
import "primevue/resources/themes/saga-blue/theme.css";
import "primevue/resources/primevue.min.css";
import  "primeicons/primeicons.css";

const app = createApp(App)
app.use(PrimeVue)
app.use(router)
app.mount('#app')
```

5.3.3 Stránky a komponenty

Stránky a komponenty majú rovnakú štruktúru, preto ich popíšeme v jednej sekcii. Stránky sú nadradené komponentom. Stránky označujeme kľúčovým slovom View v názve a komponenty kľúčovým slovom Component. Stránky, rovnako ako komponenty sa skládajú z 3 hlavných častí, ktoré tvoria párové značky:

- ⟨**template**⟩⟨**/template**⟩: Táto časť obsahuje HTML kód, ktorý definuje štruktúru a zobrazenie komponentu. Tu sa nachádza vykresľovanie obsahu komponentu.
- ⟨**script**⟩⟨**/script**⟩: Táto časť obsahuje JavaScript kód, ktorý definuje logiku a funkcie komponentu. Tu sa nachádzajú napríklad metódy, dáta a životné cykly komponentu.
- ⟨**style**⟩⟨**/style**⟩: Táto časť obsahuje CSS štýly pre daný komponent. Tu sa nachádzajú pravidlá a štýly pre vykresľovanie komponentu.

Ak chceme implementovať komponent do stránky, musíme ho najprv vložiť v časti, kde definujeme logiku a funkcie komponentu, potom ho môžeme použiť ako HTML značku. Do komponentu môžeme pridávať aj vstupné data tak, že v definícii komponentu zadefinujeme parametre v sekcii `props`. Napríklad, pre komponentu `CardComponent` definujeme parametre takto:

```
export default {
  name: "CardComponent",
  props: ['name','author','tag','alcohol','ibu']
}
```

Komponenty je potrebné pridať do stránky a to tak, že ich vložíme do časti, ktorá definuje logiku a funkcie a pridáme medzi komponenty [39].

```
import CardComponent from "@/components/CardComponent";
export default {
  components: {
    CardComponent,
  },
}
```

Ďalej ho môžeme používať v HTML časti, napríklad v kóde nižšie aj s parametrom `name`.

```
⟨CardComponent :name=”item.recipe_name”⟩⟨/CardComponent⟩
```

5.3.4 Cesty

V rámci routovacieho systému Vue.js môžeme definovať a konfigurovať cesty. Pri navigácii v aplikácii a dosiahnutí určitej URL cesty sa zobrazí priradená stránka a môžu sa vykonať prípadné ďalšie akcie alebo overenia na základe definovaných metadát.

```
{
        path: "/recept",
        name: "detail-receptu",
        component: RecipeDetailView,
        props: true,
        query: {
            id: {
                type: Number,
                required: true
            },
            author: {
                type: String,
                required: true
            }
        },
        meta: { requiresAuth: true },
    },
```

Na kóde vyššie si vysvetlíme jednotlivé položky [40]:

- **path:** Táto časť definuje cestu, na ktorej bude dostupný detail receptu. V tomto prípade je cesta nastavená na `/recept`.
- **name:** Každá cesta vo frameworku alebo knižnici pre smerovač má pridelené meno, ktoré sa používa na odkazovanie na túto cestu z iných častí kódu. Tu je pridelené meno `detail-receptu` pre cestu `/recept`.
- **component:** Táto časť určuje komponent, ktorý sa má zobraziť, keď je používateľ na ceste `/recept`. V tomto prípade je komponentom `RecipeDetailView`, čo naznačuje, že tento komponent bude zodpovedný za zobrazenie detailu receptu.
- **props:** Táto vlastnosť definuje, či sa parametre smerovača automaticky prenášajú ako vstupy do komponentu. V tomto prípade je nastavená na `true`, čo znamená, že parametre smerovača (`id` a `author`) budú automaticky prenášané do komponentu `RecipeDetailView`.
- **query:** Táto časť definuje očakávané parametre smerovača a ich typy. V tomto prípade sú očakávané dva parametre: `id` a `author`. `id` musí byť číselného typu (`Number`) a je povinný (`required: true`), zatiaľ čo `author` musí byť reťazcového typu (`String`) a je tiež povinný.
- **meta:** Táto časť definuje dodatočné metaúdaje pre túto cestu. Tu je definovaný `requiresAuth` s hodnotou `true`, čo znamená, že na prístup k tomuto zobrazeniu je vyžadovaná autentifikácia používateľa. To znamená, že používateľ musí byť prihlásený, aby mohol zobraziť detail receptu.

5.3.5 Štýlovanie pomocou TailWind a použite PrimeVue

Štýlovanie prebieha tak, že sa zapisujú jednotlivé triedy TailWindu do tried HTML značky.

```
⟨div class="flex flex-col w-[50%] pl-[3%] text-center"⟩ ⟨/div⟩
```

Napríklad v kóde vyššie môžeme vidieť HTML značku `div`, ktorá podľa pridaných tried TailWindu hovorí, že potomkovia budú mať usporiadanie `flex` a `flex-col`, čo znamená, že budú zoradené v stlpci, `w-[50%]` nám hovorí, že šírka bude 50 percent, `pl-[3%]` znamená vnútorné odsadenie o 3 percentá a text-center znamená, že text v kontajneri bude vycentrovaný. Jednotlivé triedy sú dostupné na fóre TailWindu [41].

Komponenty z PrimeVue sa používajú podobne ako ako komponenty vytvorené nami v projekte 5.3.3.

5.3.6 Zasielanie požiadavok na server

Zasielanie požiadavok na server si popíšeme na následujúcom kóde.

```
const userId = localStorage.getItem('userId');
      const oauth2Token = localStorage.getItem('authToken');
      if (this.selectedType.key == 'all') {
        this.$http.get('/recipe/all/', {
          headers: {
            'Authorization': `Bearer ${oauth2Token}`,
          }
        }).then((response) => {
          this.items = response.data;
          console.log(this.items)
        });
      }
```

Premenné `userId` a `oauth2Token` sa získavajú z lokálneho úložiska prehliadača pomocou metódy `getItem` na objekte `localStorage`.

Následne sa vykonáva podmienka `if`, ktorá kontroluje, či hodnota `key` objektu `selectedType` je rovná reťazcu `'all'`. Ak áno, vykoná sa nasledujúci blok kódu.

V tomto bloku sa pomocou `this.$http.get` vykonáva HTTP GET požiadavka na adresu `'/recipe/all/'`. Do tejto požiadavky sa pridáva hlavička `Authorization`, ktorá obsahuje hodnotu `Bearer` a `oauth2Token`.

Keď je odpoveď na túto požiadavku dostupná, spracuje sa v bloku then. Dáta z odpovede sa priraďujú do premennej items, a následne sa tieto dáta vypíšu do konzoly pomocou `console.log(this.items)`.

Celkovo, tento kód získava `userId` a `oauth2Token` z lokálneho úložiska prehliadača a v prípade, že `selectedType.key` je `'all'`, vykonáva HTTP GET požiadavku na danú URL s autentifikačným tokenom a spracúva odpoveď [42].

5.4 Kotol

5.4.1 Pripojenie na Wi-Fi

Na začiatku sa nastaví komunikácia cez sériovú linku s rýchlosťou prenosu 115200 baudov pomocou `Serial.begin(115200)`. Týmto sa otvorí spojenie medzi modulom a sériovým monitorom, čo umožňuje výstupné a vstupné správy.

Po inicializácii sériovej komunikácie nasleduje oneskorenie na 1 sekundu pomocou `delay(1000)`. Toto oneskorenie slúži na zabezpečenie, aby sa modul mohol inicializovať a pripraviť na ďalšie kroky.

Ďalej sa nastavuje režim WiFi pripojenia pomocou `WiFi.mode(WIFI_STA)`. Tým sa modul nastavuje ako klient, čo znamená, že bude hľadať existujúcu WiFi sieť, ku ktorej sa bude pripájať.

Používa sa `WiFi.begin(ssid, password)`, aby sa inicializovalo pripojenie k WiFi sieti s daným SSID (názvom siete) a heslom. Modul sa pokúsi pripojiť k zadanému WiFi smerovaču.

Výpisom správy `Connecting` pomocou Serial.println(`\nConnecting`) sa informuje o tom, že modul sa pokúša pripojiť k WiFi sieti. Táto správa sa zobrazí v sériovom monitore.

Nasleduje cyklus `while(WiFi.status() != WL_CONNECTED)`, ktorý sa opakuje, kým modul nie je úspešne pripojený k WiFi sieti. V rámci tohto cyklu sa kontinuálne kontroluje stav pripojenia pomocou `WiFi.status()`. Ak stav nie je `WL_CONNECTED` (tj. nie je úspešné pripojenie), cyklus sa opakuje.

V cykle sa vypisuje bodka `Serial.print(.)` na sériový monitor, aby sa užívateľovi ukázalo, že modul sa stále snaží pripojiť. Čakanie sa oneskoruje o 100 milisekúnd pomocou funkcie s argumentom `delay(100)`.

Keď je stav pripojenia `WL_CONNECTED`, teda modul je úspešne pripojený k WiFi sieti, vypíše sa správa `Connected to the WiFi network`.

Nasleduje výpis IP adresy pridelenej modulu v rámci siete.

Nakoniec sa nastavuje stav výstupného pinu `digitalWrite(wifiLed, HIGH)`, napríklad LED indikujúcej pripojenie k WiFi sieti [43].

```
Serial.begin(115200);
delay(1000);

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
Serial.println("\nConnecting");

while(WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(100);
    conected = true;
}

Serial.println("\nConnected to the WiFi network");
Serial.print("Local ESP32 IP: ");
Serial.println(WiFi.localIP());
digitalWrite(wifiLed, HIGH);
```

5.4.2 Prevod čísel a vyzobrazenie na obrazovku

Na začiatku sú deklarované potrebné premenné, ako sú `temperature_digits`, `timer_digits`, `temperature_value`, `minutes` a `seconds`. Tieto premenné slúžia na uchovávanie hodnôt teploty a času.

Nasledujúce riadky kódu slúžia na rozdelenie hodnoty teploty na jednotlivé číslice. Teplota je uložená v premennej `temperature_value`. Používa sa delenie a operátor modulo (`%`) na získanie jednotlivých číslic. Napríklad riadok `temperature_digits[0] = int(temperature_value / 100) % 10` získava prvú číslicu teploty. Teplota je delená 100, aby sa získalo stonásobok teploty, a potom sa používa operátor modulo 10, aby sa získala jednotlivá číslica.

Podobným spôsobom sa rozdeľuje aj čas na jednotlivé číslice. Premenné minutes a seconds obsahujú hodnoty minút a sekúnd.

Nasledujú dve podmienkové konštrukcie if, ktoré kontrolujú, jedna kontroluje či je menšia ako 100 a druhá, či je menšia ako 10 a podľa toho prispôsobia zobrazenie.

Ďalšie riadky kódu priradzujú hodnoty jednotlivým segmentom displeja pre zobrazenie čísel. Používajú sa indexy `temperature_digits` a `timer_digits` na získanie správnej číslice a priradenie jej k príslušnému segmentu displeja. Napríklad `eink_segments[0] = digit_right[temperature_digits[0]]` priradzuje hodnotu pravého segmentu pre prvú číslicu teploty zo `temperature_digits` do `eink_segments`.

5.4.3 Časovač a nastavenie času

Na časovač je využívaná funkcia `millis()`. Princíp algoritmu je následovný:

- Na začiatku je potrebné deklarovať premenné pre správne fungovanie časovača. Najdôležitejšou premennou je premenná `previousMillis`, ktorá bude uchovávať hodnotu času z predchádzajúceho cyklu `loop()`.
- V hlavnom cykle `loop()` sa kontinuálne kontroluje uplynutý čas od posledného spustenia časovača. To sa robí porovnaním aktuálneho času získaného pomocou `millis()` s hodnotou `previousMillis`.
- Ak uplynulý čas prekročí určitý interval (napríklad 1000 ms pre 1 sekundový časovač), vykoná sa požadovaná akcia. To môže zahŕňať vykonanie nejakej úlohy, zmenu stavu pinu alebo vyvolanie inej funkcie.
- Po vykonaní akcie sa aktualizuje hodnota previousMillis na aktuálny čas pomocou `millis()`, čím sa vytvorí nový referenčný bod pre ďalší interval.

Týmto spôsobom sa cyklus opakuje a časovač pokračuje v správnom odpočítavaní času a vykonávaní akcií v zadaných intervaloch [44].

5.4.4 Algoritmus na varenie bez Wi-Fi

Algoritmus je navrhnutý tak, aby umožňoval jednoduché ovládanie ohrevu pomocou tlačidiel a zobrazoval aktuálne hodnoty teploty a času na displeji. Taktiež je vybavený softvérovou reguláciou.

Po spustení programu mikrokontrolér začne nekonečný cyklus, ktorý prebieha, kým nedôjde k prerušeniu. V každej iterácii cyklu sa najprv zisťuje stav tlačidla backButton, ktoré slúži na spustenie procesu ohrevu.

Ak je tlačidlo stlačené (jeho stav je `LOW`), program vykoná oneskorenie (delay) počas 2 sekúnd. Toto oneskorenie umožňuje užívateľovi dostatok času na pridržanie tlačidla. Po oneskorení sa na sériovú linku (Serial) posiela správa `Začína ohrev`, čo informuje o spustení procesu ohrevu. Následne sa zisťuje aktuálna teplota pomocou senzorov a hodnota sa ukladá do premennej temperature_value.

Ak je tlačidlo backButton opäť stlačené a zároveň predtým nebolo ešte stlačené, program vykoná ďalšie kroky. Opäť sa vykoná oneskorenie (`delay`) počas 2 sekúnd. Následne sa volá funkcia `set_time()`, ktorá nastavuje aktuálny čas. Premenná `pressed` je nastavená na `true`, čo značí, že tlačidlo bolo stlačené a hodnota teploty sa uloží do premennej set_temperature. Na sériovú linku sa posiela správa `Uložená teplota`, čím sa informuje o uložení nastavenej teploty.

Ak je stlačené tlačidlo endButton (určené na ukončenie ohrevu), program vykoná nasledujúce kroky. Pin `heatLed` sa nastaví na `LOW`, čo spôsobí vypnutie LED indikátora ohrevu. Premenná `interupt` je nastavená na `true`, čo znamená, že došlo k prerušeniu cyklu.

Ak je aktuálna teplota vyššia ako nastavená teplota plus 2 a premenná pressed je nastavená na `true` (t.j. nastavená teplota už bola uložená), na sériovú linku (Serial) sa posiela správa `Stop ohrevu`, čo signalizuje, že ohrev sa zastavuje a pin `heatLed` sa nastaví na `LOW`, čo vypne LED indikátor ohrevu.

Týmto spôsobom je zabezpečené, že ak aktuálna teplota prekročí nastavenú hodnotu o 2 stupne Celzia, ohrev sa zastaví a uživateľ je o tom informovaný cez sériovú linku a LED indikátor.

Po týchto krokoch algoritmus pokračuje v meraní teploty, aktualizácii displeja a opätovnom overovaní podmienok v cykle, kým nedôjde k prerušeniu (napríklad stlačením tlačidla `endButton`).

5.4.5 Algoritmus na varenie s Wi-Fi

Na začiatku kódu sa nastavujú premenné `minutes` a `seconds` na hodnotu 0. Následne sa deklarujú premenné `get_temperature` a `get_time`, ktoré slúžia na uchovávanie hodnôt teploty a času získaných zo servera. Potom sa volajú funkcie `connect_wifi()` a `set_status()`, ktoré inicializujú WiFi pripojenie a nastavia stav (status) systému.

Nasleduje nekonečný cyklus s podmienkou `while (condition)`, čo znamená, že sa kód v tomto cykle vykonáva opakovane, pokiaľ je premenná `condition` nastavená na hodnotu `true`.

V tomto cykle sa najprv vykoná HTTP GET požiadavka na zadanú URL adresu, cez ktorú sa získajú teplota a čas vo formáte JSON. Odpoveď servera je kontrolovaná pomocou premenných `httpResponseCode` a `payload`.

Ak je odpoveď servera s kódom 200 (HTTP_OK), čo znamená, že požiadavka bola úspešná, tak sa získané hodnoty teploty a času spracujú. Používa sa knižnica ArduinoJson na deserializáciu JSON reťazca a prístup k jednotlivým hodnotám teploty a času.

Po spracovaní JSON dát sa vykonáva riadenie ohrevu. Najprv sa nastaví výstupný pin `heatLed` na `HIGH`, čo spustí ohrev. Potom nasleduje vnorený cyklus s podmienkou `while (boil)`, ktorý čaká, kým sa senzorová teplota dosiahne alebo presiahne hodnotu `get_temperature`. Keď sa táto podmienka splní, cyklus sa ukončí a ohrev sa zastaví nastavením výstupu `heatLed` na `LOW`.

Po zastavení ohrevu nasleduje ďalší vnorený cyklus, ktorý udržuje teplotu v rozmedzí +-2 stupne. V tomto cykle sa kontinuálne číta senzorová teplota a porovnáva sa s hodnotou `get_temperature` (požadovaná teplota). Ak je senzorová teplota vyššia ako `get_temperature` plus 2 stupne, ohrev sa zastaví nastavením výstupu `heatLed` na `LOW`. Ak je senzorová teplota nižšia ako `get_temperature` mínus 2 stupne, ohrev sa znovu spustí nastavením výstupu `heatLed` na `HIGH`. Tento cyklus sa opakuje, pokiaľ premenná `hold_temperature` je nastavená na hodnotu `true`.

V rámci cyklu sa tiež aktualizuje čas na displeji pomocou funkcie `millis()`, ktorá meria uplynutý čas od spustenia programu. Ak uplynulý čas prekročí určený interval (premenná `interval`), vykonajú sa nasledujúce kroky:

- Ak je premenná `pressed` nastavená na hodnotu `true`, volá sa funkcia `set_time()`, ktorá nastavuje čas na základe stlačenia tlačidla.
- Vykoná sa aktualizácia displeja a zobrazia sa teplotné údaje a čas.
- Vykonajú potrebné operácie na obnovenie obrazovky.
- Pomocou funkcie HTTP POST odosiela aktuálna teplota na server. Vytvára sa JSON objekt obsahujúci identifikátor produktu a aktuálnu teplotu, ktorý sa odosiela ako payload(užitočné dáta).
- Odpoveď servera sa kontroluje a ak je s kódom 200, tak je nastavená aktuálna teplota na server.

Po skončení vnorených cyklov sa volá funkcia set_status(), ktorá aktualizuje stav systému.

Kapitola 6
Použitie

6.0.1 Použitie bez WiFi

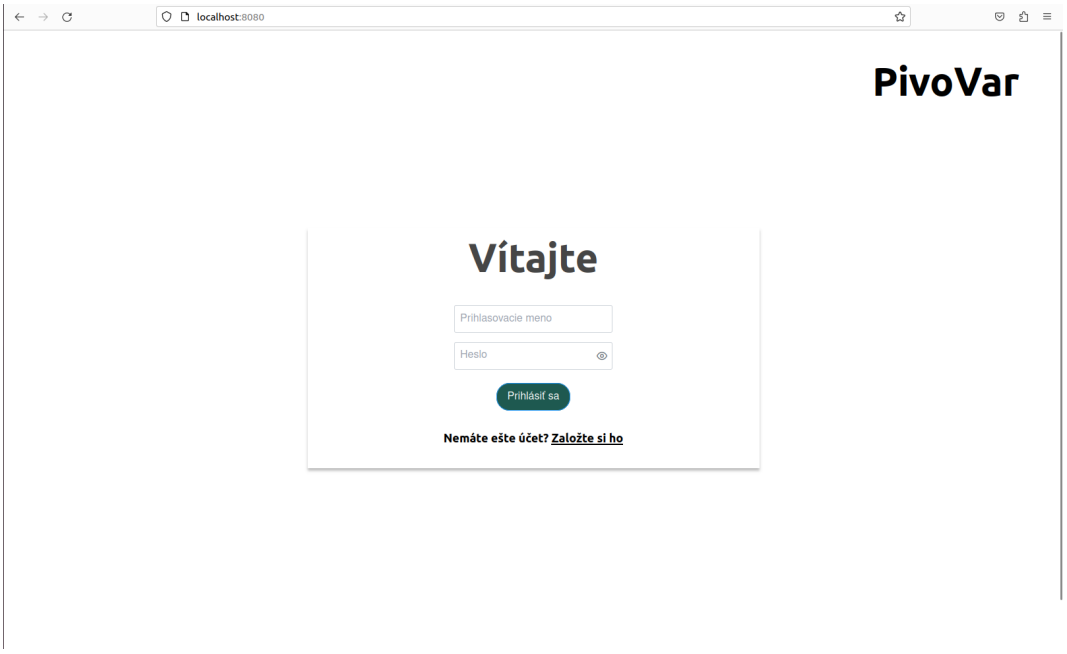
Použitie bez WiFi je veľmi jednoduché. Stačí napájať kotol sieťovým napájaním. Po zaklíknutí tlačidla OK sa začne ohrev. Ak sa dosiahne nami požadovaná teplota, tak po opätovnom zakliknutí tlačidla OK sa teplota uloží a kotol sa bude držať na rovnakej teplote s toleranciou dvoch stupňov, spustí sa časovač. Postlačení tlačidla UKONčIŤ sa celý proces ukončí a po jeho opätovnom stlačení sa premaže obrazovka.

6.0.2 Spustenie klientskej a serverovej časti

Najjednoduchšie je, ak sú obe časti spoločne s databázou nasadené na verejnom serveri. V prípade lokálneho spustenia je nutné si nainštalovať na svoj počítač Docker, Python, Node.Js a pre dodatočný vývoj nejaké vývojové prostredie. Najprv je potrebné si spustiť Docker kontajner s databázou, ktorý sa nachádza medzi súborami v prílohách, následne je možné si spustiť serverovú a klientsku časť. Pre povolenie požiadavok na lokálnu adresu z kotla je potrebné si nastaviť vo windows firewall.

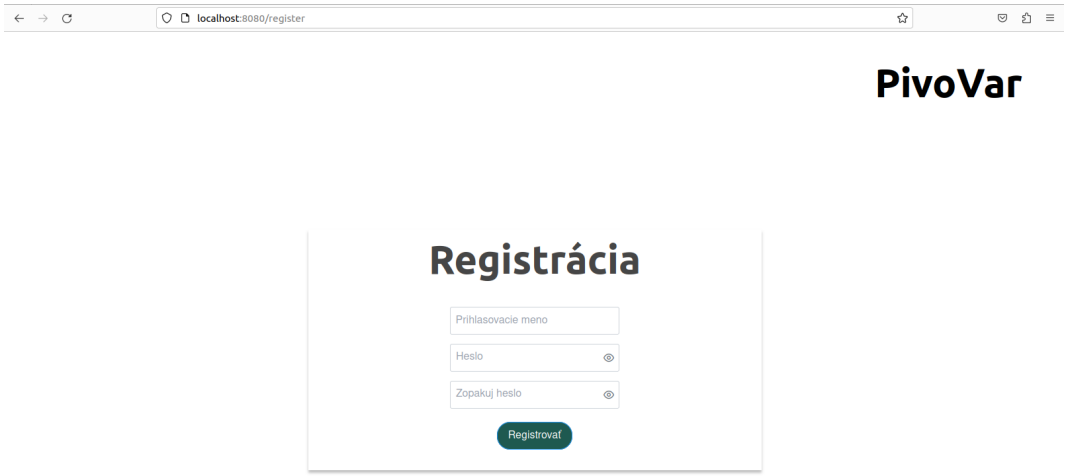
6.0.3 Ukážka portálu a použitie s WiFi

Portál začína prihlasovacou stránkou, kde je možnosť sa do svojho účtu prihlásiť, alebo si ho založiť. Po zakliknutí založiť je užívateľ presmerovaný na stránku s formulárom na registráciu 6.2. V prípade prihlásenia na stránku, kde je zoznam receptov 6.3.



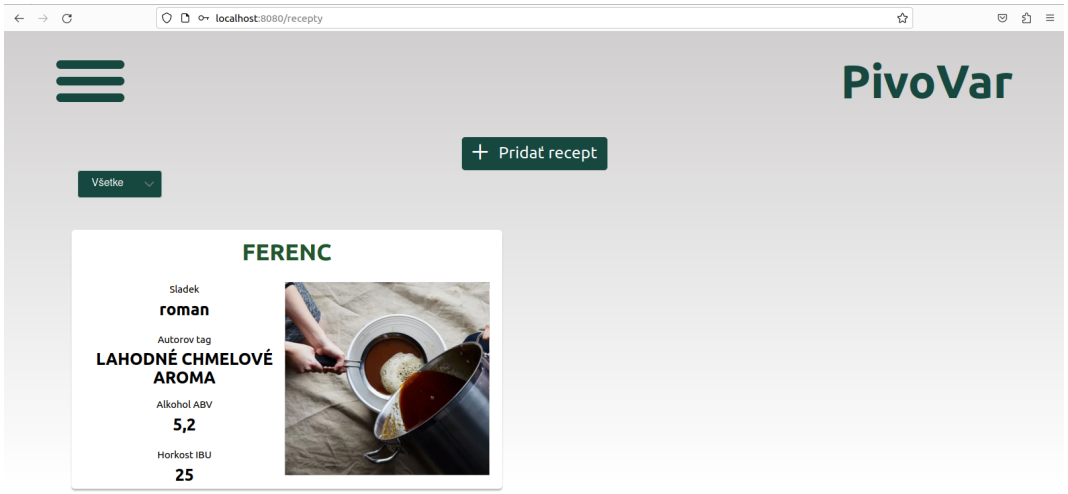
Obrázok 6.1. Ukážka prihlasovacej stránky portálu.

Po zakliknutí možnosti registrácia ste presmerovaní na formulár, kde je potrebné zadať uživateľské meno a opakovane heslo. Po zakliknutí tlačidla registrovať je užívateľ presmerovaný na prihlasovaciu stránku 6.1.



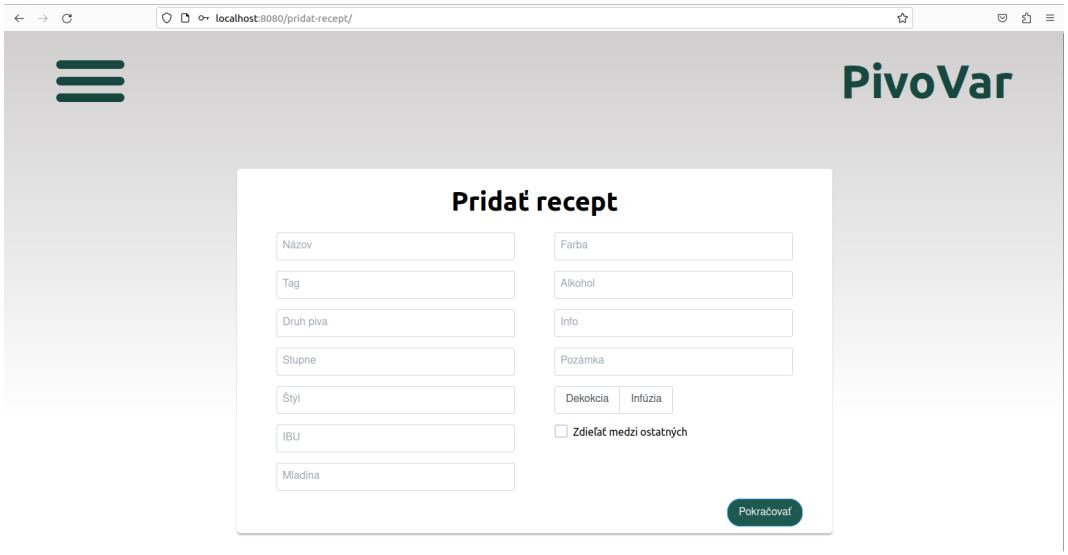
Obrázok 6.2. Ukážka registračnej stránky portálu.

Pri zozname receptov je možné si vybrať z možnosti medzi všetkými receptami a Vašimi receptami. V ľavom hornom rohu je možnosť zakliknutia panelu s menu 6.5. V strede obrazovky je tlačidlo pridať recept, ktoré presmeruje na stránku, kde sa vytvárajú recepty. Po kliknutí na recept je užívateľ presmerovaný na detail receptu.



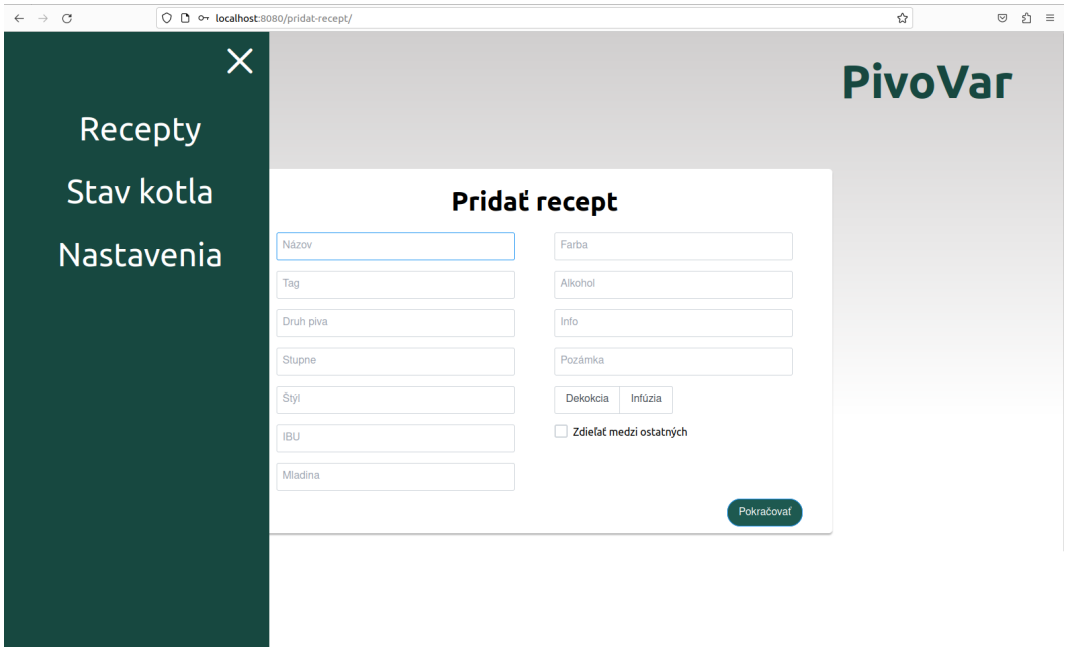
Obrázok 6.3. Ukážka úvodnej stránky so zoznamom receptov.

Stránka na pridavanie receptov obsahuje viacero formulárov s popismi. Po vyplnení stránka presmeruje užívateľa na zoznam receptov 6.3.



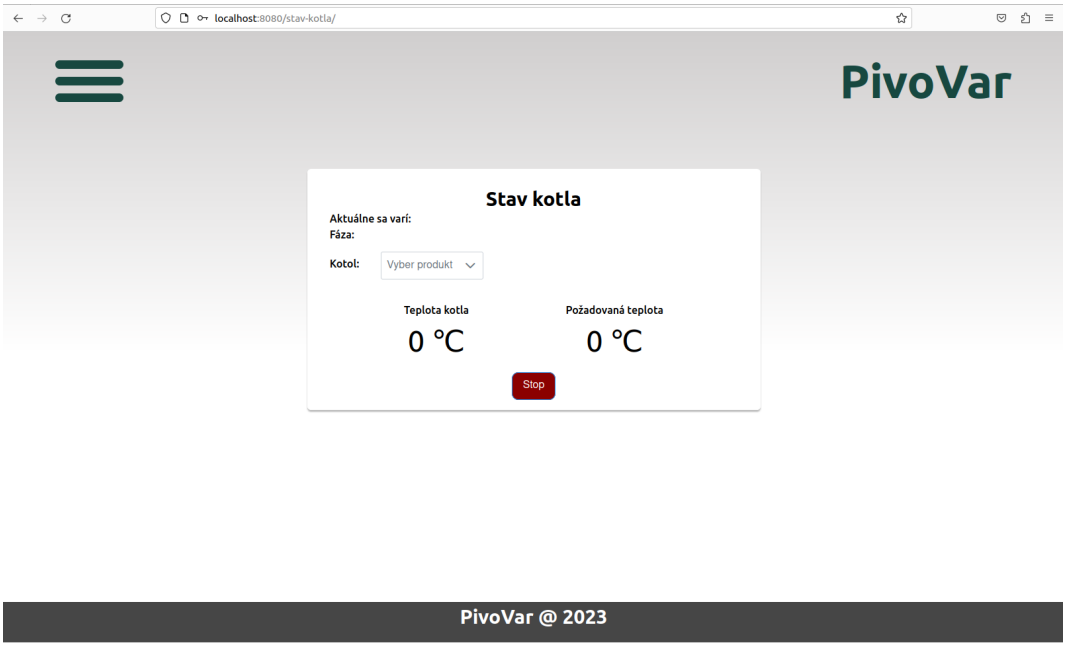
Obrázok 6.4. Ukážka stránky na pridávanie receptov.

Bočné navigačné menu 6.5 obsahuje možnosti presmerovania na recepty 6.3, stav kotla 6.6 a nastavenia 6.7.



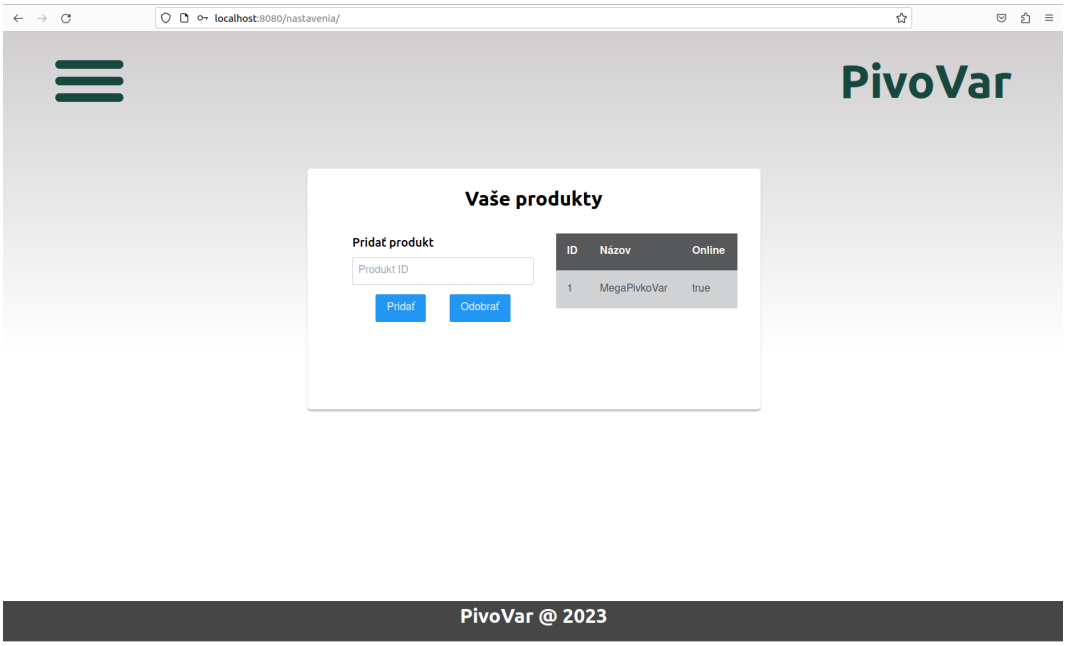
Obrázok 6.5. Ukážka navigačného menu.

V stránke so stavom kotla 6.6 máme na výber z aktuálnych produktov, pre ktorý chceme stav vidieť. Stránka zobrazuje aktuálny recept, ktorý sa varí, či prebieha rmutovanie, alebo chmeľovar, koľko stupňov je v kotli a aká je požadovaná teplota, ktorú chceme dosiahnuť. Je tam aj tlačidlo stop, ktorá zastaví varenie.



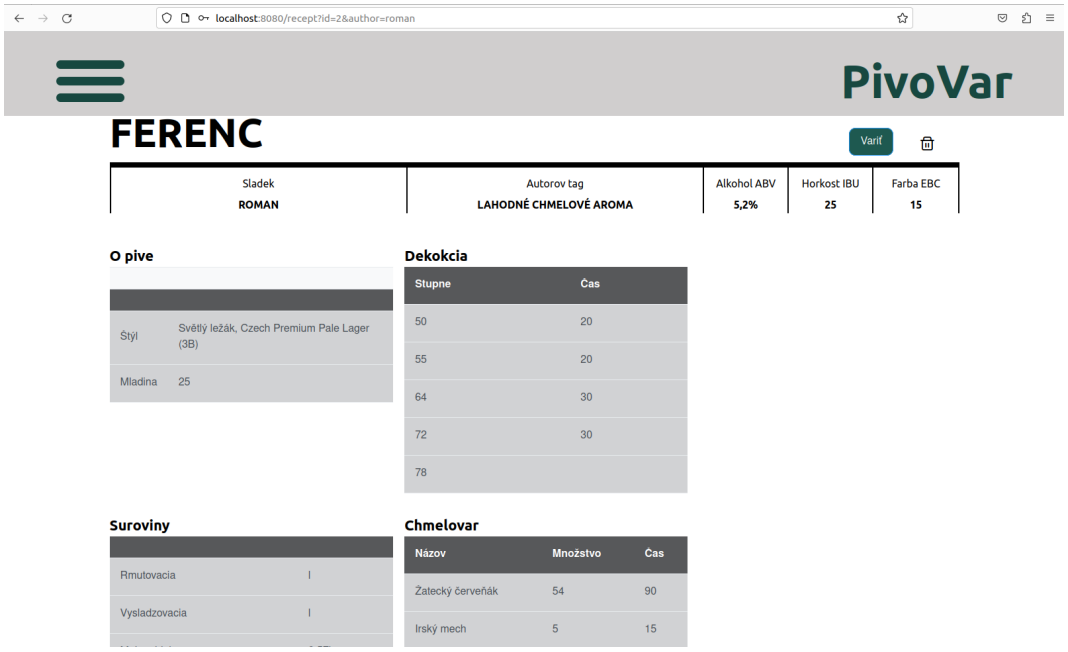
Obrázok 6.6. Ukážka stránky so stavom kotla.

V stránke so zoznamom produktov 6.7 vieme vidieť pridané produkty. Vieme pridať produkt a zároveň aj odobrať produkt.

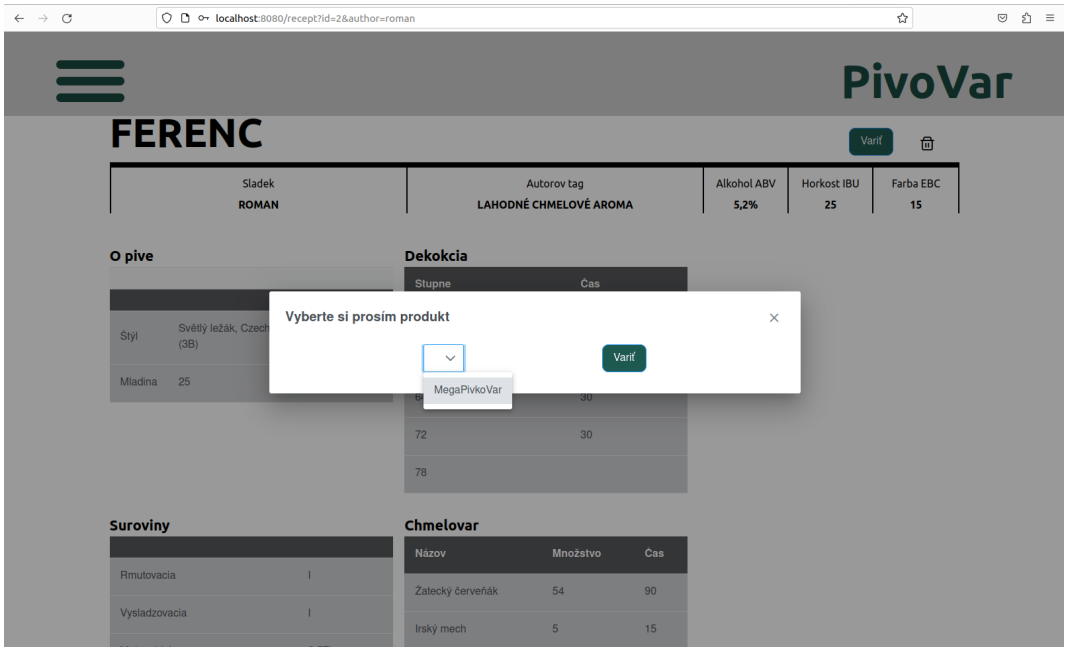


Obrázok 6.7. Ukážka stránky so zoznamom produktov.

V stránke s detailom receptu 6.8 máme prehľad o celom recepte. Medzi možnosťami máme zakliknúť tlačidlo variť, po ktorom nám naskočí dialogové okno 6.9, v ktorom si vieme vybrať aktuálny produkt a spustiť varenie, ak je produkt pripojený na WiFi. Zároveň je tam možnosť vymazania receptu v podobe koša v pravom hornom rohu.



Obrázok 6.8. Ukážka stránky s detailom receptu.



Obrázok 6.9. Ukážka stránky po zakliknutí variť v detaile receptu.

Celý proces funguje tak, že po zapnutí kotla je potrebné stlačiť tlačidlo WiFi. Po zatlačení sa kotol pripojí na WiFi a zašle požiadavku na server, že je online. Užívateľ sa prihlasí do portálu a vyberie si zo svojich pridaných receptov, poprípade, ak sú dostupné, z receptov od ostatných užívateľov. Potom si len otvorí detail receptu a zaklikne tlačidlo variť. Po zakliknutí sa otvorí dialogové okno, na ktorom je potrebné vybrať produkt a potvrdiť varenie. Zašle sa požiadavka na server, pri ktorej sa pridajú teploty a časy, na ktorých sa bude recept variť do tabuľky boiling. Užívateľ je presmerovaný na stránku status. Akonáhle dosiahne kotol požadovanú teplotu, tak sa spustí časovač na obrazovke. Po uplynutí času kotol zašle požiadavku na server, vymaže sa posledná teplota a čas a odošle sa ďalšia teplota a čas. Aktuálna teplota

sa potom zobrazuje na stránke stav kotla 6.6. Celé to pokračuje do času kým celý proces nepreruší užívateľ, alebo sa proces neukončí sám.

Klientskú časť by bolo potrebné v prvom rade optimalizovať, keďže momentálne nie je responzívna. Možno pridať možnosť videoreceptov, aby to bolo bližšie pre užívateľov a urobiť ju viac interaktívnu. Veľké plus by bola aj mobilná aplikácia.

Kapitola 7
Záver

V rámci tejto bakalárskej práce sme sa venovali téme domovarníctva a vývoju systému pre domáce varenie piva. Cieľom nášho projektu bolo vytvoriť komplexný systém, ktorý by umožnil pivovarníkom variť pivo v pohodlí vlastného domova a s pomocou moderných technológií.

V úvode práce sme sa zaoberali motiváciou pre túto tému a bližšie sme sa pozreli na obľúbenosť piva, históriu piva a domovarníctva, vývoj vo varení a súčasný stav domovarníctva. Následne sme popísali využité technológie, ktoré sme implementovali do nášho systému, ako napríklad IoT, serverovú a klientskú časť.

V časti implementácie sme detailne popísali architektúru serverovej a klientskej časti systému. Spomenuli sme jednotlivé technológie, ako REST API, Postgresql, FastAPI, SQLAlchemy, Vue.js, Tailwind a PrimeVue, ktoré sme využili pri vytváraní systému. Taktiež sme sa venovali implementácii kotla a jeho jednotlivým súčastiam. Na záver sme diskutovali o jeho použití. Jednotlivé časti boli aj zhotovené a sú dostupné v prílohách.

Identifikujeme niekoľko oblastí, v ktorých by sme mohli implementovať vylepšenia. Jedným z možných vylepšení je pridanie regulátora, ktorý by umožnil plynulejšie nastavovanie teploty pri varení piva, čím by sme dosiahli presnejšiu kontrolu nad procesom. Ďalším vylepšením by bolo vymeniť obrazovku za väčšiu, pokročilejšiu variantu s možnosťou čiastočného obnovovania, čo by umožnilo zobraziť viac informácií o procese varenia.

Okrem toho môžeme uvažovať o využití technológie LoRa pre prenos dát z kotla, čím by sme znížili závislosť systému na WiFi. Taktiež je možnosť pridať do systému automatické prečerpávanie či miešanie, čo by pivovarníkom uľahčilo proces varenia. Ak by sme sa rozhodli odstrániť zobrazovací panel priamo z kotla, mohli by sme vytvoriť samostatný informačný panel, ktorý by zobrazoval dôležité informácie o stave a postupe varenia.

V oblasti serverovej časti je možnosť rozšíriť systém o možnosť upravovania receptov a pridávania receptov medzi obľúbené. Tým by sme umožnili pivovarníkom individuálnu úpravu a prispôsobenie receptov ich preferenciám a chuťovým preferenciám. Rovnako možnosť vytvorenia komunikačnej platformy medzi sládkami, kde by si mohli vymieňať skúsenosti, rady a recepty, čo by prispelo k rozvoju komunity domácich pivovarníkov.

Klientskú časť by sme mohli ďalej optimalizovať a prispôsobiť pre responzívne zobrazenie, aby sa systém mohol pohodlne používať na rôznych zariadeniach, vrátane mobilných telefónov a tabletov. Taktiež môžeme uvažovať o pridaní možnosti

videoreceptov, ktoré by boli interaktívnym sprievodcom pri varení piva a zároveň by boli zrozumiteľnejšie pre používateľov. Veľkým prínosom by bola aj vytvorenie mobilnej aplikácie, čo by umožnilo pivovarníkom mať prístup k systému kedykoľvek a kdekoľvek, čím by sme zvýšili jeho dostupnosť a pohodlnosť.

Celkovo sme sa v rámci tejto bakalárskej práce zamerali na vývoj systému pre domáce varenie piva, ktorý využíva moderné technológie a prináša nové možnosti a pohodlie pre domácich pivovarníkov. Veríme, že naše navrhnuté vylepšenia ukážu ďalší možný rozvoj a posun v tejto oblasti a umožnia pivovarníkom ešte väčšiu kontrolu a tvorivosť pri varení piva.

Literatúra

[1] *Beer Industry Statistics. [online]. Truly Experiences.* [vid. 14.5.2023] . Dostupné z https://trulyexperiences.com/blog/beer-industry-statistics/.

[2] *The Complete History of Homebrew. [online]. MoreBeer!* [vid. 14.5.2023] . Dostupné z https://www.morebeer.com/questions/306.

[3] *Cuneiform Pictographs Recording the Allocation of Beer. [online]. Ancient Origins.* [vid. 14.5.2023] . Dostupné z https://www.ancient-origins.net/news-history-archaeology/sip-sumerian-ancient-beer-recipe-recreated-millennia-old-cuneiform-tablets-021492.

[4] *What is skunked beer? [online]. VinePair.* [vid. 14.5.2023] . Dostupné z https://vinepair.com/beer-101/what-is-skunked-beer/.

[5] *Brewing: Past, Present, and Future. [online]. Precision Fermentation.* [vid. 14.5.2023] . Dostupné z https://www.precisionfermentation.com/blog/brewing-past-present-and-future/.

[6] Petr Novotný. *Pivařka2: Průvodce domácího sládka: Teorie, rady, návody, Recepty.* Brno, Česká republika: Jota, 2019.

[7] *Recepty, které se nevešly. [online]. Český svaz pivovarů a sladoven.* [vid. 14.5.2023] . Dostupné z https://www.cech-pivo.cz/cs/pivar/downloads/recepty-ktere-se-nevesly-98yexd5?task=download.send&id=5&catid=2&m=0.

[8] *What is IoT? [online]. Oracle.* [vid. 14.5.2023] . Dostupné z https://www.oracle.com/internet-of-things/what-is-iot/.

[9] *ESP32 WROOM 32 Datasheet. [online]. Espressif.* [vid. 14.5.2023] . Dostupné z https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[10] *ESP32 DevKitC 38-pin ESP-WROOM-32 WiFi-BT Pinout. [online].* [vid. 14.5.2023] . Dostupné z https://cdn.myshoptet.com/usr/www.hwkitchen.cz/user/documents/upload/hwkitchen/esp32-devkitc-38pin-espwroom32-wifi-bt/esp32-devkitc-38pin-espwroom32-wifi-bt-pinout.jpg.

[11] *Waveshare. 1.9inch Segment e-Paper Module. [online]. Waveshare Wiki.* [vid. 14.5.2023] . Dostupné z https://www.waveshare.com/wiki/1.9inch_Segment_e-Paper_Module.

[12] *Waveshare 19.91-segmentový e-papierový displej s I2C - čierna/biela. [online]. RpiShop.cz.* [vid. 14.5.2023] .

Dostupné z https://rpishop.cz/26643-large_default/waveshare-19-91segmentovy-e-paper-displej-i2c-cernabila.jpg.

[13] *1Ch-relay.pdf [online]. Handsontec.com.* [vid. 14.5.2023] .
Dostupné z https://handsontec.com/dataspecs/relay/1Ch-relay.pdf.

[14] *1-kanálový 5V relé modul Tongling - 250VAC 10A. [online]. LaskaKit.cz.* [vid. 14.5.2023] .
Dostupné z https://www.laskakit.cz/1-kanal-5v-rele-modul-tongling--250vac-10a/?gclid=CjwKCAjwvJyjBhApEiwAWz2nLdF-tJTvLFol7gt1kbXoewbMeqf1QXMXIyE62omYmJuDM_gb-vJMOhoCdg4QAvD_BwE.

[15] *DS18B20.pdf [online]. Analog.com.* [vid. 14.5.2023] .
Dostupné z https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf.

[16] *Dallas digitálne vodotesné čidlo teploty DS18B20 1m. [online]. Laskakit.cz.* [vid. 14.5.2023] .
Dostupné z https://www.laskakit.cz/dallas-digitalni-vodotesne-cidlo-teploty-ds18b20-1m/.

[17] *Can I use a copper kettle to boil wort? [online]. MoreBeer!* [vid. 14.5.2023] .
Dostupné z https://www.morebeer.com/questions/20.

[18] *Topné těleso 1500W 230V elektrokotel, bojler. [online]. Elny.cz.* [vid. 14.5.2023.
Dostupné z https://www.elny.cz/topne-teleso-1500w-230v-elektrokotel-bojler-p4095/.

[19] *Napájecí adaptér síťový 3A 5V pro Raspberry Pi microUSB. [online]. Laskakit.cz.* [vid. 14.5.2023] .
Dostupné z https://www.laskakit.cz/napajeci-adapter-sitovy-3a-5v-pro-raspberry-pi-microusb/.

[20] *JSON - JavaScript Object Notation. [online]. JSON.org.* [vid. 14.5.2023] .
Dostupné z https://www.json.org/json-en.html.

[21] *REST API Introduction. [online]. GeeksforGeeks.org.* [vid. 14.5.2023] .
Dostupné z https://www.geeksforgeeks.org/rest-api-introduction/.

[22] *What is PostgreSQL? [online]. AWS Amazon RDS.* [vid. 14.5.2023] .
Dostupné z https://aws.amazon.com/rds/postgresql/what-is-postgresql/.

[23] *asyncio.Task. [online]. Python Docs.* [vid. 14.5.2023] .
Dostupné z https://docs.python.org/3/library/asyncio-task.html.

[24] *FastAPI - Tutorial: Dependencies. [online]. Tiangolo.com.* [vid. 14.5.2023] .
Dostupné z https://fastapi.tiangolo.com/tutorial/dependencies/.

[25] *What Is FastAPI? [online]. Real Python.* [vid. 14.5.2023] .
Dostupné z https://realpython.com/fastapi-python-web-apis/##what-is-fastapi.

[26] *SQLAlchemy - Introduction. [online]. Sqlalchemy.org.* [vid. 14.5.2023] .
Dostupné z https://docs.sqlalchemy.org/en/20/intro.html.

[27] *FastAPI - SQL Databases. [online]. Fastapi.tiangolo.com.* [vid. 14.5.2023] .
Dostupné z https://fastapi.tiangolo.com/tutorial/sql-databases/.

[28] *Vue.js - Introduction. [online]. Vuejs.org.* [vid. 14.5.2023] .
Dostupné z https://vuejs.org/guide/introduction.html##single-file-components.

[29] *What is Tailwind CSS? A Beginner's Guide. [online]. FreeCodeCamp.org.* [vid. 14.5.2023] .
Dostupné z https://www.freecodecamp.org/news/what-is-tailwind-css-a-beginners-guide/.

[30] *An In-Depth Guide to PrimeVue: Enhancing Your Vue.js Applications. [online]. Vue Community.* [vid. 14.5.2023] .
Dostupné z https://vue-community.org/vue/an-in-depth-guide-to-primevue-enhancing-your-vue-js-applications.html.

[31] *Microchip Online Documentation. [online]. MicroChip.* [vid. 14.5.2023] .
Dostupné z https://onlinedocs.microchip.com/pr/GUID-04DAA7D3-9FC2-45F2-B757-157190106490-en-US-2/index.html?GUID-00D4C648-9887-407A-BE44-268E455514E3.

[32] *Busy pin in a SPI connection. [online]. Arduino Forum.* [vid. 14.5.2023] .
Dostupné z https://forum.arduino.cc/t/busy-pin-in-a-spi-connection/556817.

[33] *The Client-server hierarchy used. [online]. ResearchGate.* [vid. 14.5.2023] .
Dostupné z https://www.researchgate.net/figure/The-Client-server-hierarchy-used_fig1_259229950.

[34] *Bigger Applications. [online]. FastAPI.* [vid. 14.5.2023] .
Dostupné z https://fastapi.tiangolo.com/tutorial/bigger-applications/.

[35] *OAuth2 with JWT Tokens (and optional JWT cookies). [online]. FastAPI.* [vid. 14.5.2023] .
Dostupné z https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/.

[36] *What is an ORM? The Meaning of Object-Relational Mapping Database Tools. [online]. FreeCodeCamp.* [vid. 14.5.2023] .
Dostupné z https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/.

[37] *Basic Relationships. [online]. SQLAlchemy.* [vid. 14.5.2023] .
Dostupné z https://docs.sqlalchemy.org/en/20/orm/basic_relationships.html.

[38] *How to Use PrimeVue with Vue 3. [online]. OpenReplay.* [vid. 14.5.2023] .
Dostupné z https://blog.openreplay.com/how-to-use-primevue-with-vue3/.

[39] *Using a Component. [online]. Vue.js.* [vid. 14.5.2023] .
Dostupné z https://vuejs.org/guide/essentials/component-basics.html##using-a-component.

[40] *How to Use Routing in Vue.js to Create a Better User Experience. [online]. FreeCodeCamp.* [vid. 14.5.2023] .
Dostupné z https://www.freecodecamp.org/news/how-to-use-routing-in-vue-js-to-create-a-better-user-experience-98d225bbcdd9/.

[41] *TailWindCSS Documentation. [online]. TailWind.* [vid. 14.5.2023] .
Dostupné z https://tailwindcss.com/.

[42] *Using Axios to Consume APIs. [online]. Vue.js.* [vid. 14.5.2023] .
Dostupné z https://v2.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html?redirect=true.

[43] *Arduino HTTP Request. [online]. Arduino Get Started.* [vid. 14.5.2023] . Dostupné z https://arduinogetstarted.com/tutorials/arduino-http-request.

[44] *Using millis() for timing. [online]. Adafruit.* [vid. 14.5.2023] . Dostupné z https://learn.adafruit.com/multi-tasking-the-arduino-part-1/using-millis-for-timing.

Príloha A
Skratky

A.1 Skratky

ADC Analógovo-digitálny prevodník (ADC) je zariadenie, ktoré konvertuje analógový signál na digitálny formát.

API API je skratka pre Application Programming Interface, čo je sada definovaných pravidiel a funkcií, ktoré umožňujú softvérovým aplikáciám komunikovať a vzájomne si vymieňať informácie.

CO Je skratka pre `Common` a označuje spoločný kontakt v relé, ktorý je pripojený k napájaniu alebo inej zložke relé.

CSS Kaskádové štýly (CSS) je jazyk používaný na definovanie vzhľadu a formátovania webových stránok. Pomocou CSS môžete určiť farbu, veľkosť písma, rozloženie a ďalšie vlastnosti prvkov na webovej stránke.

GND Ground (GND) je elektrický pól alebo uzemnenie, ktoré slúži ako referenčný bod pre elektrické obvody. Je to bežne spojené so zemou alebo negatívnym pólom napájacieho zdroja.

GPIO General Purpose Input/Output (GPIO) sú vstupno-výstupné piny, ktoré umožňujú interakciu mikrokontroléra alebo mikropočítača s vonkajším svetom. Tieto piny môžu byť konfigurované ako vstupné alebo výstupné a môžu sa používať na čítanie a zápis údajov.

HTML Hypertext Markup Language (HTML) je jazyk používaný na tvorbu webových stránok. HTML definuje štruktúru a obsah stránky pomocou značiek a atribútov.

HTTP Hypertext Transfer Protocol (HTTP) je protokol používaný na prenos hypertextových dokumentov na internete. Slúži na požiadavku a poskytovanie webových stránok a dát medzi klientmi a servermi.

I2C Inter-Integrated Circuit (I2C) je sériový komunikačný protokol, ktorý umožňuje prenos dát medzi rôznymi elektronickými zariadeniami. Používa sa pre pripojenie senzorov, displejov a ďalších periférií k mikrokontrolérom.

JSON JavaScript Object Notation (JSON) je formát na výmenu dát, ktorý je ľahko čitateľný pre ľudí aj stroje. Je široko používaný v webovom vývoji na prenos a ukladanie dát.

LED Light Emitting Diociah, ako napríklad indikačné svetlá, displeje, osvetlenie a mnoho ďalších.de (LED) je elektronická súčiastka, ktorá emituje svetlo, keď sa cez ňu prechádza elektrický prúd. Používa sa v rôznych aplikáciah, ako napríklad indikačné svetlá, displeje, osvetlenie a mnoho ďalších.

NC Normally Closed (NC) je označenie pre kontakt, ktorý je v normálnom stave uzavretý. To znamená, že v neaktívnom stave je prítomný elektrický spoj.

OAuth2 OAuth2 je protokol na autentifikáciu a autorizáciu používateľov pri prístupe k webovým službám alebo aplikáciám. Umožňuje používateľom po-

skytnúť prístup k svojim údajom tretím stranám bez zdieľania svojich prihlasovacích údajov.

ORM Object-Relational Mapping (ORM) je technika, ktorá umožňuje mapovať objekty z objektovo-orientovaného programovania na relačné databázy. Umožňuje jednoduchšiu a efektívnejšiu prácu s databázami pomocou objektov.

PWM Pulse Width Modulation (PWM) je technika riadenia signálu, pri ktorej sa mení šírka impulzov signálu, zvyčajne s cieľom regulovať výkon alebo intenzitu niektorých zariadení, ako napríklad svetelných diód alebo motorov.

REST API Representational State Transfer (REST) je architektúrny štýl pre navrhovanie webových služieb. REST API je rozhranie, ktoré umožňuje klientom komunikovať a vymieňať si dáta so servermi pomocou princípov REST.

RST Reset (RST) je signál alebo pripojenie používané na reštartovanie alebo resetovanie zariadenia. Po aktivácii signálu RST sa zariadenie vráti do svojho počiatočného stavu.

SCL Serial Clock Line (SCL) je sériová hodinová linka, ktorá slúži na synchronizáciu prenosu dát medzi zariadeniami v sériovej zbernici, ako je napríklad I2C.

SDA Serial Data Line (SDA) je sériová dátová linka, ktorá sa používa pri komunikácii medzi zariadeniami v sériovej zbernici, ako je napríklad I2C.

SQL Structured Query Language (SQL) je štandardizovaný jazyk používaný na správu a manipuláciu s relačnými databázami. SQL umožňuje vytváranie, aktualizáciu a dotazovanie dát v databázových systémoch.

URL Uniform Resource Locator (URL) je adresa, ktorá identifikuje jedinečný zdroj na internete. URL obsahuje informácie o protokole, doméne, ceste a voliteľných parametroch, ktoré umožňujú prístup k danému zdroju. Používa sa pre prístup k webovým stránkam, obrázkom, videám a ďalším dátovým zdrojom na internete.

VCC VCC je označenie pre napájací napätie v elektronických obvodoch. Zvyčajne sa používa na označenie napájacieho pólu s pozitívnym napätím.

XML Extensible Markup Language (XML) je značkovací jazyk používaný na štrukturovanie dát. XML umožňuje definovať vlastné značky a štruktúru dát, čo z neho robí populárny formát pre výmenu dát medzi aplikáciami.

Príloha B
Zdrojové kódy

B.1 Zdrojové kódy

PivnyKotol.zip - zdrojové kódy pre celú prácu.