

# SmartBrew

## Tímový projekt

### Tím 6

Alexandra Frniaková

Marek Krchňavý

Michaela Nemcová

Lukáš Novota

Peter Stríž

Marek Vajda

**Kontakt:**

[tim6\\_tp2021@googlegroups.com](mailto:tim6_tp2021@googlegroups.com)

# **1 Riadenie projektu**

# 1.1 Úvod

Dokumentáciu odporúčame sledovať na webe nášho tímu (<http://team06-21.studenti.fiit.stuba.sk/docs>), vzhľadom na to, že je oveľa prehľadnejšia ako jeden dokument, dá sa v nej vyhľadávať a je vždy aktuálna.

Tento dokument obsahuje informácie k riadeniu projektu. Poskytuje základné členenie na **Členovia**, **Zápisnice**, **Šprinty**, **Metodiky**.

## **Členovia**

- ◆ Kompetencie jednotlivých členov a podiel práce na projekte v jednotlivých šprintoch.

## **Zápisnice**

- ◆ Zápisnice z jednotlivých týždňov, spolu s ich šablónou.

## **Šprinty**

- ◆ Vyhodnotenia jednotlivých šprintov spolu s ich šablónou.

## **Metodiky**

- ◆ Rôzne metodiky a usmernenia pre efektívnu prácu.

*Last updated on 23. 11. 2021 by **Peter Stríž***

# 1.2 Členovia

Člen	Kompetencie
Alexandra Fniaková	<ul style="list-style-type: none"><li>- front-end</li><li>- zápisnice</li><li>- organizácia a správa odovzdávaných dokumentov</li></ul>
Marek Krchňavý	<ul style="list-style-type: none"><li>- databáza (návrh a implementácia)</li><li>- back-end (dopyty na databázu)</li></ul>
Michaela Nemcová	<ul style="list-style-type: none"><li>- front-end</li><li>- správa dokumentácie</li><li>- zápisnice</li></ul>
Lukáš Novota	<ul style="list-style-type: none"><li>- mock server pre testovanie</li><li>- back-end (testy)</li></ul>
Peter Stríž	<ul style="list-style-type: none"><li>- scrum master</li><li>- moduly (príprava a implementácia)</li><li>- návrhy častí systému</li><li>- automatizácia vývoja</li></ul>
Marek Vajda	<ul style="list-style-type: none"><li>- back-end (API pre moduly a pre front-end)</li><li>- testy</li><li>- automatizácia vývoja</li></ul>

*Uvedené kompetencie nie sú fixné, členovia sa môžu a aj sa podieľajú tiež na iných častiach projektu.*

## 1.2.1 Hodnotenie členov

Šprint / Člen	Saška	Marek K.	Miška	Lukáš	Pet'ó	Marek V.
01 - Zlatý Bažant	17 %	16 %	17 %	17 %	17 %	16 %
02 - Pilsner Urquell	17 %	17 %	16 %	16 %	17 %	17 %
03 - Corgoň	16 %	17 %	16 %	17 %	17 %	17 %
04 - Krušovice	17 %	16 %	17 %	17 %	17 %	16 %
05 - Šariš	17 %	17 %	16 %	16 %	17 %	17 %

Last updated on **15. 12. 2021** by **Peter Stríž**

# 1.3 Retrospektíva za zimný semester

Za prvý semester sa nám podarilo vytvoriť prvý funkčný prototyp pivovaru, čím sme splnili náš semestrálny cieľ. Pivovar podporuje funkcie ako vytváranie receptu, výber receptu a spustenie varenia. Funkcionalita je implementovaná end-to-end, tzn. je podporovaná všetkými komponentami – frontend, backend aj moduly.

Bohužiaľ, nepodarilo sa nám zrealizovať stretnutie na Strojníckej fakulte kvôli pandemickej situácii. Kvôli tomu sme nemohli otestovať, či moduly fungujú tak, ako majú.

Komunikácia v tíme prebiehala efektívne a darilo sa nám udržiavať konštantnú rýchlosť práce. Návrh architektúry prešiel viacerými iteráciami, pričom každý člen tímu sa na ňom podieľal.

Pri vývoji sme aplikovali agilnú metódu SCRUM. Počas semestra sme absolvovali 5 šprintov, pričom každý mal 2 týždne (posledný bol o 2 dni predĺžený). Pre každý šprint sme si na začiatku zadefinovali stories a ciele, ktoré chceme v danom šprinte dosiahnuť. Po ukončení šprintu sme zhodnotili vykonanú prácu a spravili retrospektívu.

Z piatich šprintov sme ukončili úspešne všetky okrem tretieho, z ktorého sme museli prenášať väčšinu úloh do ďalšieho. Takisto sme úlohu vytvorenia webového sídla ukončili až v treťom šprinte (prenášaná od prvého).

Do budúcnosti by sme chceli viac:

- dokumentovať svoju prácu (každý člen samostatne),
- inkrementálnej práce (namiesto nárazovej práce),
- testovať.

V budúcnosti by sme chceli začať:

- efektívnejšie dokumentovanie pre semestrálne odovzдания,
- testovať na reálnom zariadení,
- lepšie deliť prácu.

Chceli by sme si udržať:

- pravidelné stretnutia,
- dobrú komunikáciu,
- tempo práce.

## **2 Zázpisnice**

# 2.1 Zápisnica template

## TODO

Pri vyplňaní template je potreba prepísať hore v hlavičke atribút **title** do formátu: **DD.MM.YYYY** (napr. 04.10.2021). A *sidebarposition* na najnižšie číslo zo zápisníc (napr. predošlá má číslo 3, tak moja bude mať číslo 4)\_.

## Zápisnica - DD.MM.YYYY

### 2.1.1 Zhrnutie taskov

*Zhrnutie, kto je ako na tom so svojimi taskami, ako progressujú stories a podobne. Stačí v odrážkach.*

### 2.1.2 Prezentácia vypracovaných taskov

*Pokiaľ niekto mal task ako analýzu, alebo je to niečo, čo potrebuje celý tím odsúhlasiť.*

### 2.1.3 Identifikácia nových problémov

*S čím sme mali pri vypracovávaní taskov problém a aké nové issues vznikli.*

### 2.1.4 Vytvorenie nových stories/taskov

*Aké nové tasky vznikli a komu boli priradené.*

## Stories

*Novovzniknuté stories.*



# Tasky

*Novovzniknuté tasky.*

## 2.1.5 Diskusia

*Iné témy, ktoré sa prebrali na stretnutí.*

*Last updated on **23. 11. 2021** by **Peter Stríž***

# 2.2 Zápisnica - 27.9.2021

## 2.2.1 Zhrnutie taskov

- ♦ prvé skupinové stretnutie týkajúce sa práce na projekte

## 2.2.2 Vytvorenie nových stories/taskov

- ♦ prebehlo bližšie oboznamovanie sa s témou projektu
- ♦ základné spoločné stretnutia budú prebiehať každý týždeň v pondelok
- ♦ na komunikáciu v tíme bude využívaný Discord
- ♦ prostredie na riadenie scrumu - Jira

## Dlhodobé úlohy

- ♦ kontrola scrumu - Peter Stríž
- ♦ dokumentácia - Michaela Nemcová

## Stories

- ♦ tím si má do ďalšieho stretnutia naštudovať scrum (vhodnú granulu taskov,...)

## Tasky

- ♦ príprava Jira - Peter Stríž
- ♦ príprava Discord - Marek Vajda

## 2.2.3 Diskusia

- ♦ na stretnutí boli opísané technológie a postupy potrebné pri realizácii projektu

# 2.3 Zápisnica - 4.10.2021

## 2.3.1 Zhrnutie taskov

- Jira: je vytvorené prostredie na pridelenie úloh jednotlivým členom tímu podľa princípu scrum
- Discord: pripravený na komunikáciu členov tímu
- Github repozitár: založený na ukladanie a prístup k spoločným kódom
- Docusaurus: založený na dokumentáciu

## 2.3.2 Identifikácia nových problémov

- boli určené hlavné epiky:
  - Ako sládek (osoba, ktorá varí pivo) chcem vedieť komunikovať so senzormi a efektormi (vytvorenie endpointov, modulov,...)
  - Chcem, aby sa pivo varilo automaticky: príprava receptu = nastav teplotu a udržuj ju, reaguj na eventy - zahrievaj na určitú teplotu. Je potrebné určiť architektúru systému, databázy (teda spôsob ukladania dát)
  - Chcem vidieť stav zariadení na monitore - potrebné vizualizovať
  - My ako tím chceme splniť všetky podmienky na úspešné zvládnutie predmetu (skompletizovanie webovej stránky, dokumentácie, dodržanie správnej formy odovzdaného diela)

### Postupy a riešenia:

- ovládanie a vizualizácia dát na monitore - webová aplikácia zobrazujúca sa v browseri
- frontend: JavaScript, React (interaktivita, pridanie údajov (receptu), tie sa ukladajú, dajú sa pozmeniť; tabuľky s hodnotami,...)
- backend: Node.js, možnosť využiť postupy z existujúcej práce (Flask), Sequelize
- databáza: Postgresql

## 2.3.3 Vytvorenie nových stories/taskov

- ♦ naštudovať si komunikáciu pomocou mqtt

### Stories

- ♦ Webová stránka tímu - Marek Vajda
- ♦ Sprevádzkovať Docker na Raspberry Pi (kontajnery ...) - Marek Vajda
- ♦ Prieskum procesov pivovaru - Alexandra Frniaková
- ♦ Priemyselná vizualizácia scadu (prieskum) - Michaela Nemcová
- ♦ Analýza nástrojov (HW) - Peter Stríž

### Tasky

- ♦ Update zápisníc - Alexandra Frniaková
- ♦ Vytvorenie templates pre dokumentáciu - Michaela Nemcová
- ♦ Nástroje na tvorbu dokumentácie - Peter Stríž
- ♦ Export z Jira - Peter Stríž

# 2.4 Zápisnica - 11.10.2021

## 2.4.1 Zhrnutie taskov

Všetkým členom sa podarilo vypracovať tasky z minulého týždňa.

## 2.4.2 Identifikácia nových problémov a riešení

### Asynchrónnosť systému

#### Problém

Kroky receptu (jednotlivé moduly) nepôjdu úplne synchronne a lineárne.

Príklad: modul č. 1 dostane príkaz "zohrej na XY °C a potom teplotu udržuj XY minút" okrem toho môžem chcieť miešať, tak pošlem do 2. modulu príkaz "miešaj na XY otáčok po dobu XY minút" keď 1. modul dokončí, tak backend môže synchronizovať moduly (2. modul nastaví na 0rpm)

#### TL;DR - moduly musia vedieť bežať asynchrónne

#### Návrh riešenia

- ♦ bude centrálny backend, a moduly budú periodicky oznamovať backendu svoj stav (dáta pôjdu do DB) a **zároveň** sa môžu spýtať, čo majú robiť
- ♦ každý modul bude mať nastavený pull-rate --> koľko času musí prejsť medzi dopytmi
  - tento pull-rate musí byť variabilný a musí sa dať v priebehu varenia meniť
- ♦ podľa pull-rate si bude backend checkovať, či je modul živý alebo mŕtvy a môže to rovno backend vyriešiť
- ♦ backend musí byť robustný, aby vedel spravovať každý modul
- ♦ modul bude vedieť nahlásiť, v akom stave je (IN PROGRESS, DONE, FAILED) a sám si podľa pull-rate bude pýtať od backendu, čo má vykonať

# Architektúra HW

## Návrh riešenia

- ♦ i. ESP má oba motory
- ♦ ii. ESP má senzory z prvej nádoby
- ♦ iii. ESP má senzory z druhej nádoby
- ♦ ESPčka sú pripojené cez WiFi na RPI
- ♦ moduly si pýtajú veci od backendu, nie priamo z DB
- ♦ služby pre moduly napr.
  - zapísať hodnoty zo senzorov
  - vypýtať, čo má modul robiť
  - stav vykonávania

## Rozloženie logiky !

### Problém

**Chceme všetku logiku na backende (moduly budú "blbé"), alebo nejakú logiku delegovať na moduly?**

- ♦ ak vypadne internet, treba zabezpečiť, aby modul sám zastal

### Návrh riešenia



Tím musí vypracovať analýzu, ako by varenie prebiehalo pri oboch konfiguráciách.

## Návrh front-endu

Front-end musí odrážať granularitu receptu a zároveň byť user-friendly aj pre ľudí so základnými počítačovými skillmi.

Front-end by mal vedieť načítať údaje pre časti pivovaru z nejakého konfiguračného súboru.

# Granularita receptu

## Problém

Akú granularitu zvolit' pre recept, aby podporoval ukladanie do DB, zobrazenie na FE a asynchrónne vykonávanie inštrukcií.

## Návrh riešenia

- ♦ recept bude rozdelený na **KROKY**
- ♦ KROK - súbor inštrukcií, ktoré môžu byť vykonávané asynchrónne, ale musia byť všetky ukončené, aby sa recept mohol posunúť na ďalší krok
- ♦ každý krok sa bude skladať z jednotlivých **INŠTRUKCIÍ**
- ♦ inštrukcia - najmenšia jednotka granularity receptu, obsahuje:
  - ID zariadenia, na ktoré je mierená
  - konkrétnu inštrukciu, ktorú má vykonať (napr. naštartuj motor, nastav teplotu)
  - parametre inštrukcie (napr. trvanie, teplota)

## Priebeh receptu:

- ♦ backend si do QUEUE nastaví inštrukcie z prvého kroku
- ♦ keď backendu príde dopyt z modulu, tak pokiaľ je v queue nejaká inštrukcia pre daný modul, backend mu pošle danú inštrukciu (modul si ju spracuje sám)
- ♦ po odovzdaní všetkých inštrukcií modulom bude backend čakať, kým mu moduly neoznámia, že splnili inštrukciu
- ♦ pokiaľ všetky moduly majú inštrukcie splnené, vtedy backend prejde na ďalší krok a proces a zopakuje
- ♦ toto celé sa opakuje, kým v recepte existujú nesplnené kroky

## Poznámky

- ♦ backend si musí (napr. v databáze) držať pre každé zariadenie (efektor) súbor inštrukcií, ktoré vie dané zariadenie vykonať
  - asi bude potrebné ukladať aj to, aké parametre inštrukcia berie

## 2.4.3 Tasky do najbližšieho stretnutia

- ♦ **všetci** - deploy prvej funkčnej iterácie na RPI (aby už to malo aj nejakú jednoduchú statickú HTML stránku)
- ♦ **všetci** - kuknúť Git repozitár predošlého tímu
  - budúci týždeň diskusia, čo odtiaľ prevziať
  - vytvoriť si template z motora
  - v repozitári je problém s HW ako je navrhnutý mikropočítač
- ♦ **Saška** - dokončiť analýzu varenia piva
- ♦ **Saška** - prepísať recept na primitívne a aj pokročilejšie kroky a nastavenia
- ♦ **Miška** - návrh frontendu (inšpirovať sa Lego Mindstorms)
- ♦ **Peťo** - pohľadať ovládateľný chladiaci efektor (napr. čo sa používa pri domácich pivovaroch)
- ♦ **Lukáš a Marek K.** - návrh databázy
- ♦ **Lukáš** - analýza TimeScale
- ♦ **všetci** - vytvoriť analýzu jednotlivých operácií v oboch prípadoch rozloženia logiky, či by náhodou nenastala nejaká veľmi komplikovaná situácia

## 2.4.4 Doplnenie informácií

### Recept

- ♦ bude sa zohrievať celá nádoba (hlavne prvá), v druhej nádobe sa bude chlaďiť (celá nádoba)
- ♦ v prvej nádobe sa bude variť slad
- ♦ potom sa zapne výveva, vytvorí podtlak a nasaje obsah prvej do druhej nádoby
- ♦ v druhej nádobe sa bude chmeliť, chmel bude v sitku, to sa potom vyberie a pivo bude zrietať v druhej nádobe
- ♦ nebude sa merať hladina

### Senzory a efekторы

#### Senzory

- ♦ teplomer (viacero)



- ◆ merač prúdu (pre vývevu, či už všetko precucla)
- ◆ tlakový senzor
- ◆ menič na motor - vie nastaviť:
  - ako dlho má ísť
  - ako dlho sa má rozbiehať a spomalovať
  - na koľko otáčok má ísť

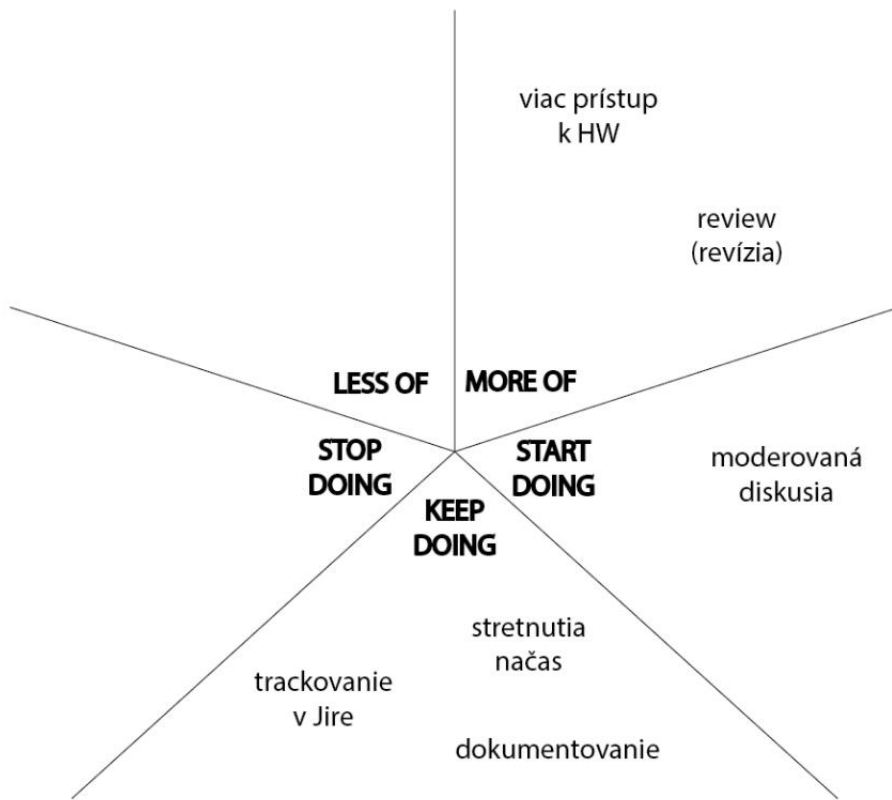
## Efektory

- ◆ motor
- ◆ servomotor
- ◆ ventil na prvej nádobe
- ◆ výveva
- ◆ ohrevné pásy
- ◆ kryostat - zariadenie na nastavenie teploty (**ešte nie je**)
- ◆ *nemáme ovládateľné chladiace efektory*

Last updated on **23. 11. 2021** by **Peter Stríž**

# 2.5 Zápisnica - 18.10.2021

## 2.5.1 Retrospektíva



## 2.5.2 Zhrnutie taskov

Všetkým členom sa podarilo vypracovať tasky z minulého týždňa.

## 2.5.3 Prezentácia vypracovaných taskov

Marek K. prezentoval svoj návrh relačno-entitného modelu DB. Po diskusii sme sa dohodli, že návrh upraví kvôli identifikovaným problémom.

Miška prezentovala návrh front-endu vo Figma. Návrh spravili aj Saška a Peťo. Po diskusii sme sa dohodli, že použijeme Peťov návrh.

## 2.5.4 Identifikácia nových problémov

- ♦ Peťo potrebuje viac HW - káblíky a breadboard

### Komunikácia medzi BE a modulmi

- ♦ nebudeme využívať MQTT, ale WebSocket
- ♦ komunikácia bude ale stále vedená jednosmerne (modul si bude periodicky pýtať inštrukcie od BE)
- ♦ paralelizované budú len moduly medzi sebou, nie inštrukcie v rámci modulu (tie budú serializované)

### Potreba nových inštrukcií a krokov

#### Inštrukcia DELAY

Zabezpečuje čakanie modulu na ďalšiu inštrukciu. Modul si odpočet času bude zabezpečovať sám.

#### Krok ALERT

Krok upozorní používateľa, že je potreba vykonať niečo manuálne (napr. nasypať kvasnice, vybrať chmeľ, ...).

Front-end (aj back-end) by čakali na potvrdenie od používateľa, že danú vec vykonal.

### Logovanie

#### Priebeh varenia

- ♦ log - čas ; či to je chyba ; správa
- ♦ logovať sa budú všetky údaje dokopy (v jednom stĺpci)

#### Back-end

- ♦ logy z BE stačí ukladať do súborov

### Nedostupnosť HW pre testovanie

Potrebujeme testovací server, ktorý by simuloval moduly.

## 2.5.5 Doplnujúce informácie

- ♦ FE si bude každú sekundu pýtať najnovšie dáta, ktoré si BE cachuje v objekte
- ♦ handlovanie správnosti hodnôt sa bude vykonávať na module (keby napr. boli vračané zlé teploty)

## 2.5.6 Vytvorenie nových stories/taskov

### Stories

- ♦ hotový relačno-entitný model (spolu s validáciou a deploy skriptami)
- ♦ chceme vedieť na FE zobrazit' teplotu - **definition of done**
  - musí byť možné vytvorit' recept, v ktorom bude inštrukcia "zvyš teplotu"
- ♦ virtuálny testovací modul

### Tasky

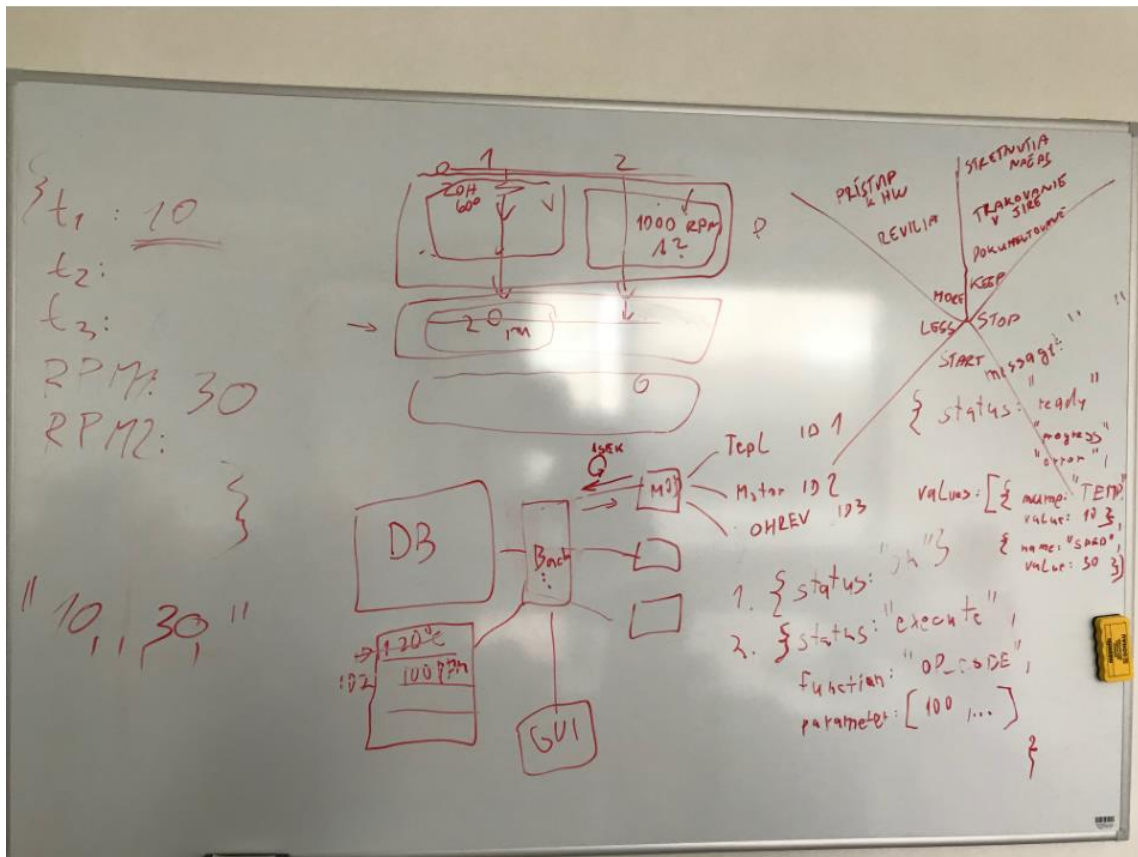
- ♦ definovať formáty JSONov pre komunikácie:
  - BE- modul
    - BE - FE
- ♦ **Lukáš** - vytvorit' testovací simulátor modulov
- ♦ **Miška, Saška** - definovať JSONy, ktoré bude FE posielat' na BE a vice versa
- ♦ **Peťo** - vytvorit' stránku pre nový šprint v dokumentácii
- ♦ **Peťo** - práca na fyzickom zariadení
- ♦ **Marek K.** - upraviť návrh DB podľa diskusie
- ♦ **Marek V.** - vytvorit' API end-pointy pre komunikáciu s FE

## 2.5.7 Diskusia

Dohodli sme sa, že je potreba viesť štruktúrovanejšiu a moderovanú diskusiu. Marek V. navrhol metodiku pre vedenie diskusie.

# Nápady

Nasledujúce nápady spolu aj s nejakou úplne základnou štruktúrou údajov sme si aj nakreslili:



# 2.6 Zázpisnica - 25.10.2021

## 2.6.1 Zhrnutie taskov

Pokračuje priebežná práca na taskoch. Bol určený spôsob ohodnocovania jednotlivých taskov. Jednotkou hodnotenia bude vytvorenie jedného komponentu (frontend/backend), od neho sa bude odvíjať hodnotenie ostatných taskov.

## 2.6.2 Prezentácia vypracovaných taskov

Bol odprezentovaný upravený návrh frontendu, ktorý pozostával z hlavnej obrazovky: pred pridaním receptu, počas varenia receptu; obrazovka výberu receptu, spolu s neho znením + postupom krokov ktoré je potrebné vykonať; obrazovka pridávania receptov.

Prebieha práca na mock serveri.

Boli pripravené prvé komponenty, na ktoré bude nadväzovať tvorba frontendu.

Pokračuje sa na návrhu databázy, spolu s prípravou komunikácie medzi jednotlivými vrstvami aplikácie.

## 2.6.3 Identifikácia nových problémov

Kroky sa nebudú posielat' a spracovávat' paralelne – všetko pôjde sériovo, pričom oneskorenia budú minimálne. Frontend si to ale bude vykresľovat' ako skupinu (podľa template, do ktorého bude autor receptu len dopísavat' hodnoty). Takýchto tamplatov bude viacero, každý krok bude mať vlastný názov, ktorý bude spolu s krokmi zobrazený na frontende.

Z čoho sa skladá informácia pri zasielaní receptu: Step (napr. heat up), parameter (napr. 70 (stupňov celzia)), poradie, id

Recepty bude možné upravovat', niektoré recepty ale budú označené príznakom, že tie sa upravovat' dat' nedajú.

Postup pri výbere receptu: frontend si vypýta list receptov - názvy aj celé recepty. Po kliknutí na vybraný

recept sa zobrazí na druhej strane obrazovky. Po potvrdení receptu sa používateľovi zobrazia podmienky, ktoré musí splniť pred začiatkom varenia. Po ich odkliknutí začína proces varenia.

## **2.6.4 Vytvorenie nových stories/taskov**

### **Tasky**

Práca na fronde - tvorba komponentov a prepojení s backendom. Natiahnutie údajov z backendu a ich zobrazenie - pri danom story pointe sa jedná hlavne o teplotu. - tento task treba rozdeliť na menšie subtasky a rozdeliť medzi ne bodové hodnotenie.

# 2.7 Zápisnica - 02.11.2021

## 2.7.1 Zhrnutie taskov

Splnené definition of done - produkt je v tejto etape funkčný a otestovaný.

## 2.7.2 Prezentácia vypracovaných taskov

Ukončenie šprintu a vytvorenie retrospektívy - vid' Šprint 2.

## 2.7.3 Vytvorenie nových stories/taskov

### Stories

- Ako používateľ chcem vytvoriť nový recept.
- Ako používateľ chcem spustiť varenie podľa receptu.
- Ako používateľ chcem otestovať novú funkčnosť na mock serveri.

### Tasky

- **Pet'o** - prementovať stories na štýl "Ako používateľ chcem ..."
- **Marek K.** - doplniť do dokumentácie example JSON pre vytváranie receptu
- **Marek V.** - nový endpoint pre BE - vytvorenie názvu bloku
- **Lukáš** - rozšírenie funkčnosti mock serveru
- **Saška a Miška** - front-end



## 2.7.4 Diskusia

### Doplnenie info k inštrukciám

- ♦ pumpa ide ako **delay** inštrukcia
- ♦ motor je instantná inštrukcia
- ♦ kontrolujeme teplotu

### Plány do budúcnosti

- ♦ na FE posielat', koľkokrát bol recept použitý, aby sme mohli ukázať "obľúbené" recepty

*Last updated on 23. 11. 2021 by **Peter Stríž***

# 2.8 Zápisnica - 08.11.2021

Išlo o klasický stand-up.

## 2.8.1 Zhrnutie taskov

Všetci členovia pokročili na svojich častiach.

- ♦ Peťo - demo verzia funkcionality na module
- ♦ Lukáš - robustnejší mock server - znovupripojenie po strate pripojenia
- ♦ Marek K. - vytvorenie receptu a načítanie receptu pred začiatkom varenia
- ♦ Miška a Saška - práca na obrazovke pre výber / pridanie receptu
- ♦ Marek V. - práca na back-ende - ošetrovanie API

## 2.8.2 Vytvorenie nových taskov

Na stretnutí nevznikli nové tasky.

*Last updated on 23. 11. 2021 by **Peter Stríž***

# 2.9 Zápisnica - 15.11.2021

## 2.9.1 Zhrnutie taskov

Šprint sa nepodarilo úspešne dokončiť. Otvorené tasky sa presúvajú do ďalšieho šprintu.

## 2.9.2 Prezentácia vypracovaných taskov

- ♦ funkcionalitu na BE sa podarilo dokončiť (Marek V. a Marek K.)
- ♦ mock server funguje pre definované use cases (Lukáš)
- ♦ FE nie je dokončený (Miška a Saška)
- ♦ Peťo vytvoril usmernenie pre formát JSONov, ktoré sa posielajú medzi časťami systému

## 2.9.3 Vytvorenie nových stories/taskov

### Stories

Pokračujeme v stories z predošlého šprintu.

### Tasky

- ♦ Marek V. - vytvoriť možnosť pre spúšťanie testov pri pushovaní na GitHub
- ♦ Peťo - tento týždeň dokončiť a napojiť modul na pivovar
- ♦ Marek K.
  - dorobiť handlovanie chýb pri vkladaní do databázy + nastavovanie statusov
  - prerobiť atribúty v DB na CamelCase
- ♦ Saška a Miška
  - pokračovať v taskoch na FE
  - menu bar - Home, Výber receptu, História (možno aj nejaký status pivovaru)
  -

shutdown tlačidlo a správa na BE

- ◆ Lukáš
  - systémový test pre BE + debug
  - funkcionálnosť pre korektné vypnutie RPi na BE (endpoint, skript)

*Last updated on 23. 11. 2021 by **Peter Stríž***

# 2.10 Zápisnica - 22.11.2021

Na stretnutí sa najprv preberalo odovzdanie dokumentácie (Mílnik 1), potom členovia prezentovali stav taskov.

## 2.10.1 Zhrnutie taskov

- ◆ Marek K.
  - čo spravil:
    - hotfix dockeru pre BE, aby sa dal vybuildovať
    - premenovanie atribútov v DB na camel case
    - handlovanie promisov na BE (nedokončené)
  - čo spraví do konca šprintu:
    - logovanie (aspoň prvotná verzia)
    - dokončenie aktívnych taskov v Jire
- ◆ Saška
  - čo spravila:
    - rozdelenie formuláru na zadávanie ingrediencií na kategórie + CSS update
    - update vizuálu stránky pre výber receptu
  - čo spraví do konca šprintu:
    - hlavné menu pre navigáciu na front-ende
    - výber a úprava receptu ako single-page app. (sidebar zostane rovnaký, len sa v ňom budú prekreslovať údaje)
- ◆ Peťo
  - čo spravil:
    - spájkovanie modulu
    - objednanie konvertera pre modul
    - dokončené ESP s handlovaním teplomeru a motoru
    - testovacia aplikácia pre posielanie inštrukcií na moduly
- ◆ Marek V.
  - čo spravil:
    - testovanie

- Jest na BE
  - ešte to potreba doladiť
- čo spraví do konca šprintu:
  - web stránka (+ zistenie, čo všetko treba na web stránke)
- ◆ Miška
  - čo spravila:
    - dokončovanie dokumentácie
    - prvotný návrh vizuálu pre hlavné menu stránky
  - čo spraví do konca šprintu:
    - funkčné skladanie inštrukcií pre recept
- ◆ Lukáš
  - čo spravil:
    - shutdown RPi z BE - nie hotové
  - čo spraví do konca šprintu:
    - testovanie procesu varenia

## 2.10.2 Identifikácia nových problémov

- ◆ robiť viac **reviews** na hotové tasky
  - pokiaľ člen nemá dost' práce, môže sa venovať tomuto
- ◆ iniciatívnejšie sa zapájať do robenia taskov, pokiaľ sám nemám tasky (člen si sám môže niečo nájsť, nemusí čakať na stretnutie)

## 2.10.3 Vytvorenie nových stories/taskov

Na stretnutí neboli vytvorené nové tasky.

## 2.10.4 Diskusia

- ◆ bolo by dobré, keby sme do konca zimného semestra (resp. do začiatku letného semestra) mali už fungujúci prototyp, aby sme sa v ďalšom semestri mohli venovať TP Cupu
- ◆ dokumentácia pre Milník 1 má obsahovať ako cieľ:
  - viem si vyklikať (vytvoriť recept)

- viem spustiť varenie
- viem si zobrazit' priebeh varenia
- postup varenia sa loguje do databázy

# 2.11 Zápisnica - 29.11.2021

## 2.11.1 Zhrnutie taskov

Šprint sa podarilo uzavrieť úspešne, tasky boli splnené.

## 2.11.2 Prezentácia vypracovaných taskov

Zapísané v retrospektíve v časti Špriny - Krušovice.

## 2.11.3 Vytvorenie nových stories/taskov

Cieľom šprintu je

- ♦ dokončiť vytváranie a zadanie receptu
- ♦ zabezpečiť fungovanie end-to-end
- ♦ vyladiť hotové funkcionality

### Stories

- ♦ dokončiť vytváranie nového receptu
- ♦ zobrazovať aktuálne vykonávanú inštrukciu na FE

### Tasky

- ♦ úprava dátových typov pri vytváraní receptov na FE + dorobiť odpovedajúcu funkcionality na BE
- ♦ dokončiť preklikávanie medzi formulármi pri vytváraní receptu
- ♦ pridať sidebar do prehľadovej stránky (zobrazenie ingrediencií a inštrukcií)
- ♦ pridať zobrazenie inštrukcií pri prehľade receptu a pri načítanom recepte
- ♦ pridať API endpoint funkcionality pre abort varenia
- ♦ pridať API endpoint funkcionality pre pauzu varenia
  -



- treba vyriešiť, ako to spraviť z pohľadu use caseov
- ◆ handle pre manuálnu inštrukciu
  - funkcionalita endpointu
- ◆ handle čakaciu inštrukciu
  - čakanie na moduloch
- ◆ transfer liquids inštrukciu prerobiť iba na „otvoriť konkrétny ventil“
- ◆ rozbehnúť docker na RPi – fix
- ◆ getovať template funkcie pri rozbehnutí FE

## Hlavné delenie

- ◆ Marek V. – vytvorenie testov pre endpointy na BE
- ◆ Miška a Saška – pokračovať vo FE
- ◆ Lukáš
  - čakacia inštrukcia na mock serveri
  - pomôže na FE
- ◆ Marek K.
  - dokumentácie pre BE
- ◆ Peťo
  - práca na moduloch

## 2.11.4 Diskusia

Počas diskusie boli prediskutované dané návrhy/problémy:

- ◆ problémy s vytváraním testov na BE kvôli typescriptu
- ◆ čakacia a manual inštrukcia
  - udržiava stav – točenie motorov, teplota atď
  - manuálna inštrukcia bude mať option pre vypnutie všetkých motorov
- ◆ pauza
  - udržiava stav – točenie motorov, teplota
  - spustená a ukončená používateľom

# 2.12 Zápisnica - 6.12.2021

Na stretnutí členovia prezentovali stav taskov, respektíve ich ďalšie plány pri ich plnení.

## 2.12.1 Zhrnutie taskov

- ◆ Saška
  - preklikávanie medzi obrazovkami
  - sidebar pri pridávaní inštrukcií
  - popup
- ◆ Miška
  - zobrazenie inštrukcií po blokoch v prehľade receptu
  - správne formátovanie parametrov a názvov inštrukcií v prehľade receptu
- ◆ Peťo
  - FE komunikácia s BE
  - wait na module (neotestované)
- ◆ Marek V.
  - rozbehal testy
  - vytvorený test, ktorý spustí recept a skontroluje, či sa vykonáva
- ◆ Marek K.
  - abort/pauza vykonávania receptu
  - riešenie problémov s ORM Prismou na 32-bitovom Raspbiane
- ◆ Lukáš
  - čakanie na BE

## 2.12.2 Identifikácia nových problémov

- ◆ Na FE je potrebné pridať ošetrovania vstupov a drobné debugy. Pri BE zase rozbehať testy, ORM a pridať dokumentáciu.

## 2.12.3 Vytvorenie nových stories/taskov

- ♦ Lukáš – endpoint pre vymazanie receptu (soft-delete)
- ♦ Saška a Miška – dolad'ovanie FE
- ♦ Marek K. – dokumentácia

## 2.12.4 Diskusia

- ♦ stále nemáme k dispozícii schému pre HMI, bez ktorej nie je možné zakresľovať aktuálny stav pivovaru (teplotu v jednotlivých nádobách a podobne)

*Last updated on 16. 12. 2021 by **Michaela Nemcová***

# 2.13 Zápisnica - 13.12.2021

## 2.13.1 Zhrnutie taskov

Šprint sa podarilo ukončiť úspešne, všetky tasky boli splnené.

## 2.13.2 Prezentácia vypracovaných taskov

Zapísané v retrospektíve v časti Šprinty - Šariš.

## 2.13.3 Vytvorenie nových stories/taskov

Vzhľadom na koniec semestra neboli vytvorené nové stories/tasky.

## 2.13.4 Diskusia

Počas diskusie vedúci projektu pripomienkoval nedostatky v dokumentácii, ktoré je potreba do odovzdania odstrániť.

*Last updated on 15. 12. 2021 by Peter Stríž*

# **3 Šprinty**

# 3.1 Názov šprintu



TODO

Pri vyplňaní template je potreba prepísať hore v hlavičke atribút **title** do formátu: **XY - [Názov šprintu]**, kde XY je číslo šprintu (*napr. 01 - Zlatý Bažant*).

## 3.1.1 Kedy?

**Od:** (dátum)

**Do:** (dátum)

## 3.1.2 Cieľ šprintu

*Zhrnutie, čo je cieľom a malo by byť výsledkom šprintu.*

## 3.1.3 Obsah šprintu

*Stručne o stories, ktoré máme naplánované na šprint.*

### Story XYZ

*Ja ako XY chcem XY, lebo ... (a podobné veci)*

### [Názov šprintu] Story XYZ

*V prípade že sa úloha prenášala, napíšeme do názvu aj názov originálneho šprintu*

## 3.1.4 Výsledky šprintu

*Čo sa nám v šprinte podarilo a čo sme nestihli.*

## Splnené úlohy

- ♦ *Úloha 1*
- ♦ *Úloha 2*
- ♦ ...

## Nesplnené úlohy

- ♦ *Úloha 1*
- ♦ *Úloha 2*
- ♦ ...

## Poznámky do budúcnosti

*Niečo, čo si máme zapamätať alebo nejaké odkazy do budúcnosti, čo sme sa naučili na tomto šprinte.*

*Last updated on 14. 12. 2021 by **Peter Stríž***

# 3.2 Zlatý bažant

## 3.2.1 Kedy?

**Od:** 3.10.2021

**Do:** 18.10.2021

## 3.2.2 Cieľ šprintu

Úlohou prvého šprintu bol prvotný setup všetkých nástrojov akými sú Git, Jira či nástroj na tvorbu dokumentácie Docusaurus.

Ďalej bolo potrebné vytvoriť repozitáre pre jednotlivé časti projektu a v rámci nich vytvoriť jednoduché demo.

## 3.2.3 Obsah šprintu

Krátky opis stories, na ktorých sme pracovali počas šprintu. Treba dodať že nie všetci členovia figurujú ako zodpovední za nejakú story, avšak na práci sa podieľal každý. Každá story totiž obsahovala viacero úloh, ktoré sme už do tohto zápisu neuviedli, lebo ich bolo veľmi veľa (stories + úloh bolo cez 40).

## Technical prototype on RPI

Hlavný story. Prototyp pivovaru, ktorý sa automaticky bude nasadzovať na RPI a pri startupe sa naštartuje automaticky.

## Frontend analysis

Základný dizajn frontendu spolu s technológiami. Viac na Frontend.



## Analysis on how to brew a beer

Ako sa vôbec varí pivo, keďže väčšina členov tímu nevie aké sú tam kroky, a následne sa z tejto analýzy dá ľahšie zistiť, že čo treba spraviť. Viac na [Brewery](#).

## Basic tools

Základné nástroje ktorými sú: [git](#), [jira](#), [dokumentácia](#).

## Architecture overview

High level pohľad na časti systému. Viac na [Architektúra](#).

## Analyze existing SCADA software

Analýza iných softvérov, ktoré sú určité na podobné použite. Viac na [Scada](#).

## Team webpage

Základná tímová stránka.

## Analysis of tools to use on HW

Aké rôzne komponenty sa dajú použiť na HW. Viac na [Modbus](#).

## 3.2.4 Výsledky šprintu

Nástroj Jira nám ponúka nasledujúce vizualizácie hlavných stories a taskov.

### Splnené úlohy

Key	Summary	Issue type	Epic	Status	Assignee
SB-3	Tool for creating documentation	Task	SPLNENIE PODMIE...	DONE	PS
SB-8	Analysis on how to brew a beer	Story	ZABEZPEČENIE AUT...	DONE	AF
SB-10	Basic tools	Story	SPLNENIE PODMIE...	DONE	PS
SB-12	Analyze existing SCADA software	Story	ZABEZPEČENIE AUT...	DONE	MN
SB-18	Architecture overview	Story	KOMUNIKÁCIA SO ...	DONE	MV
SB-24	Update documentation with new Record	Task	SPLNENIE PODMIE...	DONE	AF
SB-25	Create export from Jira	Task	SPLNENIE PODMIE...	DONE	PS
SB-28	Create templates for documentation	Task	SPLNENIE PODMIE...	DONE	MN
SB-37	Update Records from 11.10.2021	Task	SPLNENIE PODMIE...	DONE	MN
SB-38	Frontend analysis	Story	VIZUALIZOVANIE S...	DONE	MN
SB-41	Create temporary static HTML on frontend	Task	VIZUALIZOVANIE S...	DONE	MN

### Nesplnené úlohy

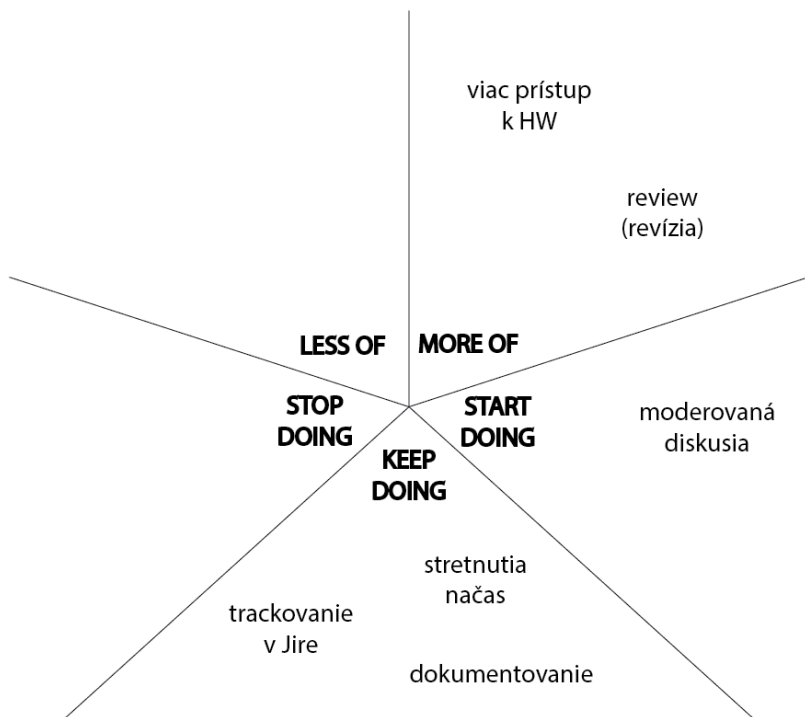
Key	Summary	Issue type	Epic	Status	Assignee
SB-5	Technical prototype on RP	Story	KOMUNIKÁCIA SO ...	IN PROGRESS	MV
SB-13	Team webpage	Story	SPLNENIE PODMIE...	IN PROGRESS	MV
SB-29	Analysis of tools to use on HW	Story	ZABEZPEČENIE AUT...	IN PROGRESS	PS

\*poznámka: *Technical prototype on RPI* sa podarilo takmer celé a hlavná funkcionálna bola, avšak nepodarilo sa nám stihnúť jednu úlohu, ktorú bolo treba odkomunikovať s ostatnými členmi tímu až po ukončení šprintu.

### Poznámky do budúcnosti

Lepšie popisovať názvy stories, aby sa držali viac metódy Scrumu.

Počas retrospektívy vznikol nasledujúci starfish diagram.



Last updated on **14. 12. 2021** by **Peter Stríž**

# 3.3 Pilsner Urquell

## 3.3.1 Kedy?

**Od:** 18.10.2021

**Do:** 2.11.2021

## 3.3.2 Cieľ šprintu

Hlavnou úlohou tohto šprintu bolo vedieť si zobrazit' nejaké údaje zo senzorov.

Znamená to, že sme museli spraviť návrh grafického rozhrania. Návrh štruktúry API a komunikácie medzi časťami systému.

## 3.3.3 Obsah šprintu

Hlavným story bolo zobrazenie údajov na grafickom rozhraní. Hlavne sa to týkalo zobrazenia aktuálnej teploty, ktorú nameral teplomer.

### **As a User, I want to see some basic data on frontend from sensors, so that I know the current status of brewery**

Hlavné story, pri ktorom bolo treba vytvorit' základné grafické rozhranie. Ďalej bolo treba umožniť získavanie najaktuálnejších dát z backendu. Takisto sme museli upraviť moduly, aby posielali správy o aktuálnom stave senzorov. Všetko toto zahrňovalo aj návrh komunikácie medzi jednotlivými komponentami spolu s jej implementáciou.

### **As a Developer, I want to create mock server that acts like module, so that I can test apps easier**

Vytvorenie testovacieho modulu, ktorý si vie každý člen tímu spustiť u seba lokálne. Dôležité to je preto, lebo nie každý má prístup k HW a aby sme si mohli otestovať jednotlivé časti, musíme nejakým spôsobom simulovať modul.

# As a Student, I want to keep documentation updated, so that I can pass the subject

Story do ktorého sme dávali veci ako pridanie novej zázpisnice, či prihlásenie sa do TP Cup.

## [Zlatý bažant] Technical prototype on RP

Z predchádzajúceho šprintu sme tento story dokončili ihneď po stretnutí, keď sme si zdefinovali niečo, čo nám chýbalo pri poslednej pod-úlohe.

## [Zlatý bažant] Analysis of tools to use on HW

Takisto z predchádzajúceho šprintu. Pričom bolo vykonaná analýza ohľadom zariadení na reguláciu teploty.

## [Zlatý bažant] As a Team, we want to have a team webpage

Práca na tímovom webe bola minimálna. Jediné čo pribudlo, je odkaz na našu dokumentáciu priamo z webu v sekcií dokumenty. Farebné štýly boli upravené tak, aby dodržovali našu identitu.

## 3.3.4 Výsledky šprintu

Nástroj Jira nám ponúka nasledujúce vizualizácie hlavných stories a taskov. Podarilo sa nám spraviť 116.5/133 (87%) story pointov. Výsledný burndown graf nebudeme ukazovať, pretože nástroj Jira neponúka zobrazenie story pointov v sub-úlohách a výsledkom bolo to, že sme mali story s 90 bodmy, ktorý sme dokončili tesne pred koncom šprintu, čiže graf vyzeral akoby sme nič nerobili a zrazu väčšina zmizla.

### Splnené úlohy

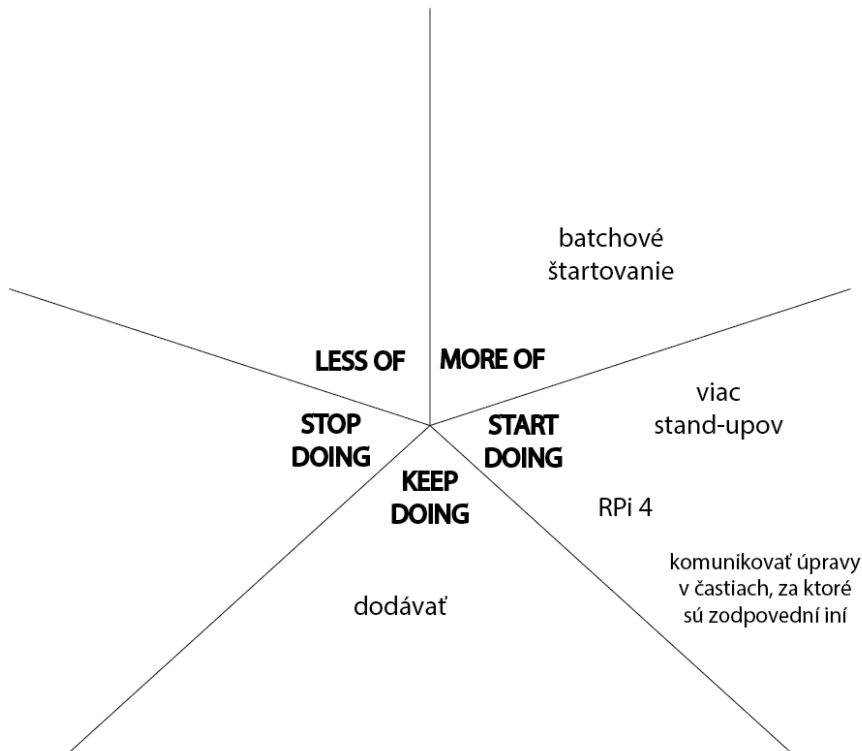
Key	Summary	Issue type	Epic	Status	Assignee
SB-42	Show basic data on frontend from sensors	Story	VIZUALIZOVANIE S...	DONE	PS
SB-44	Create mock server, that acts like module for easier testing	Story	KOMUNIKÁCIA SO ...	DONE	LN
SB-45	Keep documentation updated	Story	SPLNENIE PODMIE...	DONE	PS
SB-5	[From prev sprint] Technical prototype on RP	Story	KOMUNIKÁCIA SO ...	DONE	MV
SB-29	Analysis of tools to use on HW	Story	ZABEZPEČENIE AUT...	DONE	PS
SB-58	documentation - meeting guide	Task	SPLNENIE PODMIE...	DONE	MV
SB-69	TP-Cup registration	Task	SPLNENIE PODMIE...	DONE	AF

# Nesplnené úlohy

Key :	Summary :	Issue type :	Epic :	Status :	Assignee :
SB-13	As a Team, we want to have team webpage	Story	SPLNENIE PODMIE...	IN PROGRESS	MV

## Poznámky do budúcnosti

Počas retrospektívy vznikol nasledujúci starfish diagram.



Last updated on 14. 12. 2021 by Peter Stríž

# 3.4 Corgoň

## 3.4.1 Kedy?

**Od:** 2.11.2021

**Do:** 15.11.2021

## 3.4.2 Cieľ šprintu

Hlavným cieľom tohto šprintu bolo vedieť vytvoriť nový recept a následne tento recept použiť na varenie.

## 3.4.3 Obsah šprintu

Bolo treba premyslieť grafické rozhranie a urobiť ho viac prívetivím pre používateľa (na konci stránky sú aj obrázky). Ďalej bolo treba zjednodušiť testovanie jednotlivých častí systému - testovací server a spúšťací script.

Hlavnou časťou bolo vytváranie receptov a následné použitie týchto receptov na spústenie varenia. Čiže hlavná logika a proces akým sa presúvajú inštrukcie zo stavu **vykonávania** do **ukončeného** stavu.

### **As a Developer, I want mock server to simulate instructions for brewing, so that I can test the app**

Podpora vykonávania funkcií na testovacom servery.

### **As a Developer, I want easier startup of every app, so that I can develop faster**

Jednoduchý script ktorý spustí celý systém (viac info **Spustenie pivovaru**).

## **As a Student, I want to keep documentation updated, so that I can pass the subject**

Klasická úloha vrámci ktorej sa vypisujú zápisnice či šprinty a podobné veci dokumentačného charakteru.

## **As a User, I want to create a new recipe, so that I can use it to brew a beer**

Vytváranie nového receptu. Zahrňuje návrh GUI, zmeny v databáze a aj úpravu backendu.

## **As a User, I want to use a simple recipe, so that I can test if brewing works**

Aplikovanie receptu na varenie. Zahrňuje úpravu GUI, takisto aj logiku procesu varenia na backende a spracovanie inštrukcií na moduloch.

## **[Zlatý bažant] As a Team, we want to have a team webpage, so that We can present our work**

Práca na tímovom webe bola menšia. Pribudlo iba zopár textov, ktoré sme použili vrámci našej prihlášky do TP Cup.

### **3.4.4 Výsledky šprintu**

Nástroj **Jira** nám ponúka nasledujúce vizualizácie hlavných stories a taskov.

Podarilo sa spraviť hlavne backendovú časť celého šprintu, avšak na frontende vzniklo veľa problémov a preto sa tam nestihlo všetko dokončiť. Hlavná kostra tam už je, ale nieje to plne funkčné.

Vzhľadom na korona situáciu, sme nemohli veľa pracovať na úlohách a preto sa nestihli ani spraviť všetky zápisnice načas. Takisto sme si dali veľmi veľa úloh, ktoré boli veľmi náročné a aj preto sa nestihli dokončiť úplne.



## Splnené úlohy

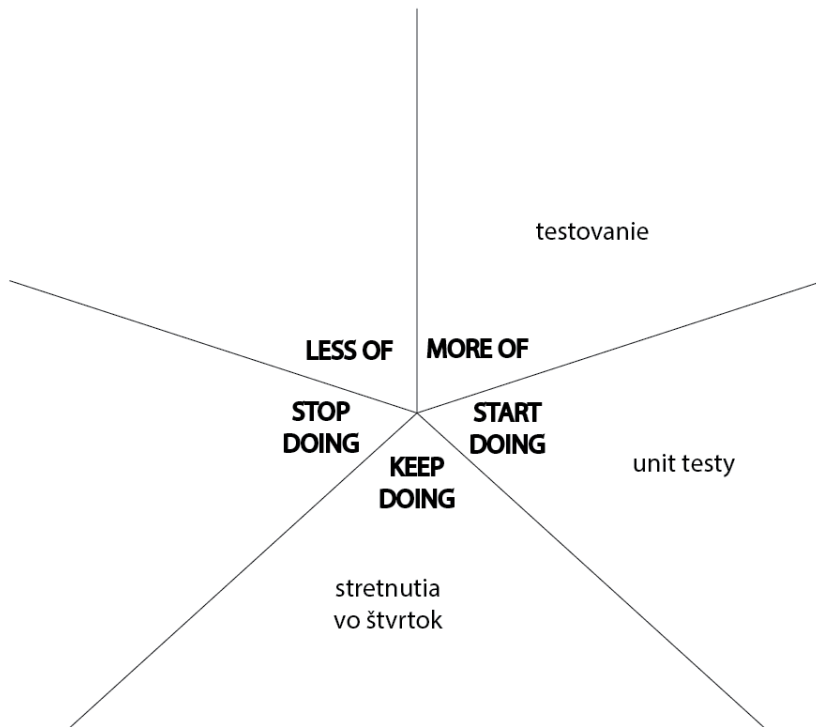
SB-75	As a Developer, I want mock server to simulate instructions for brewing	Story	KOMUNIKÁCIA SO ...	DONE	LN
SB-97	As a Developer, I want easier startup of every app	Story	SPLNENIE PODMIE...	DONE	PS

## Nesplnené úlohy

SB-78	As a Student, I want to keep documentation updated	Story	SPLNENIE PODMIE...	IN PROGRESS	PS
SB-82	As a User, I want to create a new recipe	Story	ZABEZPEČENIE AUT...	IN PROGRESS	MN
SB-83	As a User, I want to use a simple recipe to test if brewing works	Story	KOMUNIKÁCIA SO ...	IN PROGRESS	PS
SB-13	As a Team, we want to have a team webpage	Story	SPLNENIE PODMIE...	IN PROGRESS	MV

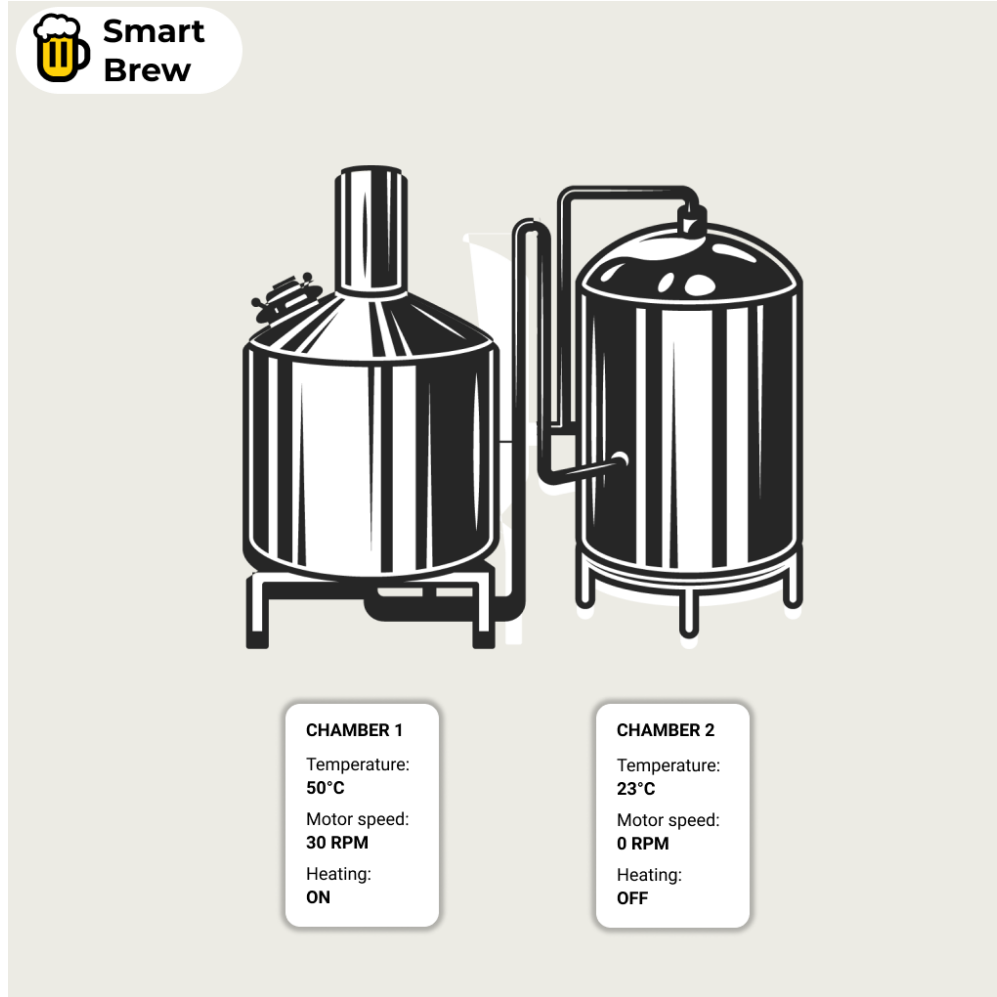
## Poznámky do budúcnosti

Nedávať si až tak veľmi náročné ciele, lebo sa to potom nestíha spraviť a členovia tímu strácajú motiváciu.



## 3.4.5 Návrh GUI

### Prehľadová obrazovka



**MENU**

#### INGREDIENTS

##### Fermentables:

5 kg **American - Pale 2-Row**  
5.6 kg **American - Caramel / Crystal 60L**

##### Hops:

0.4 oz **Magnum (Pellet)**  
5.0 oz **Perle (Pellet)**

##### Yeast:

1 **Fermentis - Safale - American Ale Yeast US-05**

##### Other:

1 **Crush whirfloc Tablet**

**Start brewing**

SETUP

Motor speed 30 RPM

Temperature 70 C

Motor speed 0 RPM

Valve OPEN

FINISH

# Výber receptu

<- back

## American Pale Ale

### Ingrediencie

#### Fermentables:

- 5 kg American - Pale 2-Row
- 5.6 kg American - Caramel / Crystal 60L

#### Hops:

- 0.4 oz Magnum (Pellet)
- 5.0 oz Perle (Pellet)

#### Yeast:

- 1 Fermentis - Safale - American Ale Yeast US-05

#### Other:

- 1 Crush whirlfoc Tablet

### Postup

#### Fermentovanie

Motor speed	30 RPM
Set temperature	70 C
Motor speed	0 RPM
Transfer liquids	-

#### Hopping

Motor speed	30 RPM
Temperature	70 C
Motor speed	0 RPM
Valve	OPEN

## Recipes

MENU

- American Pale Ale
- American Ale
- German Lager
- American Pale Ale
- American Pale Ale
- American Ale
- German Lager
- American Pale Ale
- American Pale Ale
- American Ale
- German Lager
- American Pale Ale

Edit

Load recipe

# Vytváranie receptu (časť 1.)

## American Pale Ale

### Ingredients

#### Fermentables:

[+ Add ingredient](#)

#### Hops:

[+ Add ingredient](#)

#### Yeast:

[+ Add ingredient](#)

#### Other:

[+ Add ingredient](#)

MENU

Next step

Cancel

## Vytváranie receptu (časť 2.)

Fermentacia

**Set temperature** 🗑️

Chamber 1 Value: 70 °C

**Transfer liquid** 🗑️

Chamber 1 to Chamber 2

**Unload ingredient** 🗑️

Fermentables

*Pomocný popisok  
djfnsdjfbsdf  
sdfsdfhfs*

Chmelenie

**Set temperature** 🗑️

Chamber 1 Value: 70 °C

**Transfer liquid** 🗑️

Chamber 1 to Chamber 2

**Unload ingredient** 🗑️

Fermentables

*Pomocný popisok  
djfnsdjfbsdf  
sdfsdfhfs*

## MENU

### American Pale Ale

#### INGREDIENTS

**Fermentables:**

5 kg American - Pale 2-Row

5.6 kg American - Caramel / Crystal  
60L

**Hops:**

0.4 oz Magnum (Pellet)

5.0 oz Perle (Pellet)

**Yeast:**

1 Fermentis - Safale -  
American Ale Yeast US-05

**Other:**

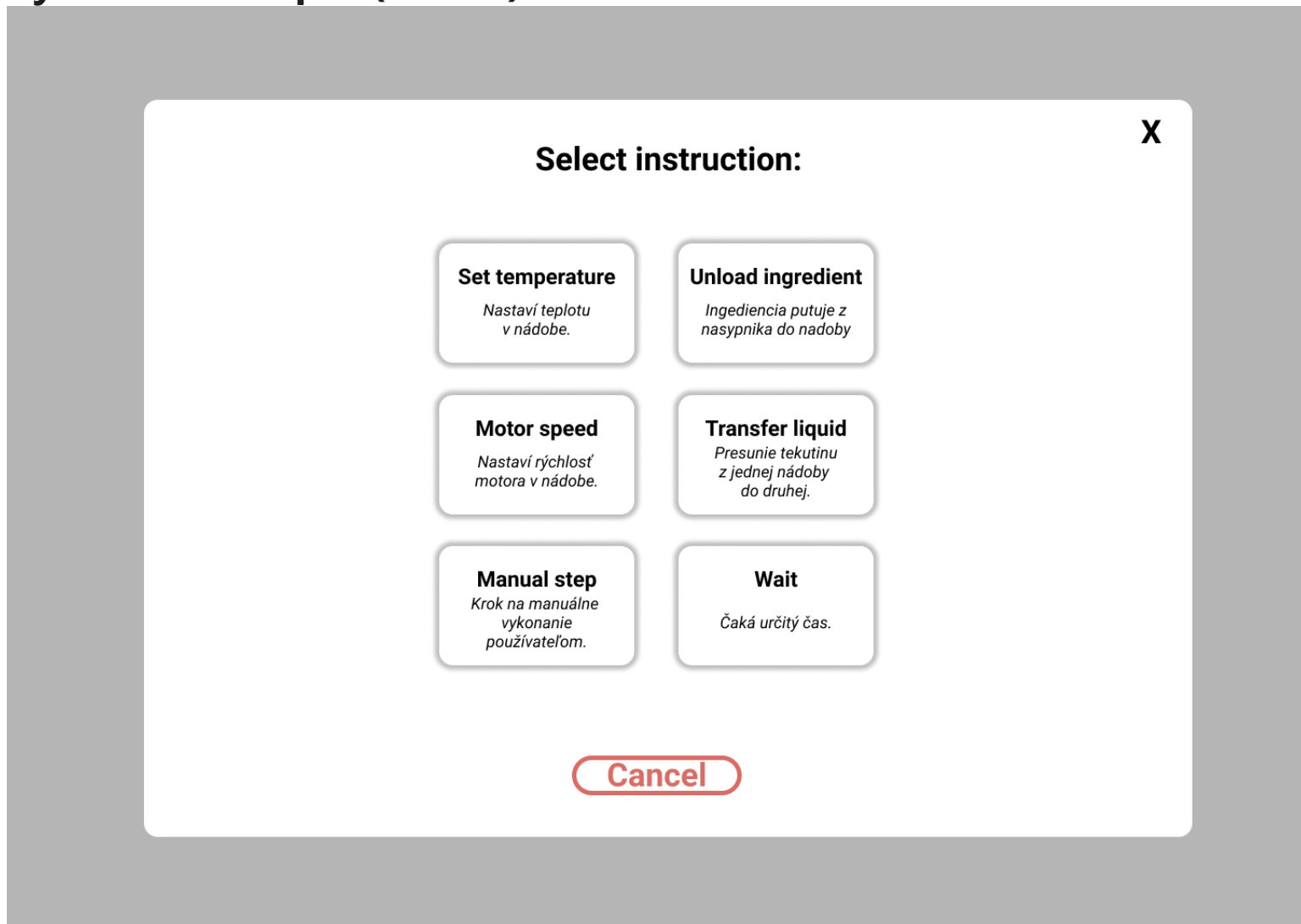
1 Crush whirlfoc Tablet

Save

Previous

Cancel

## Vytváranie receptu (časť 2.) - detail



Last updated on **14. 12. 2021** by **Peter Stríž**

# 3.5 Krušovice

## 3.5.1 Kedy?

**Od:** 15.11.2021

**Do:** 29.11.2021

## 3.5.2 Cieľ šprintu

Dokončiť nedokončené story z Corgoň, ktoré sa týkali prototypu tvorby a spúšťania receptov.

## 3.5.3 Obsah šprintu

Hlavnou časťou bolo dokončenie vytvárania receptov z GUI a následné použitie týchto receptov na spustenie varenia. Logika na backende už bola čiastočne funkčná, ale aj tú bolo treba upraviť a dokončiť.

### **[Krušovice] As a User, I want to create a new recipe, so that I can use it to brew beer**

Dokončenie vytvárania nového receptu. Implementácia návrhu inštrukcií a návrhu ingrediencií.

### **[Krušovice] As a User, I want to use a simple recipe, so that I can test if brewing works**

Načítanie receptu a následne jeho spustenie. Plus bolo potrebné otestovať celý proces - všetko robí to, čo má. Taktiež bolo treba vytvoriť moduly, aby sme ich mohli otestovať. Žiaľ sa nepodarilo zrealizovať stretnutie na Strojníckej fakulte, lebo nám to pandemická situácia neumožnila.

## As a Student, I want to keep documentation updated, so that I can pass the subject

Klasická úloha, v rámci ktorej sa vypisujú zápisnice, šprinty a podobné veci dokumentačného charakteru. Ďalej sme úspešne odovzdali priebežné semestrálne odovzdanie, kde bolo potrebné vytvoriť jednotný dokument z aktuálnej verzie dokumentácie. Takisto bolo potrebné upraviť celú dokumentáciu, aby obsahovala všetky potrebné náležitosti.

## [Zlatý bažant] As a Team, we want to have a team webpage, so that We can present our work

V tomto šprinte sme konečne dokončili našu webovú stránku. Je tam už aj text a aj obrázky každého člena tímu.

### 3.5.4 Výsledky šprintu

Nástroj Jira nám ponúka nasledujúce vizualizácie hlavných stories a taskov.

Podarilo sa nám dokončiť úlohy z minulých šprintov. Niektoré veci sme síce ešte nestihli otestovať a niektorá funkcionálna nefunguje úplne správne, ale hlavný cieľ sa podarilo dosiahnuť.

Žiaľ sa nepodarilo zrealizovať stretnutie na Strojníckej fakulte, lebo nám to pandemická situácia neumožnila. To znamená, že sme nemohli úplne otestovať, či moduly fungujú tak, ako majú.

### Splnené úlohy

Key	Summary	Issue type	Epic	Status	Assignee
SB-13	As a Team, we want to have a team webpage	Story	SPLNENIE PODMIE...	DONE	MV
SB-82	As a User, I want to create a new recipe	Story	ZABEZPEČENIE AUT...	DONE	MN
SB-83	As a User, I want to use a simple recipe to test if brewing works	Story	KOMUNIKÁCIA SO ...	DONE	PS
SB-78	As a Student, I want to keep documentation updated	Story	SPLNENIE PODMIE...	DONE	PS
SB-112	Keep DB schema updated	Task	ZABEZPEČENIE AUT...	DONE	MK

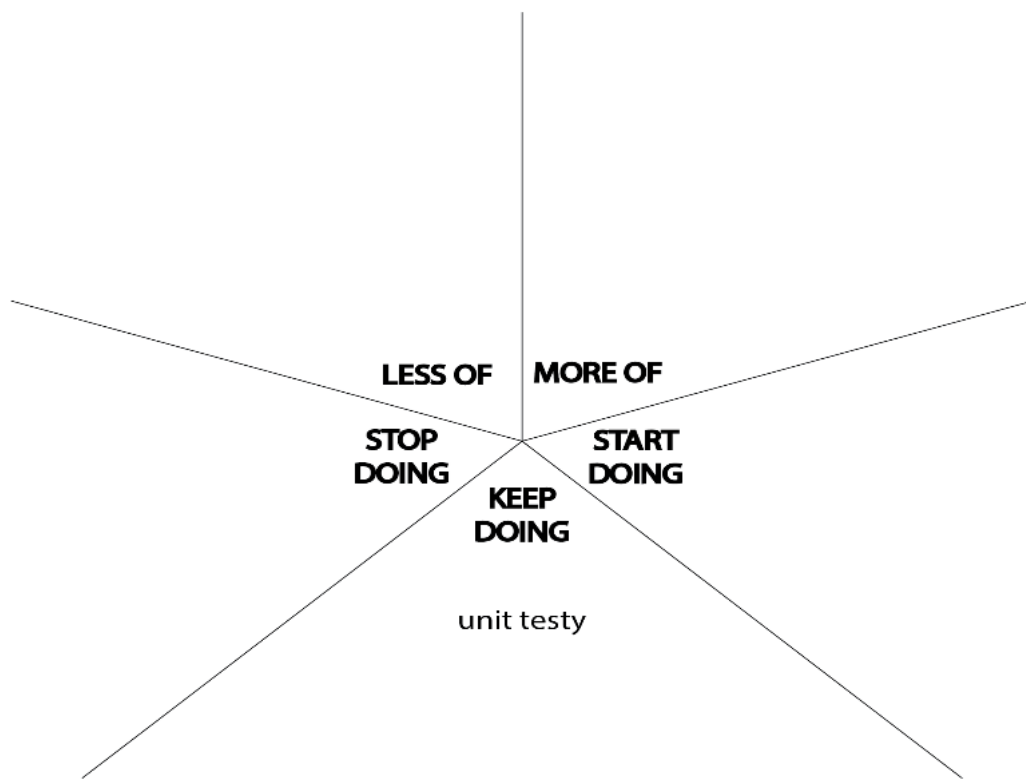


# Nesplnené úlohy

Všetky story sa dokončili.

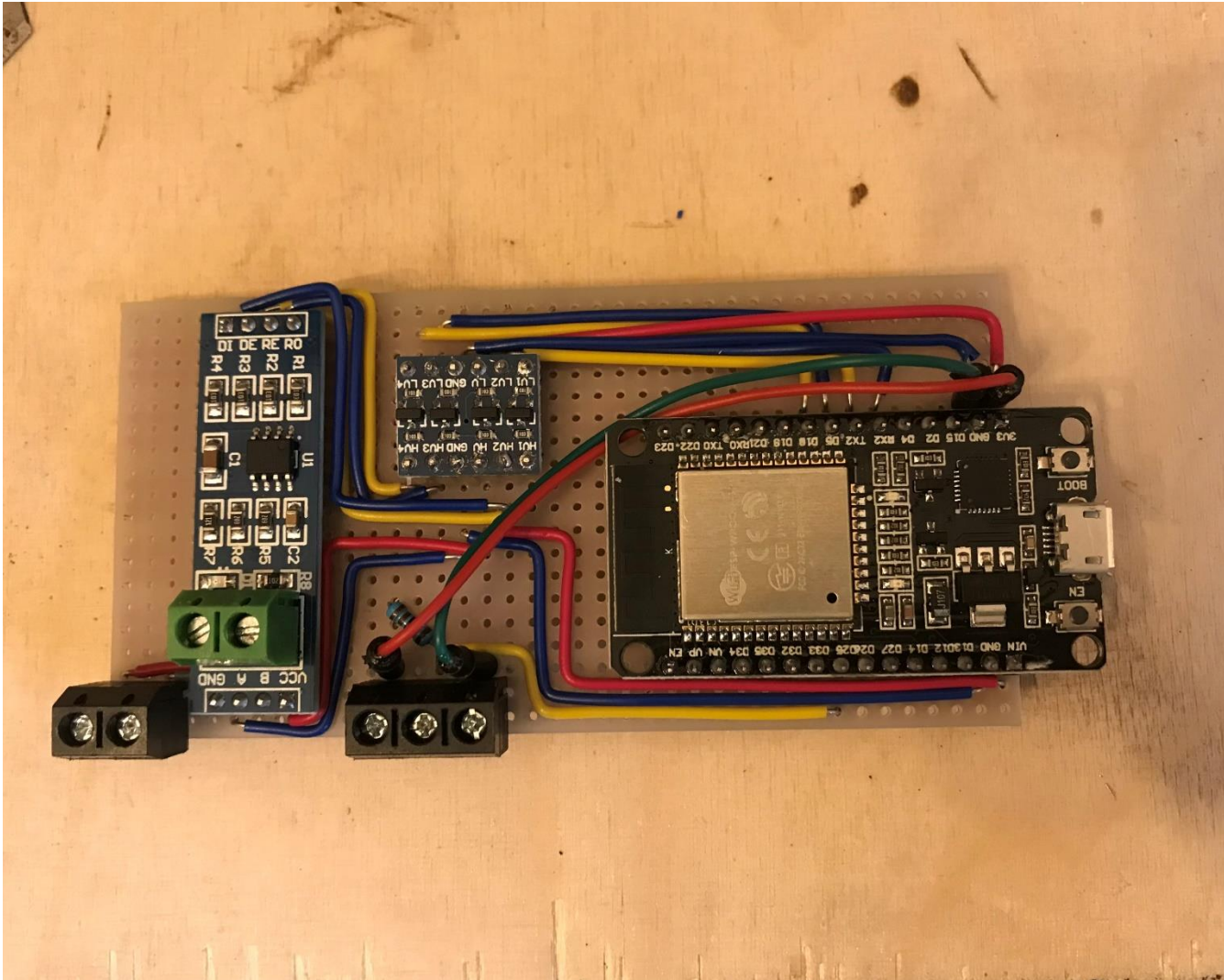
## Poznámky do budúcnosti

Viac pracovať inkrementálne po malých krokoch, aby sa vedeli ostatní členovia skorej zapojiť do práce.



## 3.5.5 Moduly

Modul, ktorý podporuje teplomer a motor



# 3.6 Šariš

## 3.6.1 Kedy?

**Od:** 29.11.2021

**Do:** 15.12.2021

## 3.6.2 Cieľ šprintu

Cieľom bolo vedieť zobrazovať informácie o aktuálnom recepte, ktoré sa používa na varenie. Takisto bolo potrebné doladiť ešte vytváranie receptov, aby fungovalo naprieč celým systémom.

## 3.6.3 Obsah šprintu

Hlavným obsahom bolo dokončiť celú API týkajúcu sa komunikácie jednotlivých častí systému. Išlo hlavne o používateľské rozhranie a aplikačný server. Takisto bolo treba pridať podporu pre nové inštrukcie (čakanie a manuálna inštrukcia) aj v rámci modulov a aj testovacieho serveru.

Problém, ktorý bol potrebné odstrániť, bolo nefunkčné ORM Prisma na Raspberry Pi, ktoré nefunguje dobre na zariadeniach 32-bit ARM (momentálne tam nemá oficiálnu podporu).

Ďalej sme sa zamerali na semestrálne odovzdanie, v rámci ktorého sme museli pripraviť všetky dokumenty, aby boli v správnom formáte a aby obsahovali všetky potrebné náležitosti.

## **As a User, I want to finish creating recipe, so that I can use it to brew beer**

Doladenie komunikácie medzi všetkými časťami systému. Ďalej bolo treba pridať prvky na lepšiu navigáciu v používateľskom rozhraní.

## **As a User, I want to see current state of brewing, so that I can better understand what instruction is being executed right now**

Zobrazenie aktuálneho stavu receptu. Takisto bolo treba upraviť celkové zobrazovanie receptov aj pri výbere receptu, a aj pri varení.

## **As a User, I want to delete recipe, so that I have better overview of recipes**

Vymazanie receptu pomocou soft delete. Pridať zmazanie na Frontende a upraviť Backend, aby posielal používateľom iba nezmazané recepty. Keď bude v budúcnosti história, budeme vedieť zobrazit' recept ktorý sa použil, ale mohol byť medzičasom zmazaný.

## **As a Student, I want to keep documentation updated, so that I can pass the subject**

Klasická úloha, v rámci ktorej sa vypisujú zápisnice, šprinty a podobné veci dokumentačného charakteru. Bolo potrebné poupraviť viaceré časti dokumentácie, ktoré sme dostali v rámci spätnej väzby k odovzdaniu v polovici semestra.

### **3.6.4 Výsledky šprintu**

Nástroj Jira nám ponúka nasledujúce vizualizácie hlavných stories a taskov.

Opäť sa nepodarilo zrealizovať stretnutie na Strojníckej fakulte, lebo nám to pandemická situácia neumožnila. To znamená že sme nemohli úplne otestovať, či moduly fungujú tak, ako majú, a tiež či spustenie receptu robí všetko čo má.

Počas šprintu sme narazili na problém, že nám nefunguje ORM Prisma na Raspberry Pi. Dôvodom je 32-bit ARM, ktorý nieje oficiálne podporovaný Prismou. Do budúcnosti rozmýšľame, že ak sa nám nepodarí problém opraviť, budeme nútení zmeniť ORM, čo bude veľmi zložité.

# Splnené úlohy

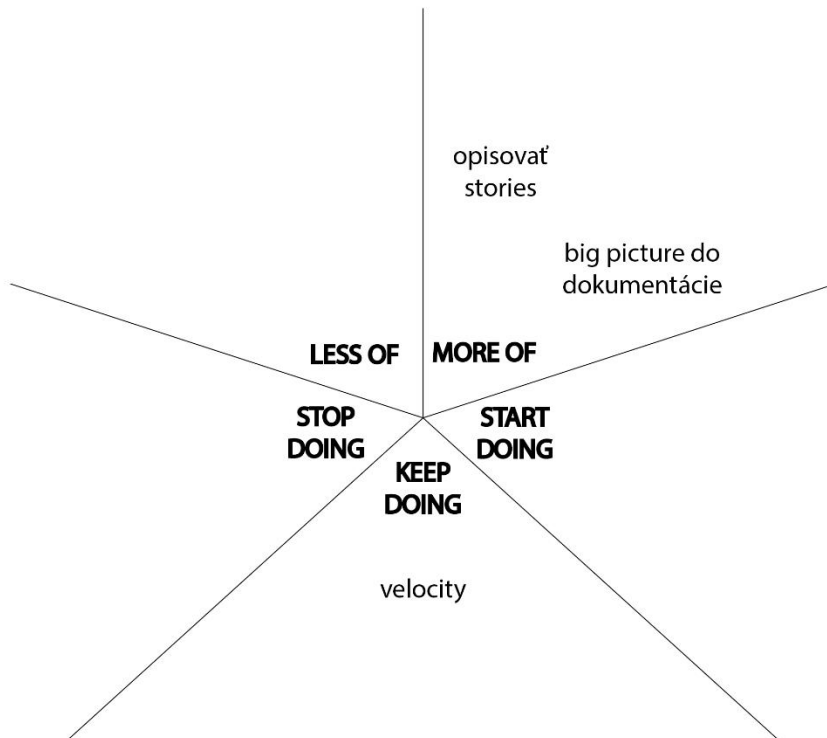
Key:	Summary:	Issue type:	Epic:	Status:	Assignee:
SB-116	As a Student, I want to keep documentation updated, so that I can pass the subject	Story	SPLNENIE PODMIE...	DONE	PS
SB-118	As a User, I want to finish creating recipe, so that I can use it to brew a beer	Story	ZABEZPEČENIE AUT...	DONE	AF
SB-119	As a User, I want to see current state of brewing, so that I can better understand what ...	Story	VIZUALIZOVANIE S...	DONE	MK
SB-120	As a User, I want to delete recipe, so that I have better overview of recipes	Story	ZABEZPEČENIE AUT...	DONE	LN

# Nesplnené úlohy

Stále sa nám nepodarilo fixnúť problém s ORM Prisma na Raspberry Pi.

Key:	Summary:	Issue type:	Epic:	Status:	Assignee:
SB-130	Fix BE docker for RPI	Task	ZABEZPEČENIE AUT...	IN PROGRESS	MK

# Poznámky do budúcnosti



# **4 Metodiky**

# 4.1 Komunikácia tímu

## 4.1.1 Kanály

Na komunikáciu sú určené viaceré kanály:

- ♦ Facebook Messenger
- ♦ Discord
- ♦ Google Meet

### Messenger

Tím cez Messenger komunikuje vzniknuté problémy a situácie, ktoré treba **rýchlo vyriešiť**. Tiež si cez daný kanál dohaduje stretnutia.

Tento kanál je primárny na oznamovanie dôležitých informácií pre celý tím.

### Discord

Tím má na Discorde vlastný server pre štruktúrovanú komunikáciu. Server má viacero textových kanálov:

- ♦ **General** - všeobecná komunikácia
- ♦ **Termíny** - informácie o dôležitých termínoch pre projekt
- ♦ **Dokumenty** - dokumenty / linky na dokumenty súvisiace s projektom
- ♦ **Credentials** - súkromné údaje pre pripájanie na používané zariadenia
- ♦ **Problémy** - riešenie vyskytnutých problémov pri vývoji

Server má tiež aj voice kanál, cez ktorý sa uskutočňujú **stand-up hovory**.

### Google Meet

Tím cez Google Meet uskutočňuje stretnutia s vedúcim tímu, ktoré sa vzhľadom na pandemickú situáciu uskutočňujú online. Pre lepšie sústredenie členov a celkovú atmosféru hovoru majú všetci zúčastnení zapnuté web kamery.

# 4.2 Písanie dokumentácie pre riadenie projektu

## 4.2.1 Čo je potrebné zapisovať

Po každom stretnutí tímu s vedúcim vypracuje vybraný člen tímu **zápisnicu zo stretnutia**.

Po každom ukončení šprintu vypracuje scrum master **vyhodnotenie šprintu**.

## 4.2.2 Forma

Zápisnice/vyhodnotenia sú vypracované podľa predpripravených šablón:

- Šablóna zápisnice
- Šablóna šprintu

## 4.2.3 Usmernenia

### Zápisnice

Je na zodpovednosti správcu dokumentácie, aby boli v dokumentácii aktuálne zápisnice a aby zápisnice boli zverejňované najneskôr do ďalšieho stretnutia (do týždňa).

Zápisnice by mali dodržiavať predpísaný formát, čo najviac sa to dá. Formát nie je striktný a nemusí sa dodržať, pokiaľ sa na stretnutí niektorá časť šablóny nepreberala.

### Šprinty

Je na zodpovednosti scrum mastera, aby boli v dokumentácii aktuálne vyhodnotenia šprintov a aby vyhodnotenia šprintov boli vypracované najneskôr do týždňa od uzavretia šprintu.



# 4.3 Metodika pre návrh front-endu

Za návrh front-endu sú zodpovední členovia implementujúci front-end (Alexandra Frniaková a Michaela Nemcová). K návrhu sa môžu pridať aj iní členovia tímu.

## 4.3.1 Nástroje

Pre návrh front-endu sa využíva nástroj Figma.

## 4.3.2 Usmernenia

Návrh front-endu začína pri low-fidelity návrhu, ktorý prechádza viacerými iteráciami na základe vzniknutej potreby alebo vstupu iných členov tímu. Postupnými iteráciami sa dopracuje až k high-fidelity návrhu, ktorý je aj implementovaný.

Pokiaľ sa iný člen tímu ako zodpovední chce vyjadriť k návrhu, resp. niečo zmeniť, musí do toho zapojiť zodpovedných členov tímu.

Rozhodnutia ohľadom návrhu nikdy nerobí len jeden člen tímu, vždy to musí odsúhlasiť aspoň jeden, ale najlepšie aspoň dvaja iní členovia.

*Last updated on 16. 12. 2021 by **Michaela Nemcová***

# 4.4 GitHub

Ako verzovací systém budeme používať GitHub po názvom organizácie `smart-brew`.

V rámci tejto GitHub organizácie budeme pracovať v nasledovných repozitároch:

- `documentation` - pre dokumentáciu
- `web` - pre tímovú stránku
- `frontend` - GUI pre používateľov
- `backend` - aplikačný server, ktorý je spojený s databázou a rieši požiadavky z frontendu cez REST API
- `websocket-module` - ESP32 modul (viac info `Moduly`)
- `module-mock-server` - testovací server, ktorý slúži ako náhrada modulu pre lokálny development (viac info `Testovací server`)
- `startup` - spustenie systému (viac info `Spustenie systému`)

## 4.4.1 Commit

Každý commit bude dodržiavať nasledujúci formát

### Formát commitov

```
typ(jira): popis
```

- `typ` je jedným z: `feature` `fix` `chore`
- `jira` je označenie Jira issue (napr. `SB-1`) - dôležité dodržať názov aby sa dalo linkovať automaticky medzi Jirou
- `popis` je krátky popis úkonu

### Ukážka commitu

```
+ feature(SB-2): Login page  
+ fix(SB-34): Missing files from upload page
```

## 4.4.2 Tvorba vetiev

Každá nová fičúra sa bude tvoriť na osobitnej vetve, pričom sa neskôr bude mergovať pomocou **Pull request** do main vetvy.

### Formát vetiev

```
typ(jira)/popis
```

- ♦ **typ** je jedným z: **feature**, **bug**, **hotfix**
- ♦ **jira** je označenie Jira issue (napr. **SB-1**)
- ♦ **popis** je krátky popis úkonu - všetko **malé písmená**, pričom slová sú **oddelené pomlčkou**

### Ukážka vetvy

```
+ Správne  
+ feature(SB-2)/login-page  
+ bug(SB-34)/missing-files-from-upload-page
```

```
- Nesprávne  
- feature(SB-2)/Login page # nesprávny formát popisu  
- bug(34)/missing-files-from-upload-page # nesprávne označenie Jira issue  
- bug(SB-34):missing-files-from-upload-page # ':' namiesto '/'
```

## 4.4.3 Pull request

Názov pre Pull request musí taktiež dodržať istý formát ako pre názvy **commitov**.

### Formát pull request

```
typ(jira)/popis
```

1. Ak sa v Github actions nachádza script, ktorý vykonáva automatický test, musí najprv **úspešne** prejsť,

aby sa mohli zmeny merge-núť

2. Ďalej je potrebné spraviť code review a zapísať tento fakt do Jiry
3. Až následne vykonať merge
4. Uzavrieť issue v Jire

*Last updated on **23. 11. 2021** by **Peter Stríž***

# 4.5 Jira

Pre prácu na projekte a trackovanie úloh využívame Jiru.

S Jirou pracujeme metodológiou Scrumu.

## 4.5.1 Úlohy

- ♦ **Epic**
  - Epic zahŕňa stories a tasky pre časť funkcionality projektu, ktorá prechádza naprieč celým systémom.
  - Spravidla sa na ňom podieľajú viacerí členovia tímu.
- ♦ **Story**
  - Story je stavané z pohľadu "Ako (niekto) chcem ..., aby som ...", napr. "Ako používateľ chcem vytvoriť nový recept, aby som ho mohol použiť pri varení piva."
  - Väčšinou sa na ňom podieľajú viacerí členovia tímu.
- ♦ **Task**
  - Task ponúka najnižšiu granularitu pre story. Ide už o konkrétne implementačné úlohy.
  - Pracuje na ňom vždy jeden člen tímu.

## Hodnotenie

Tím bodovo hodnotí zložitosť jednotlivých **taskov**. Tieto hodnotenia sa potom spočítajú a výsledok je použitý ako body pre dané story, ktorému tasky patria.

## 4.5.2 Prechody

- ♦ TO DO --> IN PROGRESS --> WAITING FOR REVIEW --> DONE
- ♦ WAITING FOR REVIEW --> IN PROGRESS

Pokiaľ člen tímu začne vykonávať pridelený task, presunie ho do **IN PROGRESS**.

Po dokončení tasku ho presunie do **WAITING FOR REVIEW** s tým, že mu ho musí iný (aspoň jeden) člen tímu odsúhlasiť. Ak ho neodsúhlasiť, tak sa môže presunúť naspäť do **IN PROGRESS** na revíziu.

## 4.5.3 Definition of DONE

Pre každý šprint je nastavený **Definition of DONE**. Ide o konkrétne story alebo viacero stories, ktoré keď sa podarí tímu splniť, šprint môže byť považovaný za úspešne splnený.

## 4.5.4 Zodpovednosti

S Jirou pracuje najmä scrum master:

- ♦ vytvára epics, stories a tasks
- ♦ dohliada na plnenie taskov a ich aktualizáciu v Jire
- ♦ uzatvára šprint po ukončení

Tasky môžu vytvárať aj iní členovia tímu po dohode so zvyškom tímu a vyhodnotením, že daný task je naozaj potrebný.

*Last updated on 15. 12. 2021 by Peter Stríž*

# 4.6 Metodika stretnutí

Tu je zapísaná základná organizácia stretnutí tímu, ktorej sa budeme snažiť držať.

## 4.6.1 Kedy?

**Pondelok** o 14:00 - konzultácie s vedúcim

**Štvrtok** o 20:30 (alebo inokedy po dohode) - krátky stand-up

## 4.6.2 Časti stretnutia a práce v tíme

Štruktúra ako by malo vyzeráť tímové stretnutie.

### Časť 1 - prezentácie

Každý člen v krátkosti oboznámi svojich kolegov na čom pracuje, aké urobil rozhodnutia a prečo ich urobil. Zároveň môže poukázať na miesto kde sa zasekol alebo potrebuje dorobiť niečo od iného člena tímu. Prezentujeme ideálne na projektore alebo zdieľame obrazovku.

### Časť 2 - voľná konzultácia

Tu sa riešia problémy, vymýšľajú sa nové veci atď.

### Časť 3 - problémy na úvahu

Táto časť prebieha neustále. Vždy keď nejaký člen chce aby sa všetci zamysleli nad spôsobom riešenia dákeho problému, povie to buď na stretnutí alebo to napíše na discord do chanelu **problémy**. Snažíme sa zapisovať všetko aj na discord.

## 4.6.3 Pravidlá

**Pravidlo 1** - V prípade, že chceme zasiahnuť do rozbehnutej diskusie, hlásime sa aby sme sa neprekrikovali počas stretnutí. Diskutujúca dvojica sa vždy posnaží dať hlásiacemu sa členovi priestor čo najskôr.

**Pravidlo 2** - Vždy sa snažiť predstaviť svoje riešenie problému ako návrh, ktorý sa môže upraviť v prípade, že to spadá pod špecializáciu, ktorú člen tímu vykonáva. Teda ak mám nariadenia databázy mám automaticky väčšie práva vykonávať rozhodnutia ohľadom databáz ak som zvažil, že sa nejedná o naozaj kritickú časť projektu. Snažíme sa byť čo najviac samostatní a prezentovať našu prácu kolegom, ktorých pripomienky musí zobrať prezentujúci člen tímu do úvahy. Preto je informačné stretnutie aspoň dvakrát do týždňa veľmi dôležité.

## 4.6.4 Iné usmernenia

Vzhľadom na pandemickú situáciu a dištančné vzdelávanie sa od 8.11. do odvolania uskutočňujú stretnutia online pomocou kanálov Google Meet / Discord (viď Metodika komunikácie).

*Last updated on 23. 11. 2021 by Peter Stríž*



# **5 Inžinierske dielo**

# 5.1 Úvod

Tento dokument obsahuje informácie k produktu. Poskytuje základné delenie na časti **Architektúra, Analýzy a Návody**.

## Architektúra

- ♦ **Api:** Spôsoby komunikácie (requesty a odpovede) medzi komponentami jednotlivých vrstiev.
- ♦ **Návrh databázy:** Štruktúra databázy vyjadrená diagramom.
- ♦ **Front-end:** Návrh používateľského prostredia a jeho opis
- ♦ **Moduly:** Opis fungovania modulu
- ♦ **Podporované údaje:** Zadefinované interfaces. Údaje z akých zariadení sa využívajú.
- ♦ **Podporované funkcie:** Zoznam podporovaných funkcií systémom a štruktúry inštrukcii.
- ♦ **Testovací server:** Opis funkcionality.

## Analýzy

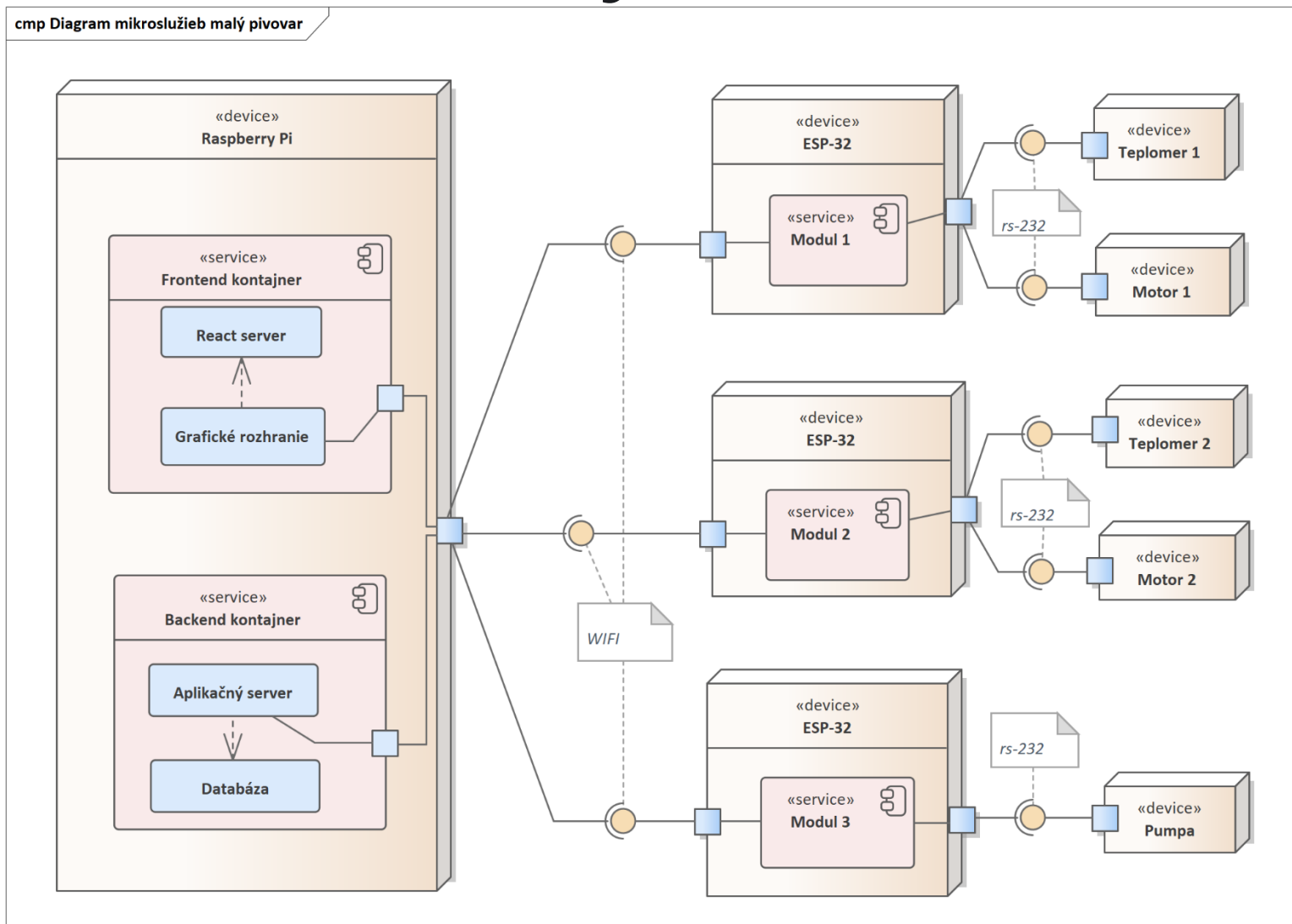
- ♦ **Brewery, modbus, SCADA:** analýza nástrojov využitých pri funkčnom pivovare daného typu

## Návody

- ♦ **Migrácie, Raspberry Pi, Spustenie pivovaru, Tímová VM, Tímový web + dokumentácia:** Návody k správne mu využívaniu daných nástrojov.

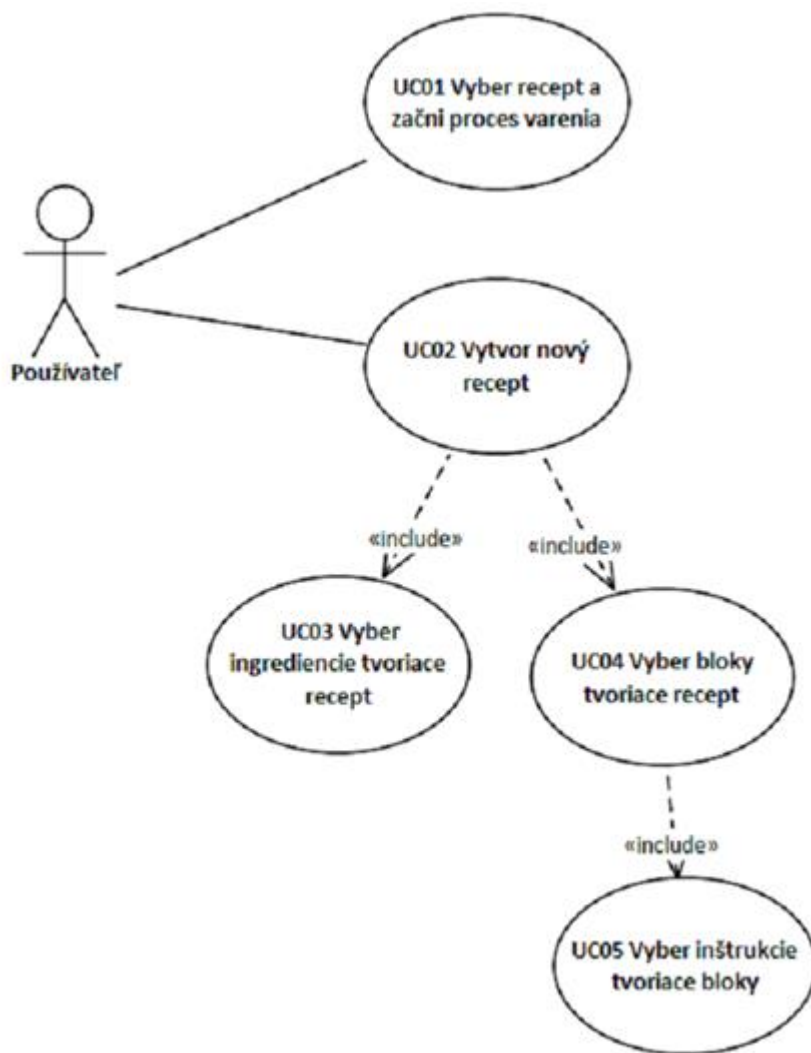
# 5.2 Pohľad na systém

cmp Diagram mikroslužieb malý pivovar



System pozostáva z mikroslužieb, ktoré medzi sebou komunikujú na základe API. Každá mikroslužba predstavuje jeden komponent. Využívame zariadenia **Raspberry Pi** a **ESP-32**. Na kontajnerizáciu v zariadení Raspberry Pi používame virtuálne prostredie **Docker**.

# 5.3 Use Cases



## 5.3.1 UC01 Vyber recept a začni proces varenia

### Hlavný scenár

1. Používateľ z hlavného menu zvolí možnosť Recipes
2. Aplikácia zobrazí okno Recipes, kde sa na pravom sidebare zobrazuje zoznam možných receptov
3. Používateľ klikne na ľubovoľný prvok zo zoznamu
4. Aplikácia v hlavnej časti obrazovky zobrazí podrobnosti o danom recepte
5. Používateľ klikne na tlačidlo Load recipe
6. Aplikácia zobrazí Overview page, kde sa na pravom sidebare zobrazujú informácie o recepte a tlačidlo Start brewing

7. Použivateľ klikne na tlačidlo Start brewing
8. Aplikácia zobrazí vyskakovacia okno s ingredienciami, ktoré treba pripraviť
9. Použivateľ klikne na tlačidlo Confirm
10. Aplikácia na pravom sidebare zobrazuje stav procesu varenia
11. Prípád použitia skončí

## 5.3.2 UC02 Vytvor nový recept

### Hlavný scenár

1. Použivateľ zvolí z hlavného menu možnosť Recipes
2. Aplikácia zobrazí okno Recipes, kde sa na pravom sidebare nachádza tlačidlo Create new
3. Použivateľ klikne na tlačidlo Create new
4. Vyvolá sa prípad UC03 Vyber ingrediencie tvoriace recept
5. Vyvolá sa prípad UC04 Vyber bloky tvoriace recept
6. Aplikácia zobrazí vyskakovacie okno pre potvrdenie uloženia receptu
7. Použivateľ potvrdí uloženie receptu tlačidlom Confirm
8. Aplikácia zobrazí okno Recipes
9. Prípád použitia skončí

## 5.3.3 UC03 Vyber ingrediencie tvoriace recept

### Hlavný scenár

1. Aplikácia zobrazí okno Ingredients
  2. Používateľ vyplní pole s názvom receptu
  3. Používateľ klikne na tlačidlo Add new ingredient korešpondujúce k vybranej kategórii ingrediencie
  4. Aplikácia zobrazí nové pole pre zadanie informácií o recepte
  5. Používateľ zadá informácie: počet, jednotku a opis ingrediencie
- Alternate: 5a. Alternatívny scenár
6. Používateľ stlačí tlačidlo Next Step
  7. Prípado použitia skončí

### Alternatívny scenár

1. Používateľ klikne na tlačidlo - korešpondujúce s vybranou kategóriou ingrediencie
2. Aplikácia odstráni danú ingredienciu zo zoznamu
3. Hlavný scenár pokračuje krokom 6

## 5.3.4 UC04 Vyber bloky tvoriace recept

### Hlavný scenár

1. Aplikácia zobrazí okno Instructions
  2. Používateľ klikne na modrý znak plus na pridanie bloku
  3. Aplikácia pridá nový blok do receptu
  4. Používateľ zadá názov bloku
- Alternate: 4a. Alternatívny scenár
5. Vyvolá sa prípad UC05 Vyber inštrukcie tvoriace bloky
  6. Používateľ stlačí tlačidlo Save recipe
  7. Prípado použitia končí

### Alternatívny scenár

1. Používateľ klikne na červené X označujúce vymazanie bloku
2. Aplikácia odstráni blok
3. Hlavný scenár pokračuje krokom 6

## 5.3.5 UC05 Vyber inštrukcie tvoriace bloky

### Hlavný scenár

1. Aplikácia zobrazí blok s možnosťami na pridanie alebo odobranie inštrukcie
  2. Používateľ klikne na zelený znak plus
  3. Aplikácia zobrazí vyskakovacie okno s možnosťami pre výber inštrukcii
  4. Používateľ kliknutím vyberie požadovanú inštrukciu
  5. Aplikácia danú inštrukciu pridá na požadované miesto v rámci bloku
  6. Používateľ vyplní údaje inštrukcie
- Alternate: 6a. Alternatívny scenár
7. Prípád použitia skončí

### Alternatívny scenár

1. Používateľ sa rozhodne kliknúť na červené X pre vymazanie inštrukcie
2. Aplikácia odstráni inštrukciu
3. Hlavný scenár pokračuje krokom 7

## 5.4 Ciele

Cieľom projektu je vytvoriť ucelené riešenie na automatizovanie procesov v priemysle. Projekt je vykonávaný v spolupráci so Strojníckou fakultou STU, ktorá z veľkej časti zastrešuje materiálne požiadavky a fyzické prevedenie pivovaru. Pivovar využíva dve prepojené nádoby, pričom v každej prebieha iná fáza varenia. Naša práca bude prebiehať nad údajmi z pivovaru, konkrétne nad rôznymi efektormi (motory, násypníky, ventily,...) a senzormi (teplomer, tlakomer,...). Centrálnym komponentom systému je Raspberry Pi, ktoré bude poskytovať zázemie pre sieť senzorov a efektorov. Automatizovaná výroba piva pozostáva z pridania nového receptu (respektíve výberu už existujúceho receptu), procesu výroby piva, priebežného zobrazovania priebehu varenia a vytvorenia zápisu o ukončenom procese.

Komunikácia bude prebiehať medzi niekoľkými vrstvami. Moduly periodicky odovzdávajú svoje údaje back-endu, ktorý ich spracuje (niektoré si ukladá v dočasnej pamäti a niektoré ukladá do databázy). Taktiež si periodicky pýtajú úlohy, ktoré majú vykonávať. Rovnakú úlohu vykonáva aj front-end. Pri procese varenia si od back-endu periodicky pýta stav varenia, ktorý zobrazuje používateľovi. Pri tvorení nového receptu zase posiela údaje na back-end, kde sa zapisujú do databázy.

Komunikácia medzi front-endom a back-endom sa bude riešiť pomocou REST API. Na komunikáciu sa bude využívať formát JSON. Štruktúra JSON-u sa bude meniť podľa vrstvy komunikácie a obsahu správy.

Kroky (inštrukcie) pre jednotlivé moduly sa budú posielat' a spracovávať sériovo. Teda inštrukcie na moduly sa budú zasielať v osobitných správach. Frontend si ale tieto kroky bude triediť a spájať do blokov podľa určitých etáp výroby pre lepšiu user experience.

Všetky potrebné údaje o receptoch a jednotlivých vareníach sa budú ukladať do databázy. To umožní znovupoužitie už vytvoreného receptu a vygenerovanie protokolu z varenia.

### 5.4.1 Ciele na prvý semester

Pripraviť funkčný prototyp tak, aby vykonával základné úlohy. V užívateľskom prostredí si používateľ bude vedieť vytvoriť nový recept. Taktiež bude vedieť začať proces varenia, a na obrazovke zobrazit' jeho priebeh. Backend spracuje dané údaje, ktoré získa z modulov, a pošle ich na frontend - používateľ bude môcť kontrolovať stav varenia. Stav varenia sa tiež bude logovať do databázy (čo sa neskôr môže použiť na vytváranie protokolu o varení). Bude funkčná aspoň určitá podmnožina modulov (nakoľko nie sú niektoré zariadenia dostupné, nedá sa s nimi zatiaľ pracovať). Front-end a back-end bude vedieť spracovávať všetky procesy, funkcionality zatiaľ nefunkčných modulov budeme simulovať.



# **6 Návody**

# 6.1 Migrácie

V projekte využívame ORM Prisma. Pre vytvorenie migrácie treba upraviť súbor `schema.prisma` podľa potreby.

Potom vytvoriť novú migráciu, ktorá sa automaticky aplikuje na databázu na adrese `localhost:5432`.

Ak nie sú vykonané zmeny, tak sa nevytvorí nová migrácia iba sa aplikujú čakajúce migrácie.

## Vytvorenie novej migrácie alebo aplikovanie migrácií

```
yarn migrate
```

Pri použití príkazu `yarn migrate` sa databáza naseeduje. Ak však už bola raz naseedovaná, pri opakovaní príkazu sa už neseeduje. Preto ak sa napríklad doplní do súboru `seed.js` ďalší záznam, je potrebné seedovanie spraviť samostatne.

## Seedovanie databázy

```
yarn seed
```

Záznamy, ktoré už v databáze sú sa opakovane nepridávajú.

*Last updated on 16. 12. 2021 by **Michaela Nemcová***

# 6.2 Raspberry pi

## 6.2.1 Pripojenie do rpi

### ⚠ VHDNÝ TERMINÁL A REŠTART

Odporúčam použiť `git bash` alebo `wsl` terminál na prístup do rpi. Taktiež by malo byť spojenie odolné voči reštartom. To znamená, že reštart virtuálneho stroja alebo rpi by nemal nič znefunkčniť a po čase cca `1 min` by malo byť pripojenie znova dostupné.

### 1. krok

Ak nemáš vygenerované ssh kľúče treba ich vygenerovať bez hesla (treba len stále štláčať enter).

Vygenerovanie ssh kľúčov

```
ssh-keygen
```

### 2. krok

### ⚠ PODMIENKA POUŽITIA PRÍKAZU

Ak budeme mať heslo do virtuálneho stroja `heslo="neznáme"` je možné použiť príkaz na skopírovanie svojho verejného kľúča do `authorized_keys`.

Ak je heslo neznáme, je potreba poslať nejakému členovi napr. (Marek Vajda alebo Peter Stríž) svoj verejný kľúč, aby ho nahrali na virtuálnom stroji do `authorized_keys`. V prípade, že skôr spomenuté možnosti nie sú dostupné, je potrebné sa prihlásiť pomocou privátneho kľúča do virtuálneho stroja, ktorý treba zadať do príkazu `ssh -i [cesta ku privátnemu kľúču] user@ip`. Následne je možné nahrat' svoj kľúč samostatne.

Skopírovanie verejného kľúča do virtuálneho stroja

```
ssh-copy-id ubuntu@team06-21.studenti.fiit.stuba.sk
```

### 3. krok

Pripojenie príkazom ssh, ktorý používa tzv. `jump host` na pripojenie sa do reverzného terminálu rpi. Port je `2222` alebo `2224`.

pripojenie do rpi cez ssh jumphost

```
ssh -J ubuntu@team06-21.studenti.fiit.stuba.sk:22 -p 2222 pi@localhost
```

### **i** ALTERNATÍVNE PRIPOJENIE

Alternatívne je možné sa pripojiť najprv cez ssh do virtuálneho stroja. A následne do reverzného príkazového riadka na porte **2222** alebo **2224**.

alternatívne príkazy

```
ssh -p 22 ubuntu@team06-21.studenti.fiit.stuba.sk  
ssh -p 2222 pi@localhost
```

## 6.2.2 Pripojenie pomocou vnc

Na pripojenie pomocou vnc si treba stiahnuť **vnc viewer**. A následne sa prihlásiť do účtu. Rpi sa následne zobrazí v účte tímu.

prihlasovací email

marek.vajda009@gmail.com

password

tp062122

## 6.2.3 Spustenie browsera do fullscreenu na Rpi

Rpi je nastavené tak aby sa po spustení otvoril webový frontend v prehliadači. Momentálne sa v grafickom rozhraní spustí len webový prehliadač `chromium` bez ostatného grafického rozhrania. Toto nastavenie sme docielili v súbore `/home/pi/.config/lxsession/LXDE-pi/autostart`. Ak chceme, aby sa otvorilo po štarte aj štandardné grafické rozhranie, musíme nasledujúci príkaz zadať do súboru `/etc/xdg/lxsession/LXDE-pi/autostart`. Alebo ak nefunguje, treba použiť `sudo nano /etc/xdg/lxsession/LXDE-pi/autostart` súbor.

príkaz na spustenie prehliadaca vo fullscreen

```
@chromium-browser localhost --start-fullscreen --kiosk
```

## 6.2.4 Ako funguje pripojenie

Pripojenie funguje pomocou reverzného ssh spojenia. Rpi v intervale `1 min` spúšťa pomocou `sudo crontab -e` skript, ktorý kontroluje vytvorenie reverzného ssh spojenia na virtuálnom stroji.

`sudo crontab -e` príkaz

```
*/1 * * * * sudo . /create_ssh_tunnel.sh > tunnel.log 2>&1
```

Skript na vytvorenie reverzného konzolového spojenia sa pripája na virtuálny stroj `team06-21.studenti.fiit.stuba.sk`. Port `2224` na virtuálnom je následne tunelovaný ako port `22` na Raspberry pi.

## create\_ssh\_tunnel

```
#!/bin/bash
createTunnel() {
  /usr/bin/ssh -N -f -R 2222:localhost:22 ubuntu@team06-21.studenti.fiit.stuba.sk
  if [[ $? -eq 0 ]]; then
    echo Tunnel to jumpbox created successfully
  else
    echo An error occurred creating a tunnel to jumpbox. RC was $?
  fi
}
/bin/pidof ssh
if [[ $? -ne 0 ]]; then
  echo Creating new tunnel connection
  createTunnel
fi
```

## 6.2.5 Zoznam príkazov na nastavenie rpi komunikácie

### príkazy

```
# add ssh file to filesystem root on boot partition of rpi to enable ssh
# add wpa_supplicant file to rpi root partition boot
# this for automatic wifi connection
# example of wpa_supplicant file(file is deleted after first reboot):
country=SK # Your 2-digit country code
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
network={
    ssid="marek-nb"
    psk="7606y;69"
    key_mgmt=WPA-PSK
}
#update and upgrade of new rpi
sudo apt-get update && sudo apt-get upgrade

#docker
curl -sSL https://get.docker.com | sh

#pip
sudo apt-get install libffi-dev libssl-dev
sudo apt install python3-dev
sudo apt-get install -y python3 python3-pip

#docker-compose
sudo pip3 install docker-compose

# docker on startup
sudo systemctl enable docker

# make script for reverse shell
# change 2222 to wanted port if needed
# this command should be run from rpi for adding to known hosts
# /usr/bin/ssh -N -f -R 2224:localhost:22 ubuntu@team06-21.studenti.fiit.stuba.sk
# after that you need to connect to rpi with ubuntu
# ssh -p 2222 pi@localhost
```



```

sudo nano create_ssh_tunnel
#script
#!/bin/bash
createTunnel() {
  /usr/bin/ssh -N -f -R 2224:localhost:22 ubuntu@team06-21.studenti.fiit.stuba.sk
  if [[ $? -eq 0 ]]; then
    echo Tunnel to jumpbox created successfully
  else
    echo An error occurred creating a tunnel to jumpbox. RC was $?
  fi
}
/bin/pidof ssh
if [[ $? -ne 0 ]]; then
  echo Creating new tunnel connection
  createTunnel
fi

chmod +x create_ssh_tunnel

# make new sudo cron schedule for script
crontab -e
*/1 * * * * ./create_ssh_tunnel > tunnel.log 2>&1

ssh-keygen
# ssh to ubuntu ssh copy id
ssh ubuntu@team06-21.studenti.fiit.stuba.sk
vim ~/.ssh/authorized_keys # add id_rsa.pub generated by rpi + increment identifier at
the pi@raspberrypiv[version-number]
#unlock port for listening firevall
sudo netstat -tulpn # check listening ports
sudo ufw allow 2222

# ssh keygen for ssh deploy keys one for frontend and one for backend
ssh-keygen -t rsa -f ~/.ssh/frontend -C "frontend"
ssh-keygen -t rsa -f ~/.ssh/backend -C "backend"
cat ~/.ssh/frontend.pub

cat ~/.ssh/backend.pub

# add deploy keys to repos
# In the server's SSH configuration file (usually ~/.ssh/config), add an alias entry for
each repository.

```

```
cat <<EOT >> ~/.ssh/config
```

```
Host github-frontend
```

```
HostName github.com
```

```
User git
```

```
IdentityFile ~/.ssh/frontend
```

```
Host github-backend
```

```
HostName github.com
```

```
User git
```

```
IdentityFile ~/.ssh/backend
```

```
EOT
```

```
git clone github-frontend:smart-brew/frontend.git
```

```
git clone github-backend:smart-brew/backend.git
```

```
# create startup file
```

```
chmod +x startup
```

```
#!/bin/bash
```

```
cd ~/frontend
```

```
sudo docker-compose up
```

```
cd ~/backend
```

```
sudo docker-compose up
```

```
#autostart of chromium
```

```
sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

```
#add
```

```
@chromium-browser localhost --start-fullscreen --kiosk
```

Last updated on **16. 12. 2021** by **Michaela Nemcová**

# 6.3 Spustenie pivovaru

Link na repozitár, ktorý obsahuje automatický script: [startup](#)

## 6.3.1 Spustenie testovacej verzie

V tejto verzii je nahradený modul testovacím serverom, ktorý simuluje funkcionality modulov.

### Postup

0. Stiahnuť si všetky repozitáre

- ♦ Testovací modul (link na repo: [module-mock-server](#))
- ♦ Frontend (link na repo: [frontend](#))
- ♦ Backend (link na repo: [backend](#))

```
git clone https://github.com/smart-brew/module-mock-server.git
git clone https://github.com/smart-brew/frontend.git
git clone https://github.com/smart-brew/backend.git
```

1. Spustiť testovací modul

```
cd module-mock-server
docker compose build
docker compose up
```

2. Spustiť backend (spolu aj s databázou)

```
cd backend
docker compose build
docker compose up
```

#### POZNÁMKA

Ak nebola doposiaľ spustená databáza a sú problémy s nesprávnymi dátami, treba vykonať manuálne migráciu spolu so seedom údajov - viac info: [Migrácie](#)

### 3. Spustit' GUI

```
cd frontend  
docker compose build  
docker compose up
```

*Last updated on **16. 12. 2021** by **Michaela Nemcová***

# 6.4 Testovanie

Na testovanie využívame unit a integračné testy vytvorené pomocou `Jest` testovacieho rámca. Nový test vytvoríme tak, že vytvoríme `*.test.ts` súbor s rovnakým menom ako `.ts` súbor, ktorý ideme testovať. Test sa rozpoznáva na základe `.test.ts` prípony. Testy vytvárame vždy v `src/tests` priečinku.

Príkaz na manuálne spustenie testov je `yarn test`, prípadne je potrebné spustiť pred týmto príkazom ešte príkaz `yarn` ak testovanie ešte nebolo spustené.

## ⚠ PRED TESTOVANÍM

Pred testovaním je potrebné spustiť databázu v docker kontajneri pomocou príkazu `yarn docker:up` v adresári backend repozitára. Na fungovanie tohto príkazu treba spustiť aj aplikáciu Docker desktop. Po skončení testovania je možné kontajner vypnúť pomocou príkazu `yarn docker:down`.

## 6.4.1 Backend unit testy

Testy pre backend sa automaticky spúšťajú pri príkaze `git push`. Pri testovaní využívame okrem rámca `Jest` aj rámec `Supertest` ktorý slúži na odosielanie HTTP requestov na API našich `Express.js` endpointov.

Testy vytvorené záznamy v databáze po skončení vymažú.

Zoznam unit testov pre endpointy:

1. `GET /api/data`
2. `GET /api/recipe`
3. `GET /api/recipe/recipeId`
4. `POST /api/recipe/recipeId/load`
5. `PUT /api/recipe`

Konzolový výpis po úspešnom vykonaní unit testov

```
PASS src/tests/RestServer.test.ts
  Get data devices info data
    ✓ GET /api/data (33 ms)
```

```
Get all recipes
  ✓ GET /api/recipe (81 ms)
Get recipe by id
  ✓ GET /api/recipe/recipeId (55 ms)
Load recipe Smoky Grove Lichtenhainer
  ✓ POST /api/recipe/recipeId/load (33 ms)
Add new recipe
  ✓ PUT /api/recipe (150 ms)
Load, start loaded recipe and abort test
  ○ skipped PUT /api/brew/0/start
```

```
Test Suites: 1 passed, 1 total
Tests:       1 skipped, 5 passed, 6 total
Snapshots:  0 total
Time:        3.858 s, estimated 4 s
Ran all test suites.
Done in 7.09s.
```

## 6.4.2 Backend integračné testy

Integračné testy sa spúšťajú manuálne odstránením `skip()` funkcie z describe bloku testu. Bežne sa tieto testy preskakujú, pretože zatiaľ nevieme spoľahlivo ukončiť všetky procesy, ktoré sa počas testu spúšťajú, a testovanie nie je automaticky ukončené.

Zoznam integračných testov pre endpointy:

1. `PUT /api/brew/0/start` - načíta recept a spustí nové varenie

Last updated on **16. 12. 2021** by **Michaela Nemcová**

# 6.5 Tímová VM

SSH

```
ssh -i path/to/key ubuntu@team06-21.studenti.fiit.stuba.sk
```

## 6.5.1 Údaje od SSH pripojenia

Adresa

team06-21.studenti.fiit.stuba.sk

Username

ubuntu

Key (ukážka)

```
-----BEGIN RSA PRIVATE KEY-----  
MIIEpA*****  
*****  
*****8+1y9A==  
-----END RSA PRIVATE KEY-----
```

Last updated on 23. 11. 2021 by **Peter Stríž**

# 6.6 Tímový web + dokumentácia

Tímový web aj spolu s dokumentáciou sa automaticky deploy-uje na `http://team06-21.studenti.fiit.stuba.sk`, pričom dokumentácia je na `http://team06-21.studenti.fiit.stuba.sk/docs/`.

## 6.6.1 Setup deploymentu

Deployment sa robí automaticky z najnovšieho commitu na `master` vetve za pomoci `GitHub Actions`

Po commite sa vykoná krátky script, ktorý zbuilduje stránku a nahrá súbory pomocou `rsync` z GitHubovej virtuálky na náš stroj `team06-21.studenti.fiit.stuba.sk` do priečinka `~/web` a `~/web/docs`.

## 6.6.2 Server

Ako server na hostovanie webu sa používa `http-server`. Keďže všetky súbory sú statické, stačí ich len vedieť poskytnúť používateľovi.

Automatické spustenie servera:

1. Mať nainštalovaný `http-server` na VM
2. Vo VM spustiť príkaz:

```
crontab -e
```

3. Na koniec súboru pridať:

```
@reboot cd ~/web && sudo http-server -p 80
```

Last updated on **16. 12. 2021** by **Michaela Nemcová**



# **7 Analýzy**

# 7.1 Brewery

## 7.1.1 Základná charakteristika

Pozostáva z dvoch nádob, ktoré sú navzájom prepojené. Na rozdiel od zariadenia s tromi nádobami mu chýba špeciálna nádoba pre ohrev vody. Tím sa zníži nielen potreba priestoru na skladovanie pivovaru, ale aj čas k vareniu piva.

## 7.1.2 Základné postupy pri varení piva

1. Rozdrvený jačmenný slad sa zmieša s teplou (nie vriacou) vodou. Nechá sa namáčať. Možné pridať iné prísady, ako napríklad pomletú ryžu. Dôkladným zamiešaním vzniká hustá kaša, tzv. vystierka.
2. Zmes sa za stáleho miešania zahrieva pri teplote do 100 stupňov niekoľko hodín. Existujú rôzne spôsoby udržiavania teploty, ktoré sú popísané v ďalšej časti. Pri tomto procese sa škrob mení na cukry.
3. Následne je potrebné scediť tuhú časť zmesi od tekutej. Výsledkom je roztok sladkej chuti nazývaný slatina.
4. Slatina sa ďalej varí s chmeľom, ktorý do zmesi pridá horkú príchuť. Chmeľ sa necháva len vylúhovať, po tomto procese sa odstráni. Výsledkom je mladina, ktorá sa vírením zbaví kalov.
5. Po schladení sa pridávajú kvasnice. Dochádza k zmene cukrov na alkohol.

## 7.1.3 Zahrievanie

Existujú dva hlavné spôsoby ako zabezpečiť zohrievanie pri varení piva v dvoch nádobách. Sú nimi RIMS a HERMS.

### Spoločné znaky:

- ♦ na redistribuovanie sa využíva pumpa
- ♦ poskytujú zdroj tepla na zohrievanie
- ♦ neustále prihrievajú zmes mimo kade a navracajú ju tak, aby sa udržala konzistentná teplota počas daného procesu

**HERMS** Využíva ďalšiu nádobu, v ktorej sa zahrieva voda. Cez vodu vo vnútri tejto nádoby vedie cievka, cez ktorú je tlačaná zmes z kade. Tento proces prebieha v cykloch kedy sa zmes zohreje a následne navráti do kade.

**RIMS** Zdroj tepla prichádza do priameho kontaktu so zmesou. Tento proces taktiež prebieha cyklicky. Ďalším spôsobom je pôsobenie tepla priamo na spodok kade, v tom prípade je ale potrebné obsah dostatočne premiešavať aby nedošlo ku pripáleniu.

## 7.1.4 Možné kroky automatizácie pri procese varenia

- ♦ miešanie v kadi - určenie rýchlosti miešania (počet otáčok za minútu) a potrebného času (minúty)
- ♦ pohyb medzi kaďami. Nastavenie cirkulácie na istý objem (napr. cm<sup>3</sup> za minútu). Otvorenie a uzatvorenie kohútikov (open/close).
- ♦ udržanie správnej teploty - teplomer (v stupňoch celzia). Podľa toho ako je zabezpečené zahrievanie je potrebné: zvýšiť teplotu / znížiť teplotu
- ♦ pridanie chmeľu / kvasiniek. Ak automaticky tak na timer (v minútach/hodinách) podľa etapy procesu, možná kontrola či je vhodná teplota už dosiahnutá alebo podobne.

## 7.1.5 Recept:

- ♦ Vodu na varenie piva predohriať na 68 stupňov.
- ♦ Pomlieť jačmeň a namáčať ho v tejto teplote po 30 minút.
- ♦ Prefiltrovať pevnú zložku zmesi.
- ♦ Nechať prevrieť po 70 minút. Pridávať chmeľ v intervaloch. Pridať Whirlfloc tabletu 15 min do konca.
- ♦ Ochladiť na teplotu 18 stupňov, premiestniť do suda a pridať kyslík spolu s kvasnicami.
- ♦ Uzatvoriť nádobu a držať 2 týždne.

## 7.1.6 Postupy pri vykonávaní receptu, ako je zakreslenév diagrame

Vstup: voda v ohrevnom kotli, namletý jačmeň v násypníku, kvasnice v násypníku, chmeľ pripravený vo

vrecúšku, iné prísady pripravené (whirlgloc tablety, pomletá ryža, cukor,...).

- ♦ Ohrievač sa má zohriať vodu na 68 stupňov. Teplomer bude každých n sekúnd hlásiť priebežnú teplotu vody. Keď je teplota dosiahnutá, ohrievač túto teplotu musí udržiavať – t.j. ak klesne pod nejakú hranicu tak ju dohreje.
- ♦ Násypník dostane príkaz pridať jačmeň do vody. Nastaviť časovač na 30 minút, počas toho miešať zmes pri určitom počte otáčok za minútu. Po vypršaní času sa ukončí miešanie. Otvorí sa ventil na prvej nádobe, zapne sa výleva – podľa nejakého nastavenia bude pumpovať tekutinu do druhej nádoby. Zatvorí sa ventil na druhej nádobe.
- ♦ Druhá nádoba sa zohreje na požadovanú teplotu 98 stupňov (ohrievač, kontroluje teplomer). Časovač sa nastaví na 70 minút po dosiahnutí teploty varu. N minút do konca časovača sa z násypníka pridá do nádoby chmeľ na vylúhovanie. N minút do konca procesu sa pridá whirlfloc tableta. Časovač ohlási koniec 70 minút.
- ♦ Odstránenie kalov?
- ♦ Tekutina sa má zchladit' na teplotu 18 stupňov // vykonáva kryostat??. Pridajú sa kvasnice z násypníka, nechá da odležať pričom sa teplota udržiava.

Výstup: pivo pripravené na točenie, zvyšky pomletého jačmeňa v prvej nádobe, odstránený chmeľ

Koniec: voda dosiahla 68 stupňov	Ohrievač1: Zohrievaj vodu	Teplomer1: Meraj teplotu vody		
Koniec: jačmeň je pridaný do kade (	Ohrievač1: Udržuj teplotu vody 68 stupňov	Teplomer1: Meraj teplotu vody	Násypník1: pridať jačmeň do kade	Tlakový senzor meraj či ešte padá jačmeň?
Koniec: domiešané			Zapni miešač na istú rýchlosť, čas je zadaný tiež	Zapni časovač na 30 minút // kontrola, respektive upozornenie
Koniec: ventily na oboch kadiach sú otvorené	Ventil 2: otvoriť	Ventil 1: otvoriť		
Koniec: merač prúdu hlási, že je všetko preliate	Výleva: zapnúť na istý objem tekutiny	Merač prúdu: je už všetko preliate?		
Koniec: tekutina je v druhej nádobe	Výleva: vypnúť	Ventil2: zatvoriť		
Koniec: voda dosiahla 98 stupňov	Ohrievač2: Zohrievaj vodu na 98 stupňov	Teplomer2: Meraj teplotu vody	Zapni časovač na 70 minút // kontrola, respektive upozornenie	
Koniec: prešlo n minút a pridáva sa ingrediencia	Ohrievač2: Udržuj teplotu vody 98 stupňov	Teplomer2: Meraj teplotu vody	Násypník2: pridať prísadu do kade	Miešanie??
Koniec: prísada bola pridaná				Možné opakovať x krát v intervaloch n minút
Koniec: ukončenie času				
Koniec: chmeľ odstránený	Odstraň chmeľ			
Koniec: 18 stupňov	Ohrievač2: Ochlad vodu na 18 stupňov	Teplomer2: Meraj teplotu vody		
Koniec: kvasnice pridané	Ohrievač2: Udržuj teplotu	Teplomer2: Meraj teplotu vody	Násypník 3: pridať kvasnice	
Koniec: proces dokončený //čas				

## 7.1.7 Zdroje

- ♦ <https://www.homebrewhappyhour.com/rims-herms-systems-explained-compared/>
- ♦ <https://siov.sk/wp-content/uploads/2020/04/Kvasn%C3%A9-technol%C3%B3gie-v%C3%BDroba-piva.pdf>
- ♦ <https://buchvald.sk/postup-pri-vyrobe-piva/>
- ♦ [https://beerandbrewing.com/beer-recipes/?cbb\\_web\\_beer\\_recipes%5Bquery%5D=easy](https://beerandbrewing.com/beer-recipes/?cbb_web_beer_recipes%5Bquery%5D=easy) // inšpirácia k receptu

*Last updated on 23. 11. 2021 by **Peter Stríž***

# 7.2 Modbus

Modbus je otvorený protokol pre vzájomnú komunikáciu rôznych zariadení (RTU, PLC, dotykové displeje, I/O rozhrania apod), ktorý umožňuje prenášať dáta po rôznych sieťach a zberniciach. Komunikácia funguje na princípe predávania datových správ medzi klientom a serverom (**master** a **slave**).

## 7.2.1 Ako modbus funguje?

Zvyčajne býva jeden master a viacero slave zariadení (pri TCP môže byť viac mastrov). Inicializovať komunikáciu môže **iba master** a slave mu vždy len posiela odpoveď.

Protokol modbus určuje že každé zariadenie má nejakú štruktúru paketu. Býva tam:

- ♦ id ciela
- ♦ funkcia, ktorú chcem vykonať
- ♦ (optional) data
- ♦ CRC checksum

## 7.2.2 Prenosové médiá

- ♦ Ethernet cez TCP/IP
- ♦ Sériový prenos (RS-232C, RS-422, RS-485, optické vlákno, radiový prenos)
- ♦ MODBUS PLUS vysokorychlostná sieť

Preferovaný je sériový prenos cez RS-485, za režimu linky 19200 baudov, 8 dátových bitov a párna parita.

## 7.2.3 Funkcie a úložisko

Typ	Operácia	Veľkosť	Povolené adresy
Coil	Read-Write	1 bit	0x0000 – 0xFFFF

Discrete input	Read	1 bit	0x0000 – 0xFFFF
Input register	Read	16 bits	0x0000 – 0xFFFF
Holding register	Read-Write	16 bits	0x0000 – 0xFFFF

#### Modbus function codes

Function type		Function name	Function code	Comment	
Data Access	Bit access	Physical Discrete Inputs	<b>Read Discrete Inputs</b>	2	
		Internal Bits or Physical Coils	<b>Read Coils</b>	1	
			<b>Write Single Coil</b>	5	
			<b>Write Multiple Coils</b>	15	
	16-bit access	Physical Input Registers	<b>Read Input Registers</b>	4	
		Internal Registers or Physical Output Registers	<b>Read Multiple Holding Registers</b>	3	
			<b>Write Single Holding Register</b>	6	
			<b>Write Multiple Holding Registers</b>	16	
			Read/Write Multiple Registers	23	
			Mask Write Register	22	
			Read FIFO Queue	24	
	File Record Access	Read File Record	20		
		Write File Record	21		
	Diagnostics	Read Exception Status	7	serial only	
Diagnostic		8	serial only		
Get Com Event Counter		11	serial only		
Get Com Event Log		12	serial only		
Report Server ID		17	serial only		
Read Device Identification		43			
Other	Encapsulated Interface Transport	43			

## Zdroje

- ♦ <https://modbus.org>
- ♦ <https://en.wikipedia.org/wiki/Modbus>
- ♦ <https://www.youtube.com/watch?v=JBGaInI-TG4>
- ♦ <https://www.npmjs.com/package/modbus-serial>



- ♦ Meniče H-300 <https://docs.google.com/document/d/1RQNIFq4vbOvbZFfxEuz4CkOA-SDlksPzpciTiBFUaeA/edit?usp=sharing>
- ♦ <https://www.youtube.com/watch?v=i46jdhvRej4>

# 7.3 SCADA

**SCADA** je skratka pre **S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition.

## 7.3.1 Základná charakteristika

Ide o koncept kontrolného systému, kde je potreba monitorovať, kontrolovať, získavať a analyzovať real-time dáta z priemyselných zariadení a procesov.

Any application that gets data about a system in order to control that system is a SCADA application.

## 7.3.2 Kľúčové komponenty

SCADA systém sa najčastejšie skladá z komponentov:

- ♦ Supervisory computer
- ♦ Remote terminal units
- ♦ Programmable logic controllers
- ♦ Communication infrastructure
- ♦ Human-machine interface

### Supervisory computer

- ♦ **jadro SCADA systému** - zbiera dáta z procesu a posiela príkazy do pripojených zariadení
- ♦ zahŕňa počítač a softvér komunikujúci s pripojenými zariadeniami + HMI softvér bežiaci na pracovných staniciach operátorov
- ♦ pri veľkých systémoch môže ísť o viacero serverov, viacero HMI softvérov na klientských počítačoch (proste massive)

### Remote terminal/telemetry units (RTUs)

- ♦ prepája fyzické zariadenia (senzory, efektory) na hlavný kontrolný systém (SCADA systém)
- ♦

prenáša telemetrické dáta na hlavný systém

- ♦ pomocou správ z hlavného systému kontroluje pripojené fyzické zariadenia

## **Programmable logic controllers (PLCs)**

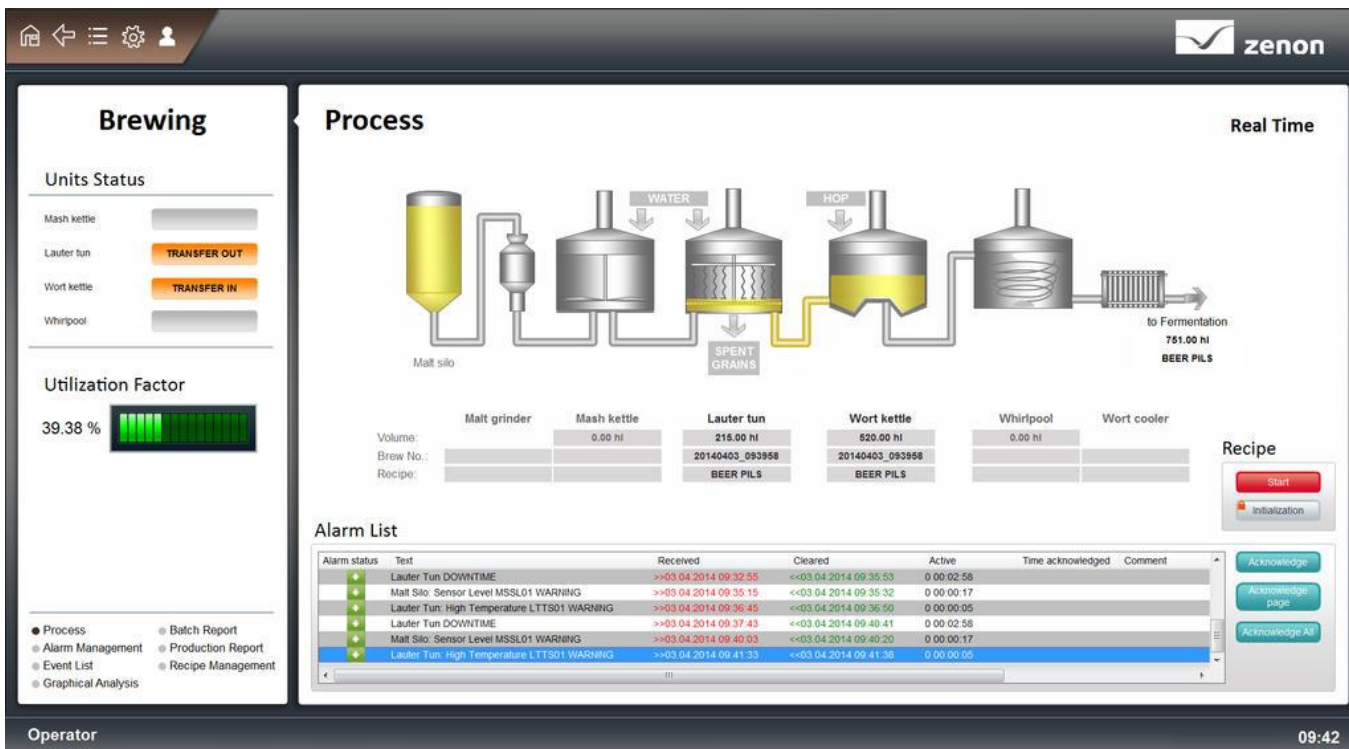
- ♦ počítač používaný na automatizáciu elektromechanických procesov
- ♦ môže mať veľa inputov a outputov
- ♦ má naprogramovanú logiku, ktorá spracováva dáta z pripojených senzorov a riadi pripojené efekty

## **Communication infrastructure**

- ♦ prepája hlavný kontrolný systém s RTU-čkami a PLC-čkami
- ♦ využíva komunikačné protokoly podľa industry štandardov

## **Human-machine interface (HMI)**

- ♦ využívané pre vizualizáciu a kontrolu procesu
- ♦ má nejakú obrazovku (napr. monitor), cez ktorú môžu používatelia interagovať priamo so strojmi alebo kontrolovať zariadenia
- ♦ prezentuje dáta získané zo zariadení (PLC / RTU)
- ♦ obrazovka zobrazuje grafickú reprezentáciu vybavenia / procesov, ktoré je treba monitorovať
  - mimic diagram
- ♦ grafická reprezentácia môže byť vytvorená cez špeciálny softvér
  - napr. SIMATIC WinCC



## 7.3.3 Využitie

- ♦ elektrárne
- ♦ vodárne, čističky vôd
- ♦ inteligentné budovy, továrne
- ♦ ...

## 7.3.4 Zdroje

- ♦ <https://en.wikipedia.org/wiki/SCADA>
- ♦ <https://steemit.com/steemstem/@ibk-gabriel/scada-systems-principle-and-applications>
- ♦ <https://www.copadata.com/en/industries/food-beverage/food-beverage-insights/creative-brewing/>
- ♦ <https://www.automationworld.com/products/control/blog/13307242/rtu-or-plc-which-is-right-for-you>

Last updated on **23. 11. 2021** by **Peter Stríž**

# **8 Architektúra**

# 8.1 API

Rôzne komponenty spolu komunikujú rôznym spôsobom, preto je dôležité mať jedno miesto, kde sa dá prehľadne pozrieť, ako fungujú naše API.

## 8.1.1 Module <--> Backend

Modul komunikuje smerom na backend cez WebSocket a posiela periodicky nasledujúce správy:

### Posielanie údajov z modulu na backend

Posielanie údajov z modulu na backend

```
{
  "moduleId": "<názov modulu>",
  "status": "<ok | progress | error>",
  <podporované údaje>
}
```

- ♦ **<podporované údaje>** - sú rovnaké údaje ako v [Podporované údaje](#)

### Posielanie inštrukcií z backendu na modul

Posielanie inštrukcií z backendu na modul

```
{
  "type": "instruction",
  "moduleId": "<názov modulu>",
  "category": "<ketegória>", // "MOTOR", "PUMP", ...
  "device": "<názov zariadenia>", // "MOTOR_1", "PUMP_2", ...
  "instruction": "<inštrukcia>", // "SET_TEMPERATURE", "SET_MOTOR_SPEED", ...
  "param": "<parameter>"
}
```

## 8.1.2 Backend <--> Frontend

Pri každom dopyte a jeho odpovedi, sa budú nachádzať aj `created_at` a `updated_at`, avšak pre prehľadnosť, tu nie sú napísané.

### Podporované funkcie

FE môže získať informácie o podporovaných funkciách v rámci systému.

GET /api/function

```
[
  {
    "id": 1,
    "codeName": "SET_TEMPERATURE",
    "name": "Set temperature",
    "category": "TEMPERATURE",
    "units": "°C",
    "inputType": "float",
    "description": "Sets temerature for selected chamber",
    "options": [
      {
        "id": 1,
        "name": "Chamber 1",
        "codeName": "TEMP_1"
      },
      {
        "id": 2,
        "name": "Chamber 2",
        "codeName": "TEMP_2"
      }
    ]
  },
  {
    "id": 2,
    "codeName": "SET_MOTOR_SPEED",
    "name": "Set motor speed",
    "category": "MOTOR",
    "units": "RMP",
    "inputType": "float",
    "description": "Sets rpms for selected motor",
    "options": [
      {
        "id": 3,
        "name": "Motor 1",
        "codeName": "MOTOR_1"
      },
      {
        "id": 4,
```



```
    "name": "Motor 2",
    "codeName": "MOTOR_2"
  }
]
},
{
  "id": 3,
  "codeName": "TRANSFER_LIQUIDS",
  "name": "Transfer liquids",
  "category": "PUMP",
  "units": null,
  "inputType": null,
  "description": "Transfers liquids from first chamber to second",
  "options": [
    {
      "id": 5,
      "name": "Pump 1",
      "codeName": "PUMP_1"
    }
  ]
},
{
  "id": 4,
  "codeName": "UNLOAD",
  "name": "Unload ingredient",
  "category": "UNLOADER",
  "units": null,
  "inputType": null,
  "description": "Unloads selected ingredient into chamber",
  "options": [
    {
      "id": 6,
      "name": "Fermentables",
      "codeName": "FERMENTABLE"
    },
    {
      "id": 7,
      "name": "Yeast",
      "codeName": "YEAST"
    }
  ]
}
```

```

    },
    {
      "id": 8,
      "name": "Hops",
      "codeName": "HOPS"
    },
    {
      "id": 9,
      "name": "Other",
      "codeName": "OTHER"
    }
  ]
},
{
  "id": 5,
  "codeName": "WAIT",
  "name": "Wait",
  "category": "SYSTEM",
  "units": "Minutes",
  "inputType": "float",
  "description": "System will wait for given amount of minues",
  "options": []
},
{
  "id": 6,
  "codeName": "MANUAL",
  "name": "Manual step",
  "category": "SYSTEM",
  "units": null,
  "inputType": "string",
  "description": "System will wait for manual inervention",
  "options": []
}
]

```

## Výber všetkých receptov

Pri zapnutí si používateľ bude môcť vybrať, či chce použiť už vytvorený recept, alebo či chce vytvoriť nový recept.

Pokiaľ si vyberie použitie vytvoreného receptu, front-end pošle GET request na back-end URL:

```
GET /api/recipe
```

Ako odpoveď obdrží základné informácie (id, názov, čas vytvorenia) pre **všetky** recepty v JSON formáte:

```
[
  {
    "id": 3,
    "name": "Smoky Grove Lichtenhainer",
    "description": "Light, gently tart, and smoked-lichtenhainer is an unusual beer, yet surprisingly good for all seasons and one you'll want to brew and enjoy often.",
    "locked": false
  },
  {
    "id": 4,
    "name": "Burke-Gilman The Hopsplainer",
    "description": "Courtesy of the brewing team at Burke-Gilman in Seattle, here is a homebrew-scale recipe for the double hazy IPA that won GABF gold in 2020.",
    "locked": true
  },
  ...
]
```

## Výber jedného receptu

FE ponúkne používateľovi výber z prijatých receptov podľa názvu. Pri vybratí receptu sa odošle GET request na backend:

```
GET /api/recipe/{recipe-id}
```

Odpoveďou bude JSON so všetkými dátami receptu:

```
{
  "id": 3,
  "name": "Smoky Grove Lichtenhainer",
  "description": "Light, gently tart, and smoked-lichtenhainer is an unusual beer, yet surprisingly good for all seasons and one you'll want to brew and enjoy often.",
  "locked": false,
  "Ingredients": [
    {
      "id": 5,
      "recipeId": 3,
      "name": "American - Pale 2-Row",
```

```

    "amount": 5.6,
    "type": "Fermentable",
    "units": "Kg"
  },
  {
    "id": 6,
    "recipeId": 3,
    "name": "Fermentis - Safale - American Ale Yeast US-05",
    "amount": 1,
    "type": "Yeast",
    "units": ""
  },
  {
    "id": 7,
    "recipeId": 3,
    "name": "Magnum (Pellet)",
    "amount": 1,
    "type": "Hops",
    "units": "oz"
  },
  {
    "id": 8,
    "recipeId": 3,
    "name": "Crush whilrfoc Tablet",
    "amount": 1,
    "type": "Other",
    "units": ""
  }
],
"Instructions": [
  {
    "id": 11,
    "recipeId": 3,
    "templateId": 2,
    "codeName": "SET_MOTOR_SPEED",
    "param": 30,
    "category": "MOTOR",
    "optionCodeName": "MOTOR_1",
    "blockId": 1,
    "blockName": "Fermentation",
    "ordering": 1
  },
  {

```

```
"id": 12,  
"recipeId": 3,  
"templateId": 1,  
"codeName": "SET_TEMPERATURE",  
"param": 85,  
"category": "TEMPERATURE",  
"optionCodeName": "TEMP_1",  
"blockId": 1,  
"blockName": "Fermentation",  
"ordering": 2  
},  
{  
"id": 13,  
"recipeId": 3,  
"templateId": 2,  
"codeName": "SET_MOTOR_SPEED",  
"param": 0,  
"category": "MOTOR",  
"optionCodeName": "MOTOR_1",  
"blockId": 2,  
"blockName": "Yeastng",  
"ordering": 3  
},  
{  
"id": 14,  
"recipeId": 3,  
"templateId": 1,  
"codeName": "SET_TEMPERATURE",  
"param": 23,  
"category": "TEMPERATURE",  
"optionCodeName": "TEMP_1",  
"blockId": 2,  
"blockName": "Yeastng",  
"ordering": 4  
}  
]  
}
```

## Výber receptu na varenie

Pokiaľ je používateľ s vybraným receptom spokojný, klikne na tlačidlo **"Vybrať recept"**. Recept sa následne načíta na hlavnú obrazovku a tiež sa odošle POST request na BE s vybraným receptom (aby BE vedel, že čo sa bude robiť):

```
POST /api/recipe/{recipe-id}/load
```

## Pridanie nového receptu

Pokiaľ používateľ chce vytvoriť nový recept, pomocou obrazovky pre pridanie receptu vykliká všetky požadované kroky a klikne na tlačidlo **"Pridať recept"**. Na BE sa odošle PUT request:

```
PUT /api/recipe
```

```
{
  "name": "Perfect Northeast IPA (NEIPA)",
  "description": "I have no idea what I'am doing",
  "locked": false,
  "Ingredients": [
    {
      "name": "American - Pale 2-Row",
      "amount": 5.6,
      "type": "Fermentable",
      "units": "Kg"
    },
    {
      "name": "Fermentis - Safale - American Ale Yeast US-05",
      "amount": 1,
      "type": "Yeast",
      "units": ""
    },
    {
      "name": "Magnum (Pellet)",
      "amount": 1,
      "type": "Hops",
      "units": "oz"
    },
    {
      "name": "Crush whilrfoc Tablet",
      "amount": 1,
      "type": "Other",
      "units": ""
    }
  ],
  "Instructions": [
    {
      "templateId": 4,

```

```

    "param": null,
    "optionCodeName": "FERMENTABLE",
    "blockName": "Fermentation",
    "ordering": 4
  },
  {
    "templateId": 1,
    "param": "60",
    "optionCodeName": "TEMP_1",
    "blockName": "Fermentation",
    "ordering": 3
  },
  {
    "templateId": 5,
    "param": "5",
    "optionCodeName": null,
    "blockName": "Yeasting",
    "ordering": 2
  },
  {
    "templateId": 2,
    "param": "100",
    "optionCodeName": "MOTOR_1",
    "blockName": "Yeasting",
    "ordering": 1
  }
]
}

```

Ako odpoveď na FE príde JSON s vygenerovaným ID pridaného receptu:

```

200 OK
{
  "id" : xxx
}

```

Po vytvorení receptu sa vrátíme na výber receptov, kde bude už nový recept zobrazený.



## Vymazanie receptu

Pokiaľ chce používateľ vymazať jeden z receptov, klikne na ten, ktorý chce vymazať a následne klikne na tlačidlo **"Vymazať"**. Po kliknutí na tlačidlo sa zobrazí potvrdzovacie okno a po kliknutí tlačidla **"Potvrdiť"** sa pošle na BE POST request:

```
POST /api/recipe/{recipe-id}/delete
```

Ako odpoveď na FE príde JSON s ID vymazaného receptu:

```
200 OK
{
  "id" : 3
}
```

Po vymazaní receptu sa obrazovka s receptami načíta znova a vymazaný recept už nebude zobrazovaný.

## Začiatok varenia

Po vybratí receptu sa recept načíta na hlavnú obrazovku. Používateľ dostane príležitosť skontrolovať, či je recept správny. Pokiaľ je s receptom spokojný, spustí varenie. Pri spustení varenia sa na BE pošle PUT request:

```
PUT /api/brew/0/start
```

```
{
  "recipeId": 123 // id vybraného receptu
}
```

FE čaká na odpoveď z BE, či sa všetko úspešne spustilo:

```
200 OK
{
  "brewId" : xxx
}
```

FE si uloží ID varenia pre ďalšie dopyty.

Pokiaľ na BE nastane porucha alebo chyba pri spúšťaní, na FE pošle odpoveď:

```
500 SERVER ERROR
{
  "error" : "Temp error message."
}
```

FE túto chybu následne ohlásí používateľovi.

## Periodické dopyty na back-end

Pokiaľ sa varenie spustí úspešne, FE prejde do módu, kde sa periodicky dopytuje BE na stav receptu. Každú 1 sekundu na BE odošle GET request na URL:

```
GET /api/data
```

### Pokiaľ všetko prebieha v poriadku

V ideálnom prípade BE odpovie formou:

```
200 OK
{
  "data": {
    <podporované údaje>
  },
  "instruction" : {
    "currentInstruction": 23,
    "currentValue": 36.5,
    "status": "IN_PROGRESS",
  },
  "brewStatus": "IN_PROGRESS"
}
```

- ♦ **<podporované údaje>** - sú rovnaké údaje ako v Podporované údaje

FE túto odpoveď spracuje a obnoví obrazovku.

## Pokiaľ nastane chyba

Pokiaľ došlo k chybe niekde v pipeline, BE odpovie formou:

```
500 SERVER ERROR
{
  "error": "Temp error message.",
  "module": {
    "moduleId": 234,
    "device": "MOTOR_1",
    "category": "MOTOR",
    "error": "Cannot communicate with device"
  },
  "instruction" : {
    "currentInstruction": 23,
    "currentValue": 36.5,
    "status": "FAIL"
  },
  "brewStatus": "STOPPED"
}
```

## Úspešné ukončenie varenia

Pokiaľ BE úspešne ukončil varenie, pri najbližšom GET dopyte (`GET /api/data`) BE pridá položku `status: "FINISHED"`:

```
200 OK
{
  ...
  ...
  "brewStatus": "FINISHED"
}
```

FE o tejto skutočnosti upovedomí používateľa a ukončí mód periodických dopytov.

## Potvrdenie manuálnej inštrukcie

Keď nastane situácia, že treba vykonať manuálnu inštrukciu, systém bude čakať až kým používateľ nepotvrdí jej vykonanie.

```
POST /api/brew/{brewId}/instruction/{instructionId}/done
```

```
200 OK
```

Až po nasledujúcom dopyte v Periodické dopyty na back-end sa prejde na ďalšiu inštrukciu, keď nám BE pošle, že sme na nasledujúcej inštrukcii.

## Úprava parametrov počas varenia

Pokiaľ nastane zmena parametrov nejakých krokov, ktoré ešte neboli vykonané, FE odošle POST request na BE:

```
POST /api/brew/{brewId}/instruction/{instructionId}
```

```
{  
  // celá inštrukcia s upravenými parametrami, vid' štruktúra JSONu inštrukcie"  
}
```

Pokiaľ úprava prebehne v poriadku, BE odpovie správou:

```
200 OK
```

Pokiaľ úpravu nebolo možné vykonať, BE odpovie správou:

```
400 BAD REQUEST  
{  
  "error" : "Temp error message."  
}
```

Note: je potrebné, aby FE po tomto POST requeste spustil timer na periodické dopyty odznova, aby sme sa vyhli neprijemnostiam s asynchronitou BE.

## Pauza počas varenia

Na front-ende by mala byť možnosť pauznúť varenie používateľom. Používateľ klikne na tlačidlo "**Pauza**". FE sa opýta, či si je používateľ istý.

Po potvrdení je na BE odoslaný POST request:

```
POST /api/brew/{brewId}/pause
```

BE by mal zastaviť časovače na všetkých aktívnych procesoch a odpovedať formou:

```
200 OK
```

Pokiaľ pri zrušení nastane chyba, BE odpovie formou:

```
500 SERVER ERROR
{
  "error" : "Temp error message."
}
```

## Pokračovanie varenia

Ak chceme pokračovať vo varení, pošleme:

```
POST /api/brew/{brewId}/resume
```

## Zrušenie varenia

Na front-ende by mala byť možnosť prerušiť varenie používateľom. Používateľ klikne na tlačidlo "**Zrušiť varenie**". FE sa opýta, či si je používateľ istý.

Po potvrzení je na BE odoslaný POST request:

```
POST /api/beer/{beerId}/abort
```

BE by mal zrušit všetky procesy, správne vypnúť všetky zariadenia a odpovedať formou:

```
200 OK
```

Pokiaľ pri zrušení nastane chyba, BE odpovie formou:

```
500 SERVER ERROR
{
  "error" : "Temp error message."
}
```

Now it's time to panic.

## Vypnutie systému

Používateľ môže kliknúť na tlačidlo vypnutia systému. FE sa opýta, či si je používateľ istý.

Po potvrzení je na BE odoslaný POST request:

```
POST /api/shutdown
```

BE následne vypne celé RPI, na ktorom beží systém pivovaru.

*Last updated on 15. 12. 2021 by **Peter Stríž***

# 8.2 Back-end

Link na repozitár - [Backend](#)

## 8.2.1 Technológie

Server využíva technológiu **Node JS**.

Pre spojenie s databázou a obsluhu požiadaviek na databázu využívame ORM **Prisma**.

## 8.2.2 Komunikácia

Pre komunikáciu s frontendom je k dispozícii REST API, ktoré je dostupné na porte 8000.

Pre komunikáciu s jednotlivými modulmi pivovaru používame **websocket**. Spojenie prebieha na porte 8001.

Viac info v časti [API](#).

## 8.2.3 Súčasný stav

### Web socket

Pomocou web socketového servera je možné pripojiť viacero klientov, ktorí predstavujú moduly pivovaru. Tie slúžia na obsluhu efektorov a zisk údajov zo sensorov. Server s nimi pravidelne udržiava spojenie (keep-alive) a odosiela im jednotlivé inštrukcie. Bližšie informácie o nich je možné nájsť v [Podporované inštrukcie](#).

Pokiaľ spojenie k požadovanému modulu neexistuje, proces varenia sa ukončí špeciálnou inštrukciou **Abort**.

### REST API

Zoznam endpointov v rámci REST API sa nachádza v [API](#), avšak tento zoznam sa môže podľa potrieb aktualizovať. Momentálne pre obsluhu požiadaviek frontendu implementujeme nasledovné endpointy:

- ◆ Získanie všetkých receptov
- ◆ Získanie konkrétneho receptu
- ◆ Získanie všetkých podporovaných funkcií
- ◆ Vytvorenie nového receptu
- ◆ Načítanie receptu pre varenie
- ◆ Spustenie varenia
- ◆ Pozastavenie a pokračovanie varenia
- ◆ Vynútené zrušenie varenia

## 8.2.4 Do budúca

V nadchádzajúcich šprintoch budeme musieť doplniť a implementovať endpointy pre nasledovné požiadavky:

- ◆ Vymazanie receptu
- ◆ Úprava receptu
- ◆ Úprava inštrukcie počas varenia
- ◆ História varení
- ◆ Prehľad štatistík varenia

*Last updated on 16. 12. 2021 by **Michaela Nemcová***



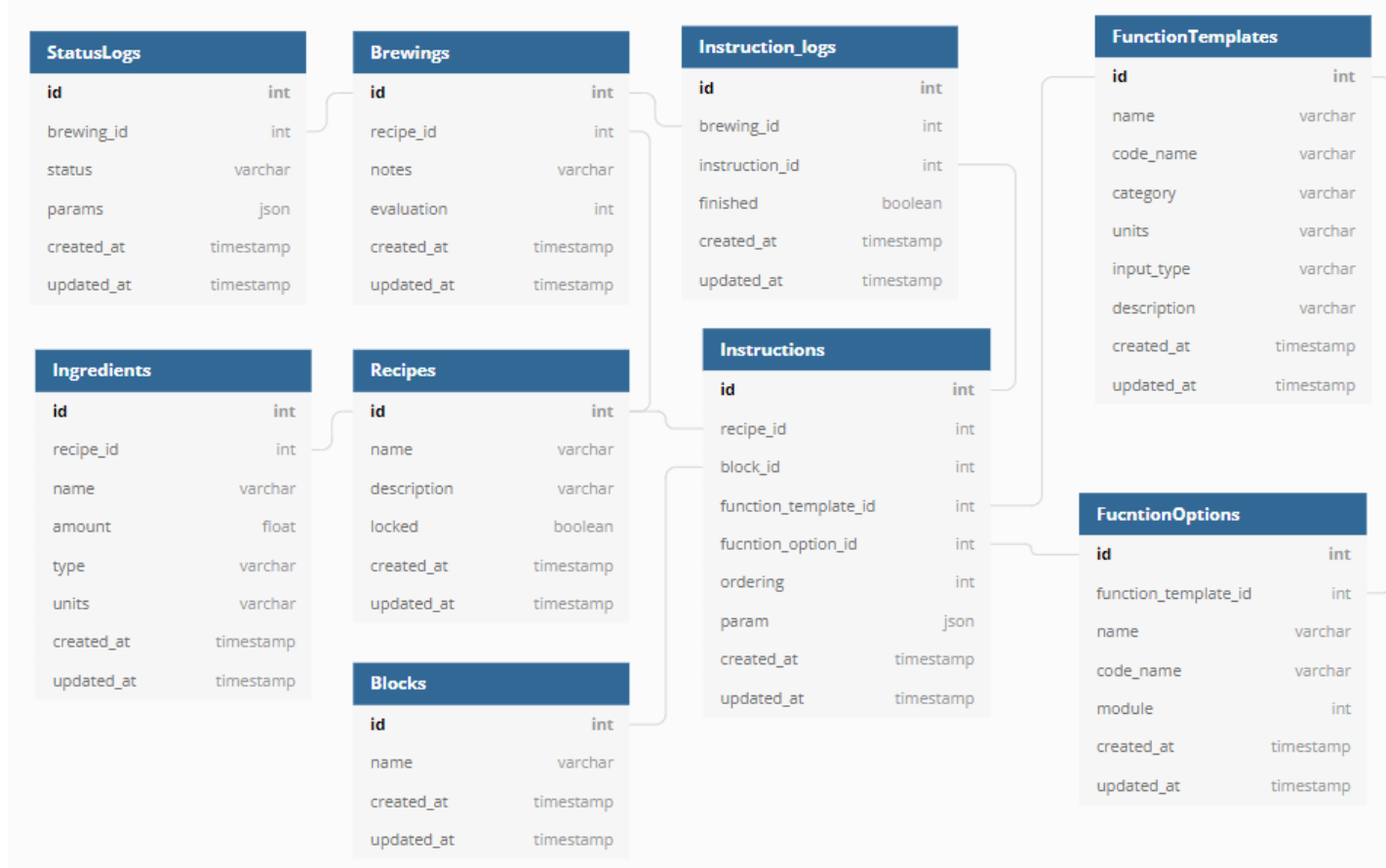
## 8.3 Databáza

Na databázu používame technológiu `PostgreSQL` a na manipuláciu dát sa využíva ORM `Prisma`.

### 8.3.1 Návrh databázy

Hlavnou entitou je **Recipe**, ktorá reprezentuje recept pre varenie piva. S ňou sú spojené ďalšie entity, ako **Ingredient** a **Instruction**, ktoré reprezentujú ingrediencie použité v recepte a konkrétne inštrukcie, ktoré sa v recepte majú vykonať. Entita **Block** slúži pre frontend na zoskupenie inštrukcií do väčších celkov. Entity **Function Template** a **Function Option** predstavujú podporované funkcie v rámci tohto systému a k nim prislúchajúce možnosti, z ktorých má používateľ možnosť výberu. Entita **Brewing** reprezentuje konkrétne varenie podľa vybraného receptu. Entity **Status Log** a **Instruction Log** predstavujú pravidelné záznamy z varenia a vykonávania jednotlivých inštrukcií.

# Návrh databázy po prvom semestri



# 8.4 Front-end

Link na repozitár - [Frontend](#)

## 8.4.1 Technológie

Pre používateľské rozhranie budeme používať **React Native framework**.

Pre IDE je treba doinštalovať tieto rozšírenia - **eslint**, **prettier**, **editorconfig** (odporúčaný je VSCode).

## 8.4.2 Komunikácia s back-endom

Pre komunikáciu využijeme REST API cally a správy vo formáte JSON. Viac info na [API](#).

## 8.4.3 Opis prostredia

Návrh užívateľského prostredia prešiel viacerými kolami úprav. Dole je opísaná verzia aktuálna po dokončení 5. šprintu, spolu s návrhmi úprav do ďalších šprintov.

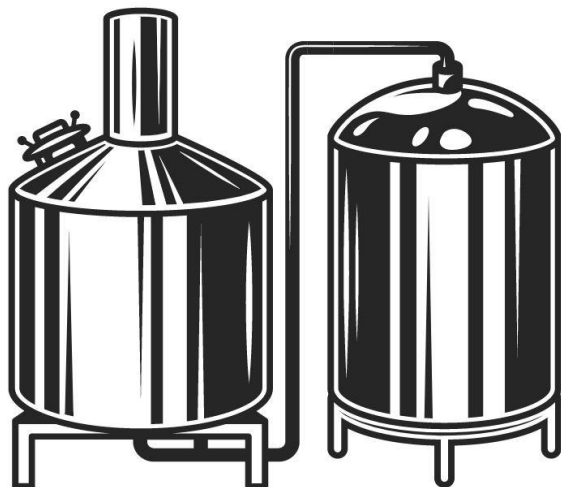
# Hlavná obrazovka



Overview

Recipes

History



Temperature  
**50 °C**  
Motor speed  
**1000000000 RMD**  
Heating

Temperature  
**21.5 °C**  
Motor speed  
**1000000000 RMD**  
Heating

Start brewing

## Smoky Grove Lichtenhainer

### Fermentable

5.6 Kg American - Pale 2-Row

### Yeast

1 Fermentis - Safale -  
American Ale Yeast US-05

### Hops

1 oz Magnum (Pellet)

### Other

1 Crush whirlfoc Tablet

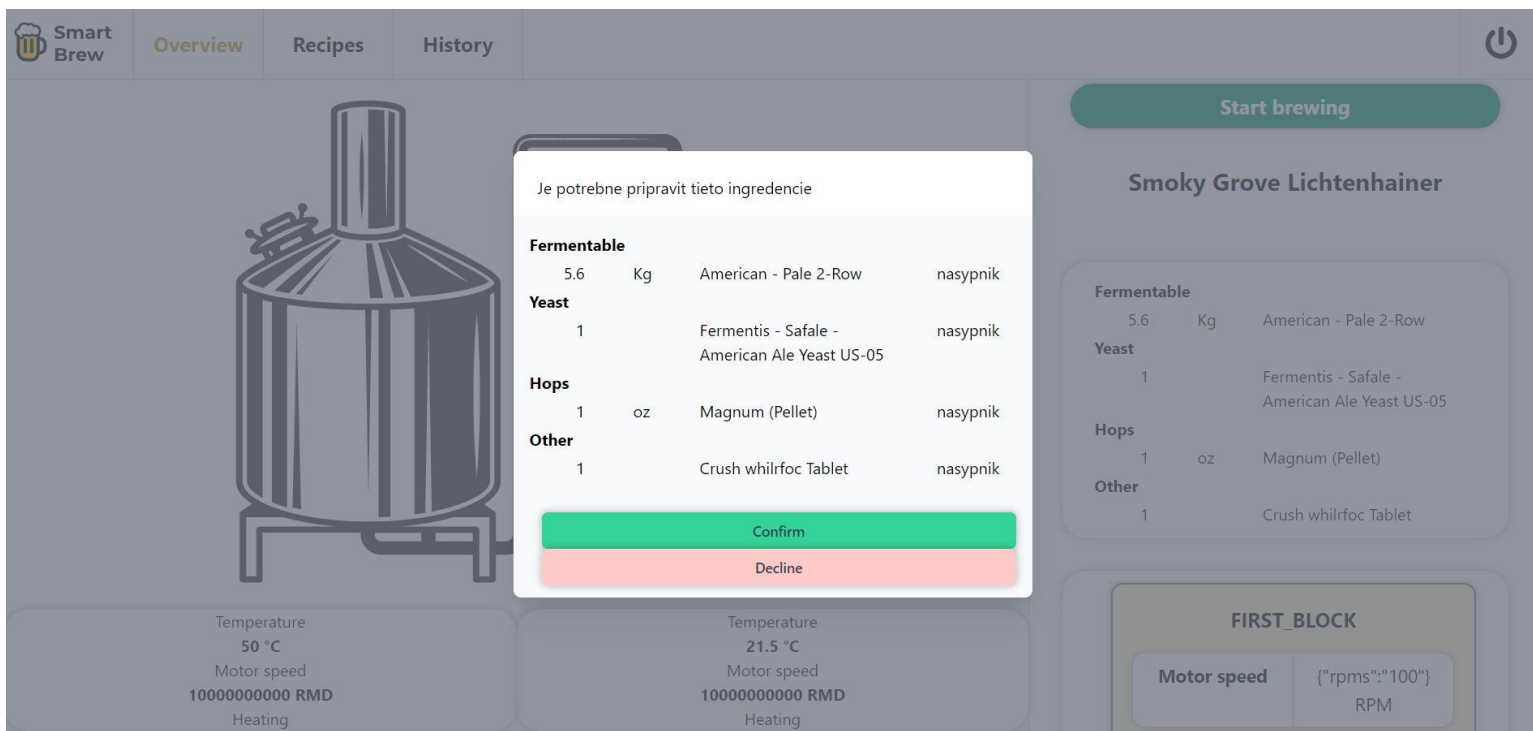
### FIRST\_BLOCK

Motor speed 100 RPM

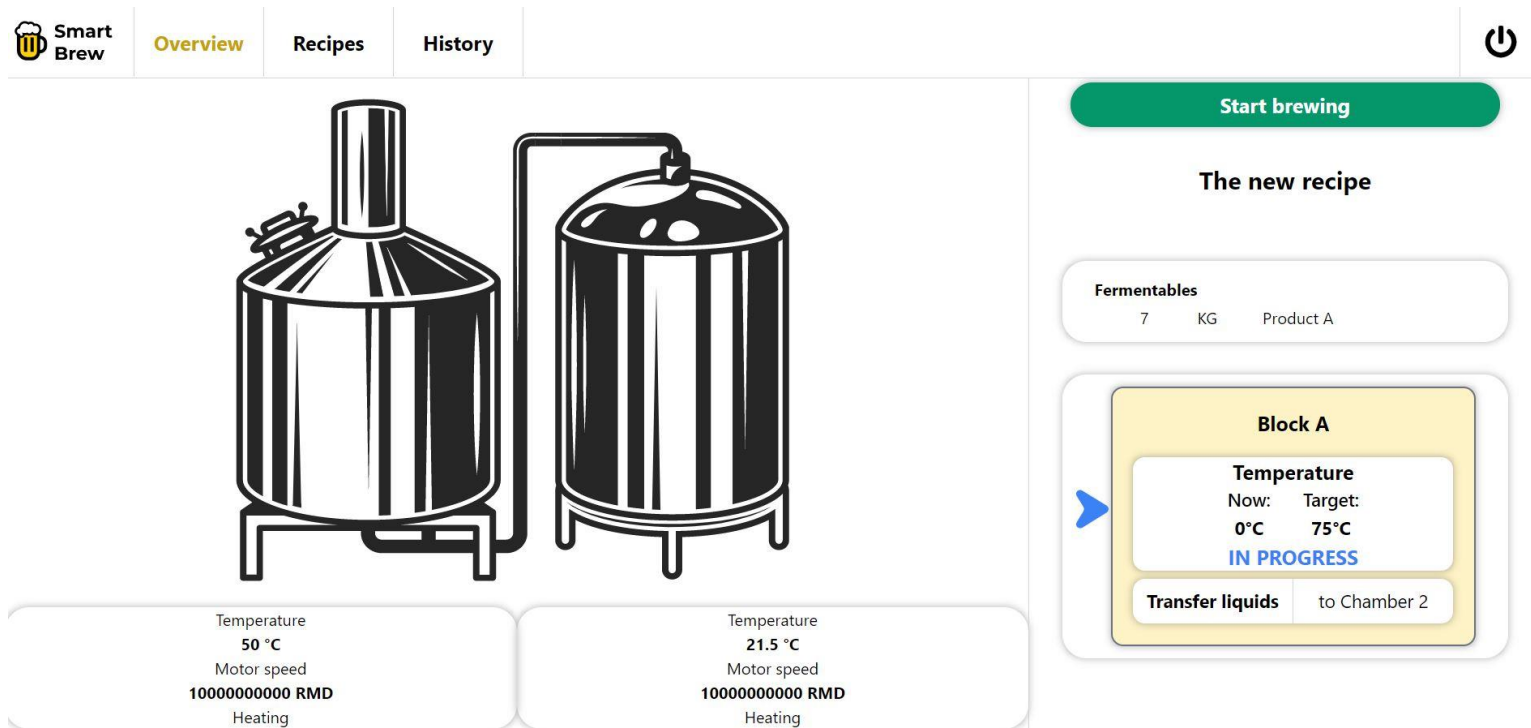
Wait 00:00:00

Obrázok zobrazuje stav, kedy je navolený recept, ale nie je potvrdené začatie procesu. Na ľavej strane obrazovky sa nachádza zariadenie, spolu s informáciami o procesoch vykonávaných sa v jednotlivých nádobách. Napravo sú zobrazené potrebné ingrediencie, spolu s inštrukciami, ktoré sa budú vykonávať. (Používateľ si môže skontrolovať, či má všetky dostupné ingrediencie, a v prípade potreby vybrať iný recept).

Plánom je na ľavú stranu obrazovky pridať presný náčrt celého zariadenia. Pri jeho jednotlivých častiach by sa následne zobrazoval ich stav, napr. aktuálna teplota v nádobe.



Pred začatím varenia treba potvrdiť, že sa potrebné ingrediencie nachádzajú v správnom násypníku.



Po začatí varenia sa používateľovi zobrazujú bližšie informácie o prebiehajúcich procesoch. V budúcnosti je možné pridanie ďalšieho obsahu na obrazovku, napríklad časovača na zobrazenie, kedy proces skončí.

# Výber receptu



Overview

Recipes

History



## Smoky Grove Lichtenhainer

### Fermentable

5.6 Kg American - Pale 2-Row

### Yeast

1 Fermentis - Safale - American Ale Yeast US-05

### Hops

1 oz Magnum (Pellet)

### Other

1 Crush whirlfoc Tablet

### FIRST\_BLOCK

Motor speed 100 RPM

Wait 00:00:00

## Recipes

Smoky Grove Lichtenhainer

Vanilla Cream Ale

Load recipe

Create new

Edit

Delete

Pokiaľ si používateľ chce zvoliť recept na varenie, kliknutím na možnosť **Recipe** v menu prejde na obrazovku výberu receptu. Na pravej strane obrazovky sa zobrazí zoznam všetkých dostupných receptov. Po kliknutí na recept sa zobrazia informácie k danému receptu. Používateľ tento recept môže použiť tak, aký je a načítať ho na hlavnú obrazovku, alebo ho upraviť. Tiež si môže vytvoriť úplne nový recept. Je taktiež možné recept vymazať.

## Pridanie ingrediencií



Overview

Recipes

History



Name of the recipe

### Ingredients

#### Fermentables

3 KG Product A - [+ Add ingredient](#)

#### Yeast

0,5 KG Product B - [+ Add ingredient](#)

0,4 KG Product C - [+ Add ingredient](#)

#### Hops

[+ Add ingredient](#)

#### Other

40 l Product D - [+ Add ingredient](#)

Next step

Cancel

Pokiaľ si používateľ vyberie možnosť upraviť/vytvoriť recept, dostane sa na obrazovku pre vytvorenie zoznamu potrebných ingrediencií. Na hornej časti stránky sa nachádza pole na pridanie názvu receptu. Formulár na pridanie ingrediencií obsahuje delenie na jednotlivé kategórie ingrediencií, a možnosť pridať a odobrať ingredienciu. Vždy je potrebné nastaviť množstvo, jednotku a popis ingrediencie (jej názov). Po stlačení tlačidla **Next step** používateľ prejde na úpravu/zostavovanie inštrukcií v recepte. Pri **Cancel** sa najprv zobrazí upozornenie, že sa dané údaje stratia, pokiaľ bude používateľ pokračovať. Pri pokračovaní sa vracia na stránku s výberom receptu.

## Pridanie inštrukcií

The screenshot displays the 'Smart Brew' application interface. At the top, there are navigation tabs: 'Overview', 'Recipes' (highlighted), and 'History'. A power button is visible in the top right corner. The main content area is divided into two columns. The left column contains a recipe editor with two process blocks: 'Process A' and 'Process B'. 'Process A' includes a 'Temperature' step (set to 70°C) and a 'Wait' step (0 days, 0 hours, 15 minutes). Each step has a red 'X' icon for deletion and a green '+' icon for addition. The right column shows a 'Recipe name' field and a list of ingredients categorized into 'Fermentables', 'Yeast', and 'Other'. At the bottom right, there are three buttons: 'Save recipe', 'Previous step', and 'Cancel'.

Recipe name: Name of the recipe		
<b>Fermentables</b>		
3	KG	Product A
<b>Yeast</b>		
0.5	kg	Product B
0.4	KG	Product C
<b>Other</b>		
40	litre	Product D

Na obrazovke úpravy/zostavovania inštrukcií v recepte sa používateľovi vpravo zobrazuje zoznam ingrediencií, ktoré si v predošlom kroku navolil. Inštrukcie sú zobrazené naľavo. Delené sú do osobitných blokov označených vlastným názvom. Je možné pridávať a odoberať bloky, a rovnako aj inštrukcie v rámci blokov. Každá inštrukcia obsahuje pomocný popisok, ktorý jednoduchým spôsobom objasňuje používateľovi, čo daná inštrukcia vykonáva. Po kliknutí na zelené plusko sa zobrazí obrazovka výberu inštrukcie, je zobrazená nižšie.

**Select instruction:****Temperature**

*Sets temperature for selected chamber*

**Motor**

*Sets rpms for selected motor*

**Transfer liquids**

*Transfers liquids from first chamber to second*

**Unload**

*Unloads selected ingredient into chamber*

**Wait**

*System will wait for given amount of minutes*

**Manual step**

*System will wait for manual intervention*

Cancel

## História varenia

Túto obrazovku je potrebné zostaviť.

## 8.4.4 Do budúcna

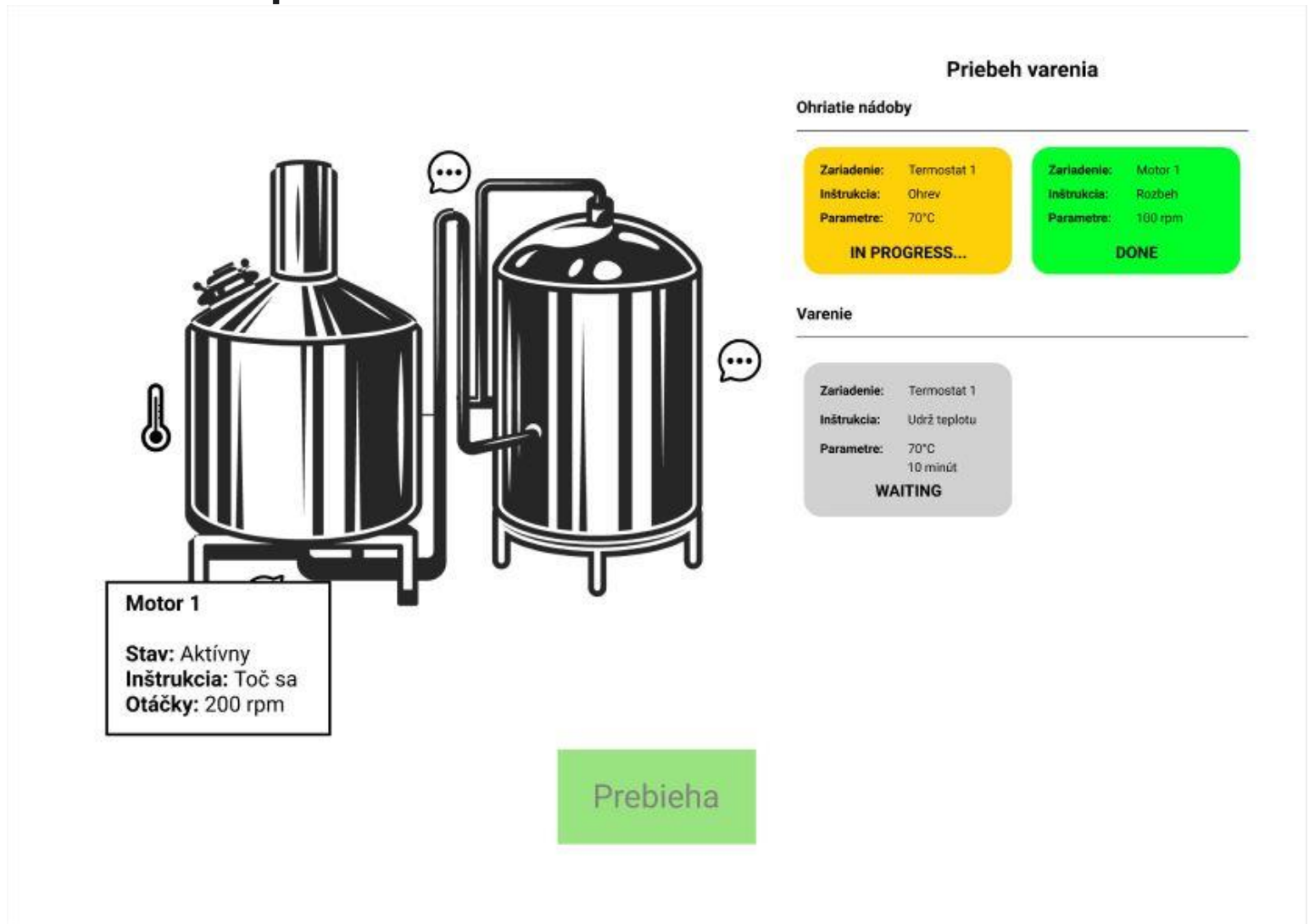
### Je potrebné

- označovať jednotlivé násypníky (Akú úlohu môžu zastávať - jačmeň sa bude pridávať iba jedným z nich, a podobne.)
- definovať, odkiaľ sa bude načítavať stav zariadenia (backend/konfiguračný file). Napríklad ktoré násypníky sú dostupné a môžu sa využiť pri varení jednotlivých receptov
- šipka ukazujúca či teplota v jednotlivých nádobách stúpa alebo klesá



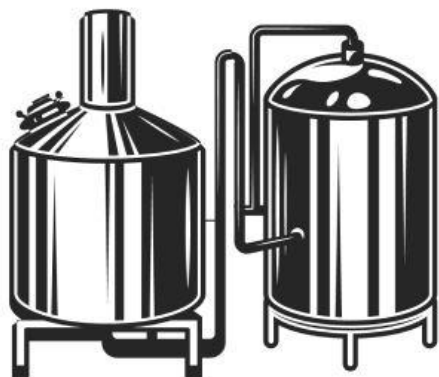
## 8.4.5 Priebek návrhu

### Zobrazovanie procesu varenia



V prvotnom návrhu zobrazovania varu sa obrazovka delila na dve časti, na ľavej sa nachádzalo zariadenie, na pravej priebek varenia. V prvých návrhoch boli taktiež inštrukcie podelené do blokov, tie ale mohli v istých prípadoch prebiehať aj naraz, a preto sa zakreslovali vedľa seba.

Prebiehajúci recept: České pivo



Nádoba 1

Nádoba 2

NEAKTÍVNE

V PROCESE VARENIA

Teplota: 70°C ↑

Teplota: 70°C ↓

Otáčky: ON

Otáčky: OFF

Doterajší čas varenia: 3:54

Predpokladaný čas varenia: 4:15

Dĺžka prebiehajúceho procesu: 0:12

Najbližšie stopky: neaktívne

Dokončené: premiestnenie do druhej nádoby

**Aktuálny proces: Ochlad' na 18 stupňov**

**Nasleduje: Pridaj kvasinky**

STOP

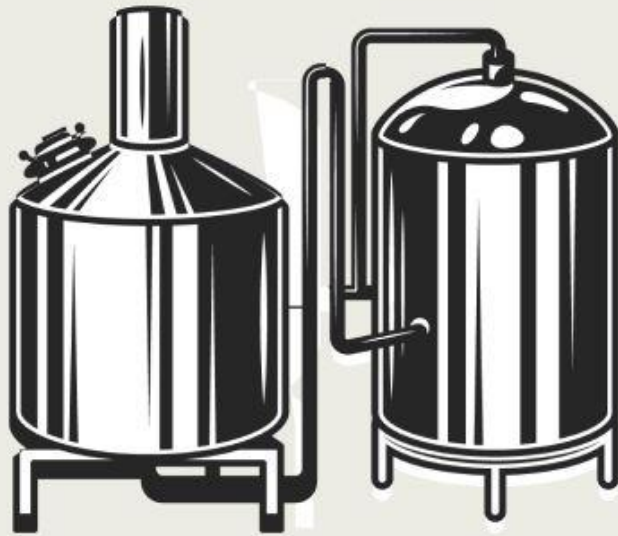
Aktívne ak je napríklad 10 minút so prídania ďalšej ingrediencie, alebo bude sa to miesať ešte x minút,...

Na kliknutie sa zobrazia podrobnosti k procesu ako sú rozpisane na frontend page

Vyskakovacie okienko na potvrdenie

You can scroll  
Could possibly be bigger and show time differently

Druhý návrh sa viac zameriaval na zobrazovanie inštrukcií, ako po sebe nasledujú. Zároveň neboli na obrazovke vždy zobrazované všetky inštrukcie, čo sa vykonalo najmä kvôli lepšiemu prehľadu na stránke. Ku každej nádobe sa pridali údaje o prebiehajúcych procesoch. Návrh taktiež obsahoval časovače. Tie boli veľmi vítanou funkcionalitou návrhu, a niektoré z nich pravdepodobne implementujeme v budúcich etapách projektu.



**CHAMBER 1**

Temperature:  
**50°C**  
Motor speed:  
**30 RPM**  
Heating:  
**ON**

**CHAMBER 2**

Temperature:  
**23°C**  
Motor speed:  
**0 RPM**  
Heating:  
**OFF**

**MENU**

SETUP

Motor speed 30 RPM

**Heat up Chamber 1**  
Now: 51.2°C Target: 70°C

IN PROGRESS

Wait 20 min

Motor speed 0 RPM

Valve OPEN

FINISH

Spojením týchto dvoch návrhov vznikol návrh 3. Na ľavej strane je obrázok zariadenia s informáciami k nádobám. Napravo sú vypísané prebiehajúce inštrukcie. Details k inštrukcii sú zobrazené, len keď práve prebieha.

Rozdiel medzi posledným návrhom a aktuálnym riešením je výzor pravého sidebaru. Do aktuálneho riešenia bol pridaný zoznam ingrediencií, tlačidlo na prerušenie a zastavenie procesu. Pri procese totiž vždy môže nastať situácia, kedy používateľ potrebuje varenie krátkodobo alebo dlhodobo pozastaviť.

# Tvorba nového receptu

## Nový recept ✎

---

**✎ Krok 1**

Zariadenie:  
 ▼

Inštrukcia:  
 ▼

Parametre:  
Teplota:  °C

+

Pridať novú inštrukciu...

---

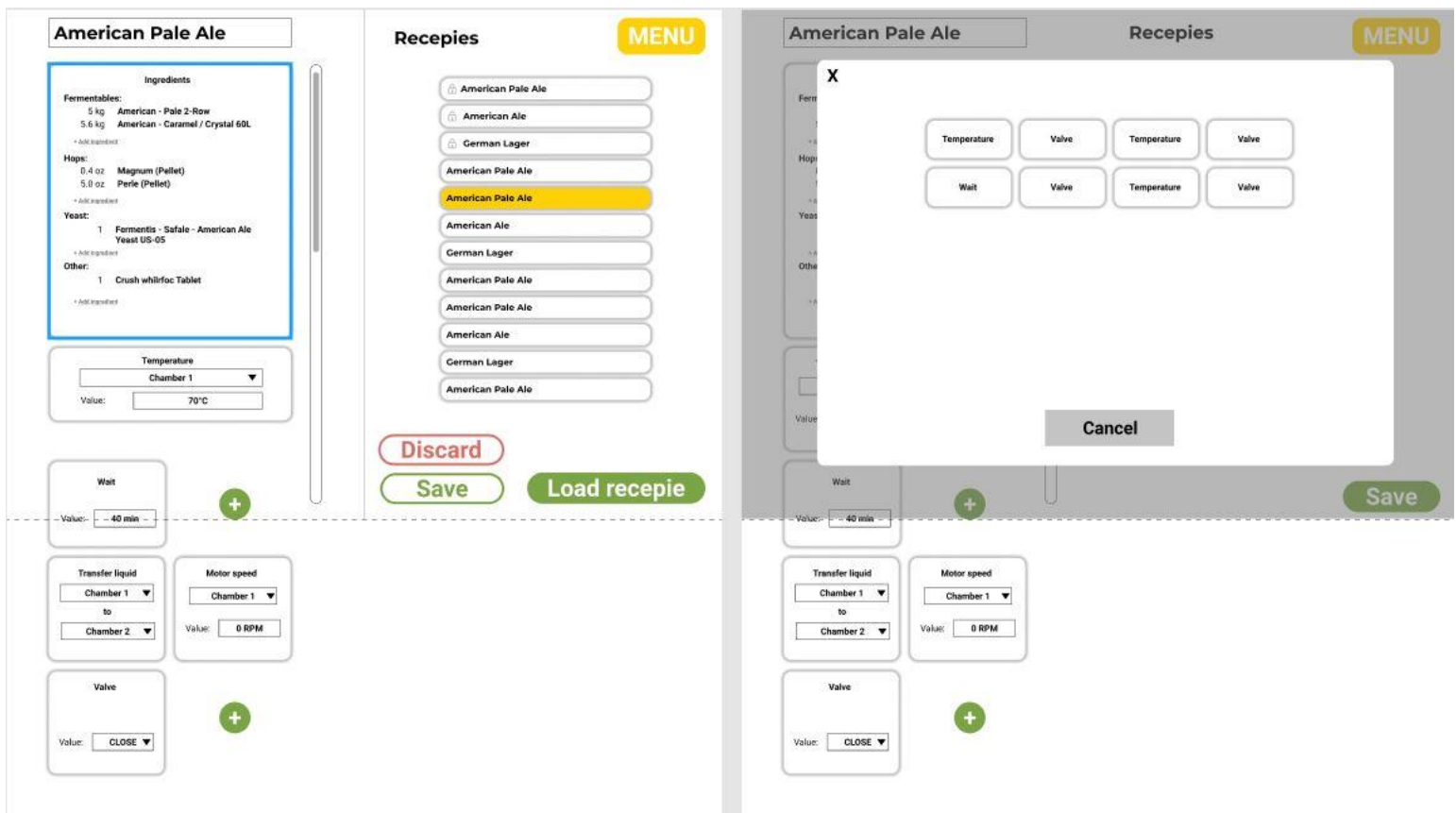
+

Pridať nový krok...

Zrušiť

Uložiť

Na prvom návrhu mali všetky inštrukcie rovnaký spôsob pridania. Používateľ si vybral typ zariadenia, inštrukciu a parametre. To ale samozrejme nepostačuje pre všetky typy inštrukcií, napríklad ak by sa jednalo o dve zariadenia (čo sa nemusí nutne stať v našom zadaní, ale vo všeobecnosti je to možné).



Preto vznikol druhý návrh. Pri ňom už mala každá inštrukcia vlastné "telo" prispôbené svojim požiadavkám. Pre pridanie novej inštrukcie bolo potrebné kliknúť na zelené plus, čím sa otvorilo vyskakovacie okno a zobrazila ponuka inštrukcií. Taktiež sa na tej istej obrazovke pridávali ingrediencie.

Rozdiel medzi posledným návrhom a aktuálnym riešením: Na jednej obrazovke sa nachádzalo priveľké množstvo informácií (ingrediencie aj inštrukcie naraz). Preto bola vytvorená osobitná obrazovka pre pridávanie ingrediencií a osobitná pre pridávanie inštrukcií. S využitím spôsobu spracovania inštrukcii taktiež nebol dôvod ukladať ich vedľa seba. Inštrukcie sa začali ukladať pod seba, a kategorizovať do blokov podľa jednotlivých fáz procesu. Názvy inštrukcií taktiež nemusia byť dostatočne výstižné, ku každej bol pridaný jej slovný popis.

Last updated on **16. 12. 2021** by **Michaela Nemcová**

# 8.5 Moduly

Link na repozitár - [Module](#)

## 8.5.1 Technológie

Používame ESP32 Dev board. Kód je písaný v **C++**.

Odpurúčaný setup pre ~~development~~ development je v [README](#) v rámci git repozitára.

## 8.5.2 Komunikácia s back-endom

Pre komunikáciu využijeme websocket a správy budú vyzerat' ako v [Podporované údaje](#).

Periodický update

```
{  
  "moduleId": 123,  
  "status": <status modulu>, // ERROR, OK  
  <podporované údaje>  
}
```

## 8.5.3 Ako v skratke funguje modul?

### Setup

Najprv si treba zadať aké zariadenia máme kde pripojené a dať ich do poľa všetkých zariadení (v budúcnosti by sme chceli, aby tento krok nebolo treba manuálne robiť pred nasadením modulu, ale dala by sa modulu poslať konfigurácia).

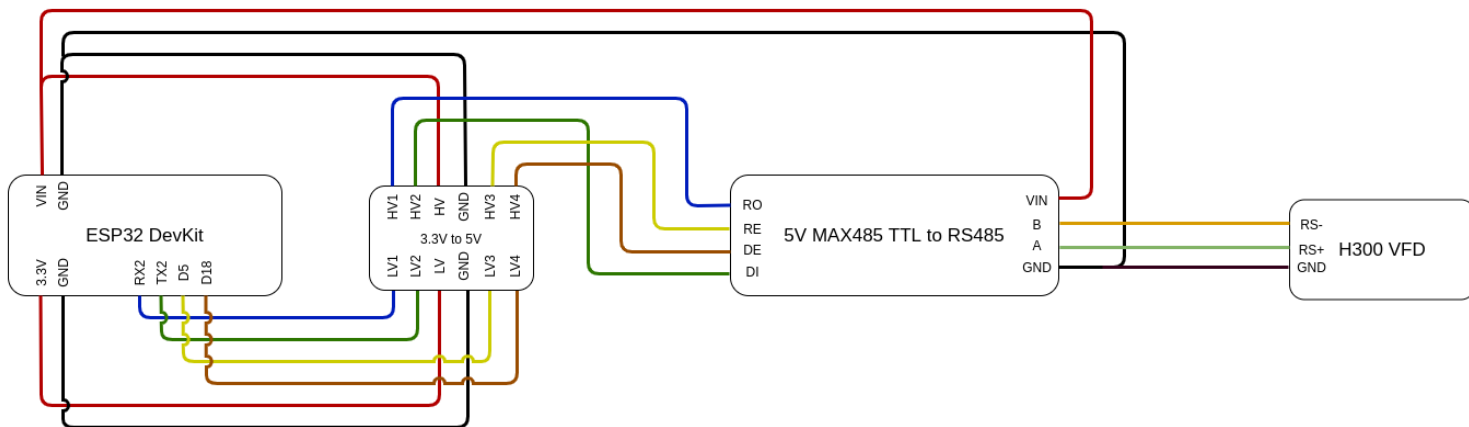
## Kód

Modul sa pripojuje na WiFi podľa konfigurácie a následne sa pripojuje cez WebSocket na backendový server. Modul potom periodicky posiela svoj aktuálny stav, a aj stav všetkých pripojených zariadení.

Na modul je možné poslať z backendu aj inštrukciu, ktorú má modul vykonať. Formát je zadefinovaný v Podporované funkcie. Modul následne vo svojich periodických updatoch posiela aj informáciu o aktuálnom stave danej inštrukcie.

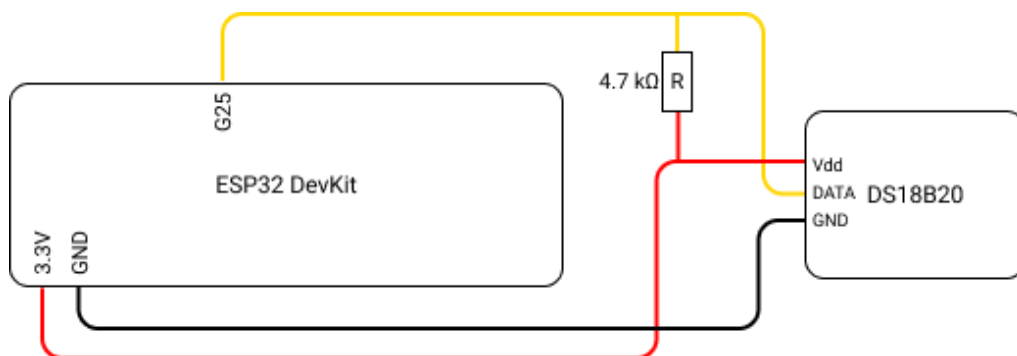
## Diagram zapojenia

### Zapojenie meniča H300



\*Originálny autor diagramu H300: <http://team18-19.studenti.fiit.stuba.sk/>

### Zapojenie teplomera DS18B20



## 8.5.4 Notes

- ♦ pridať viac diagramov zapojenia modulu a rôznych zariadení
- ♦ automatická kontrola WiFi spojenia (ak bolo prerušené)
- ♦ dynamická konfigurácia pripojených zariadení

*Last updated on **16. 12. 2021** by **Michaela Nemcová***



# 8.6 Podporované údaje

Tieto `interface` predstavujú kategórie, ktoré sú využívané aj v Podporované funkcie.

Zoznam podporovaných údajov systémom:

```
interface Temperature {
  temp: number;
  regulation_enabled: boolean;
  state: string;
  device: string; // "TEMP_1" or "TEMP_2"
}

interface Motor {
  speed: number;
  rpm: number;
  state: string;
  device: string; // "MOTOR_1" or "MOTOR_2"
}

interface Unloader {
  unloaded: boolean;
  state: string;
  device: string; // "FERMENTABLE", "YEAST", "HOPS", "OTHER"
}

interface Pump {
  enabled: boolean;
  state: string;
  device: string; // "PUMP_1"
}

interface Timer {
  remaining: number; // in seconds
  state: string;
  device: string; // "TIMER"
}
```

## 8.6.1 Príklad

Údaje sa budú posielat' tak, že najprv sa zdefinuje do akej kategórie patrí zariadenie, potom jeho bližšie údaje. "DEVICE" bude jedinečný názov naprieč všetkými zariadeniami z danej kategórie.

Príklad posielaných údajov

```
{
  "TEMPERATURE": [
    {
      "temp": 50,
      "regulation_enabled": true,
      "state": "IN_PROGRESS",
      "device": "TEMP_1"
    },
    {
      "temp": 21.5,
      "regulation_enabled": false,
      "state": "WAITING",
      "device": "TEMP_2"
    }
  ],
  "MOTOR": [
    {
      "speed": 30,
      "rpm": 25,
      "state": "WAITING",
      "device": "MOTOR_1"
    },
    {
      "speed": 0,
      "rpm": 0,
      "state": "WAITING",
      "device": "MOTOR_2"
    }
  ],
  "UNLOADER": [
    {
      "unloaded": true,
      "state": "WAITING",
      "device": "FERMENTABLE"
    }
  ],
}
```

```
{
  "unloaded": false,
  "state": "WAITING",
  "device": "YEAST"
},
{
  "unloaded": false,
  "state": "WAITING",
  "device": "HOPS"
},
{
  "unloaded": false,
  "state": "WAITING",
  "device": "OTHER"
}
],
"PUMP": [
  {
    "enabled": false,
    "state": "WAITING",
    "device": "PUMP_1"
  }
],
"TIMER": [
  {
    "remaining": 180,
    "state": "IN_PROGRESS",
    "device": "TIMER"
  }
]
}
```

Last updated on **5. 12. 2021** by **Peter Stríž**

# 8.7 Podporované funkcie

Zoznam podporovaných funkcií systémom:

Name	Input type	Units	Category	Code name	Type	Choices
Temperature	field	°C	TEMPERATURE	SET_TEMPERATURE	float	Chamber 1, Chamber 2
Motor	field	RMP	MOTOR	SET_MOTOR_SPEED	float	Motor 1, Motor 2
Transfer liquids	executable	-	PUMP	TRANSFER_LIQUIDS	-	Pump 1
Unload	executable	-	UNLOADER	UNLOAD	-	Fermentables, Yeast, Hops, Other
Wait	field	ms	SYSTEM	WAIT	float	-
Manual step	field	-	SYSTEM	MANUAL	string	-

- ♦ **Name** - pekný názov na Frontende
- ♦ **Code name** - ako sa bude v skutočnosti volať daná funkcia
- ♦ **Category** - je kategória, do ktorej patrí daná funkcia (využívané v Podporované údaje)
- ♦ **Input type** - či bude mať možnosť meniť parametre alebo sa daná funkcie iba vykonáva
- ♦ **Type** - validácia na Frontende, v skutočnosti všade posielam string
- ♦ **Choices** - aké možnosti budú ponúkané - ktorý konkrétny prvok sa má spustiť
- ♦ **Code choices** - predstavujú jednotlivé zariadenia a ich "skutočné" meno - toto meno sa bude zhodovať naprieč systémom s menom ktoré bude ukazovať stav jednotlivých senzorov

V skutočnosti Frontend ukladá a prijíma iba: **Code name**, **Code choices**, **Category** a hodnoty **parametra**.

Ľtatné veci si už sám vyhľadá a doplní podľa špecifikácie.

## 8.7.1 Štruktúra posielaných údajov Backend --> Modul

### Inštrukcia

```
{
  "type": "instruction",
  "moduleId": <id modulu>,
  "instruction": <code name pre inštrukciu>,
  "param": <hodnota parametru>,
  "category": <kategória inštrukcie/zariadenia>,
  "device": <zariadenie>,
}
```

#### Ukážka

```
{
  "moduleId": 1,
  "instruction": "SET_MOTOR_SPEED",
  "param": 30,
  "category": "MOTOR",
  "device": "MOTOR_1"
}
```

### Abort

```
{
  "type": "abort"
}
```

## 8.7.2 Štruktúra posielaných údajov Backend --> Frontend

### Inštrukcia

```
{
  "id": <id inštrukcie>,
  "recipeId": <id receptu>,
  "templateId": <id template pre inštrukciu>,
  "codeName": <code name pre inštrukciu>, // aj toto sa bude mapovať na krajší názov
  "param": <hodnota parametru>,
  "category": <kategória inštrukcie/zariadenia>,
  "optionCodeName": <zariadenie>, // na FE sa bude mapovať na krajší názov - napr.
  "TEMP_1" -> "Nádoba 1",
  "blockId": <id bloku>, // ak by boli dva názvy rovnaké, ale chceme to mať ako dva
  rôzne bloky
  "blockName": <názov bloku>,
  "ordering": <poradové číslo>,
}
```

### Ukážka

```
{
  "id": 234,
  "recipeId": 123,
  "templateId": 1,
  "codeName": "SET_MOTOR_SPEED",
  "param": 30,
  "category": "MOTOR",
  "optionCodeName": "MOTOR_1",
  "blockId": 1,
  "blockName": "Fermentation",
  "ordering": 2
}
```

# Template pre inštrukciu

FE si vie získať ako vyzerajú všetky inštrukcie, ktoré sú podporované systémom.

```
{
  "id": <id template>,
  "codeName": <code name pre inštrukciu>,
  "name": <pekny nazov instrukcie>,
  "category": <kategoria>,
  "units": <jednotka>,
  "inputType": <typ inputu>,
  "description": <popis>,
  "options": [
    {
      "id": <id zariadenia>,
      "name": <pekny nazov zariadenia>,
      "codeName": <zariadenie>
    },
    ...
    ...
  ]
}
```

## Ukážka

```
{
  "id": 1,
  "codeName": "SET_TEMPERATURE",
  "name": "Set temperature",
  "category": "TEMPERATURE",
  "units": "°C",
  "inputType": "float",
  "description": "Sets temerature for selected chamber",
  "options": [
    {
      "id": 1,
      "name": "Chamber 1",
      "codeName": "TEMP_1"
    },
    {
      "id": 2,
      "name": "Chamber 2",
      "codeName": "TEMP_2"
    }
  ]
}
```



## 8.7.3 Štruktúra posielanej inštrukcie Frontend --> Backend

```
{  
  "templateId": <id vzoru pre danú inštrukciu>,  
  "blockName": <nazov bloku>,  
  "param": <hodnota parametru>,  
  "optionCodeName": <zariadenie>,  
  "ordering": <poradové číslo>,  
}
```

### Ukážka

```
{  
  "templateId": 123,  
  "blockName": "Fermentation",  
  "param": 85,  
  "optionCodeName": "TEMP_1",  
  "ordering": 1  
}
```

## TODO

- domyslieť vytiahnutie chmeľu
- ukončenie

Last updated on **8. 12. 2021** by **Peter Stríž**

# 8.8 Testovací server

Testovací server slúži na simuláciu správania modulov. Od vývoja modulov závisí hlavne vývoj backendu a aj frontendu, preto sme sa rozhodli vytvoriť testovací server, ktorý simuluje správanie reálneho modulu, čím napomáha k rýchlejšiemu vývoju práve backendu a frontendu.

Server posiela a prijíma údaje cez WebSocket. Posielanie údajov a prijímanie inštrukcií prebieha rovnako ako je zadané v API.

Server po pripojení na backend posiela periodicky namodelované dáta. Server taktiež prijíma inštrukcie z backendu, na ktoré odpovedá zmenenými dátami podľa jednotlivých inštrukcií. Zoznam podporovaných inštrukcií s konkrétnymi parametrami sa nachádza v Podporované funkcie.

V prípade zmien na backende so spusteným testovacím serverom, nie je potrebné tento testovací server spúšťať znova, automaticky sa pripojí späť na backend.

## 8.8.1 Príklad komunikácie

Server posiela údaje typu:

```
{
  "moduleId": "test-module-1",
  "state": "ok",
  "TEMPERATURE": [
    {
      "TEMP": 50,
      "REGULATION_ENABLED": true,
      "STATE": "IN_PROGRESS",
      "DEVICE": "TEMP_1"
    },
    {
      "TEMP": 21.5,
      "REGULATION_ENABLED": false,
      "STATE": "WAITING",
      "DEVICE": "TEMP_2"
    }
  ],
  "MOTOR": [
    {
      "SPEED": 30,
```

```

    "RPM": 25,
    "STATE": "WAITING",
    "DEVICE": "MOTOR_1"
  },
  {
    "SPEED": 0,
    "RPM": 0,
    "STATE": "WAITING",
    "DEVICE": "MOTOR_2"
  }
],
"UNLOADER": [
  {
    "UNLOADED": true,
    "STATE": "WAITING",
    "DEVICE": "FERMENTABLE"
  },
  {
    "UNLOADED": false,
    "STATE": "WAITING",
    "DEVICE": "YEAST"
  },
  {
    "UNLOADED": false,
    "STATE": "WAITING",
    "DEVICE": "HOPS"
  },
  {
    "UNLOADED": false,
    "STATE": "WAITING",
    "DEVICE": "OTHER"
  }
],
"PUMP": [
  {
    "ENABLED": false,
    "STATE": "WAITING",
    "DEVICE": "PUMP_1"
  }
]
}

```

Príklad inštrukcie, ktorú server prijme z backendu:

```
{
  "moduleId": "test-module-1",
  "type": "instruction",
  "category": "TEMPERATURE",
  "device": "TEMP_2",
  "instruction": "SET_TEMPERATURE",
  "parameter": "100"
}
```

Na základe prijatej inštrukcie prebehne buď okamžitá, alebo postupná zmena dát. Po prijatí inštrukcie vyššie môžu odosielané dáta vyzerat' nasledovne:

```
{
  "moduleId": "test-module-1",
  "state": "ok",
  "TEMPERATURE": [
    {
      "TEMP": 50,
      "REGULATION_ENABLED": true,
      "STATE": "IN_PROGRESS",
      "DEVICE": "TEMP_1"
    },
    {
      "TEMP": 90,
      "REGULATION_ENABLED": true,
      "STATE": "IN_PROGRESS",
      "DEVICE": "TEMP_2"
    }
  ],
  "MOTOR": [
    {
      "SPEED": 30,
      "RPM": 25,
      "STATE": "WAITING",
      "DEVICE": "MOTOR_1"
    },
    {
      "SPEED": 0,

```

```

    "RPM": 0,
    "STATE": "WAITING",
    "DEVICE": "MOTOR_2"
  }
],
"UNLOADER": [
  {
    "UNLOADED": true,
    "STATE": "WAITING",
    "DEVICE": "FERMENTABLE"
  },
  {
    "UNLOADED": false,
    "STATE": "WAITING",
    "DEVICE": "YEAST"
  },
  {
    "UNLOADED": false,
    "STATE": "WAITING",
    "DEVICE": "HOPS"
  },
  {
    "UNLOADED": false,
    "STATE": "WAITING",
    "DEVICE": "OTHER"
  }
],
"PUMP": [
  {
    "ENABLED": false,
    "STATE": "WAITING",
    "DEVICE": "PUMP_1"
  }
]
}

```

Last updated on **16. 12. 2021** by **Michaela Nemcová**