VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

SOFTWAROVÝ NÁSTROJ PRO VIZUÁLNÍ SPECIFIKACI ŘÍZENÍ PRŮBĚHU LABORATORNÍCH EXPERIMENTŮ

SOFTWARE TOOL FOR VISUAL SPECIFICATION OF LABORATORY EXPERIMENTS EXECUTION FLOW

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAN TRUHLÁŘ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Truhlář Jan, Bc.**
Program: Informační technologie Obor: Počítačové a vestavěné systémy
Název: **Softwarový nástroj pro vizuální specifikaci řízení průběhu laboratorních experimentů**
Software Tool for Visual Specification of Laboratory Experiments Execution Flow
Kategorie: Uživatelská rozhraní
Zadání:

1. Seznamte se s principem činnosti specializovaného laboratorního zařízení pro analýzu léčiv. Prostudujte formát komunikačního protokolu mezi obslužným software a stávající řídicí jednotkou.
2. Seznamte se s principy konceptu tzv. vizuálního programování a dostupnými softwarovými nástroji pro tento účel. Připravte krátkou přehledovou studii na toto téma.
3. S výzkumným týmem z VFU Brno detailně konzultujte charakter typických činností vykonávaných během experimentů s laboratorním přístrojem pro analýzu léčiv.
4. Připravte návrh vzhledu a rozložení funkčních prvků uživatelského rozhraní nástroje pro vizuální specifikaci řízení průběhu laboratorních experimentů.
5. S využitím vhodného modelovacího nástroje (např. UML) navrhněte strukturu implementační realizace prostředí. Dbejte na modulární přístup a snadnou rozšiřitelnost.
6. Proveďte implementaci softwarového nástroje dle poznatků z bodů 4) a 5) zadání s využitím vhodné multiplatformní grafické knihovny, např. Qt či Java Swing.
7. Vytvořené řešení detailně otestujte. Zhodnoťte dosažené výsledky a navrhněte případná rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Šimek Václav, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 20. dubna 2020

Abstrakt

Cílem této práce je navrhnout a vytvořit softwarový nástroj, který umožní vizuální specifikaci průběhu laboratorních experimentů na laboratorním přístroji GOLEM, který vyvíjí výzkumná skupina Ústavu technologie léků Veterinární a farmaceutické univerzity v Brně. Aplikace byla realizována na základě moderních webových technologií, nástrojů pro jejich návrh a implementaci. Vytvořené řešení poskytuje ergonomické uživatelské rozhraní s vysokou mírou modularity a možností snadného rozšíření při dalším vývoji laboratorního přístroje. Přínosem této práce je zejména poskytnutí na míru vytvořeného nástroje, který výzkumné skupině umožní efektivní a ergonomické ovládání přístroje při provádění experimentů.

Abstract

Aim of this work is to design and create a software tool, which will allow visual specification of laboratory experiments flow on device called GOLEM. This device was created by research group from Department of Chemical Drugs, University of Veterinary and Pharmaceutical Sciences Brno. This software was based on modern web technologies, tools for their design and implementation. The software created in this work offers ergonomic and the highly modular user interface which can be easily extended in case of upgrade. Main benefit of this work is providing research group with a new software tool which will help them perform experiments more effectively.

Klíčová slova

GOLEM, VFU Brno, vizuální programování, GUI, uživatelské rozhraní, řízení průběhu experimentu, návrh rozhraní, disoluce

Keywords

GOLEM, UVPS Brno, visual programming, GUI, user interface, visual specification of experiment execution flow, disolution, user interface design

Citace

TRUHLÁŘ, Jan. *Softwarový nástroj pro vizuální specifikaci řízení průběhu laboratorních experimentů*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek

Softwarový nástroj pro vizuální specifikaci řízení průběhu laboratorních experimentů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Václava Šimka. Další informace mi poskytli členové výzkumné skupiny Ústavu technologie léků Veterinární a farmaceutické univerzity v Brně, jmenovitě PharmDr. Martin Čulen, Ph.D., Mgr. Tomáš Bílik a PharmDr. Jakub Vysloužil, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Truhlář
30. července 2020

Poděkování

Tímto bych chtěl poděkoval vedoucímu mé práce, Ing. Václavu Šimkovi, za rady, pomoc a velmi přátelský přístup při vedení této práce. Dále bych rád poděkoval Bc. Zbyšku Vodovi za spolupráci a vzájemnou pomoc při realizaci našich diplomových prací. Také děkuji vědeckým pracovníkům z Veterinární a farmaceutické univerzity Brno, zejména PharmDr. Martinu Čulenovi, Ph.D., za veškeré poskytnuté informace, konzultace a čas strávený při návrhu a testování této diplomové práce. V neposlední řadě děkuji také mé rodině a přátelům za veškerou pomoc a podporu během všech fází této práce i celého studia.

Obsah

Kapitola 1

Úvod

Nedílnou součástí vývoje všech nových léků s perorálním podáním je analýza mechanismů, jakými se jejich účinné látky uvolňují v trávicí soustavě. Za tímto účelem vzniklo mnoho vědeckých přístrojů, jejichž cílem je napodobit tyto procesy. Jedním z těchto přístrojů je i GOLEM, který vyvinula výzkumná skupina Ústavu technologie léků na Veterinární a farmaceutické univerzitě v Brně.

Cílem této práce je navrhnout a vytvořit softwarový nástroj, který umožní členům výzkumné skupiny efektivní prováděni experimentů na tomto přístroji. Návrh nového rozhraní vychází z již existujícího, ale nevyhovujícího rozhraní, které bylo vytvořeno při výrobě tohoto přístroje v roce 2010.

Návrh nového rozhraní byl realizován pomocí interaktivního modelu aplikace (mockup) jehož finální podoba je výsledkem mnoha iterací testování a úprav dle přání a připomínek výzkumné skupiny. Hlavní důraz při návrhu a vývoji nového software je kladen na ergonomické ovládání, které umožní obsluze přístroje efektivně provádět experimenty.

K realizaci byly použity moderní webové technologie, nástroje pro jejich návrh a implementaci. Webové technologie byly vybrány zejména pro jejich multiplatformnost, jednoduchou rozšiřitelnost a vysokou modularitu. Komunikace mezi webovou aplikací a zařízením je realizována metodou zasílání zpráv na aplikační rozhraní řídící jednotky. Aplikační rozhraní bylo navrženo a implementováno ve spolupráci s Bc. Zbyškem Vodou, jehož diplomová práce se zaměřuje na návrh a vytvoření nové řídící jednotky.

Přínosem této práce je zejména poskytnutí na míru vytvořeného nástroje, který výzkumné skupině umožní efektivní a pohodlné ovládání přístroje při řízení průběhu experimentů.

V rámci druhé kapitoly jsou představeny základní anatomické rysy lidské trávicí soustavy a také úvod do farmakokinetiky, vědy zabývající se studiem mechanismu vstřebávání léků. Třetí kapitola je pak věnována podrobné analýze laboratorního přístroje GOLEM dle dokumentace dodané výzkumnou skupinou a osobních konzultací. Čtvrtá kapitola slouží jako teoretický základ pro následující fázi návrhu inovovaného uživatelského rozhraní a jako úvod do moderních webových technologií, zejména frameworku Vue. v páté kapitole je provedena analýza požadavků zadavatele, návrh inovovaného uživatelského rozhraní a vybrány technologie pro jeho následnou implementaci. Šestá kapitola shrnuje postup implementace a také slouží jako dokumentace klíčových částí nově vytvořeného systému. Sedmá kapitola se věnuje procesu testování, vyhodnocování a opravám nedostatků, které testování odhalilo. Poslední kapitola pak shrnuje dosažené výsledky a uvádí plán dalšího postupu inovace přístroje GOLEM.

Kapitola 2

Základy farmakokinetiky

K pochopení účelu a principu funkce přístroje GOLEM je nezbytné uvést potřebné teoretické základy z oblasti farmakokinetiky, biologie a analýzy léčiv. Z tohoto důvodu se v této kapitole věnuji základním pojmům a informacím z této vědní oblasti. Text této kapitoly vychází ze zdrojů [16], [20].

2.1 Lidská trávicí soustava

Lidská trávicí soustava je tvořena dvěma typy orgánů. Prvním typem jsou orgány trávicí trubice, které mají za úkol zejména rozmělňování potravy (žaludek) a vstřebávání živin z potravy (tenké střevo). Druhou skupinu pak tvoří žlázy (například slinivka břišní a játra), které vylučují enzymy a jiné látky nezbytné k trávení.

Jednotlivé orgány a jejich vzájemné propojení můžete vidět na obrázku 2.1, který rovněž naznačuje směr, kterým potrava putuje během procesu trávení. V následujících podkapitolách jsou uvedeny podrobnější informace k orgánům, které jsou důležité z hlediska pochopení přístroje GOLEM.

2.1.1 Žaludek (stomach)

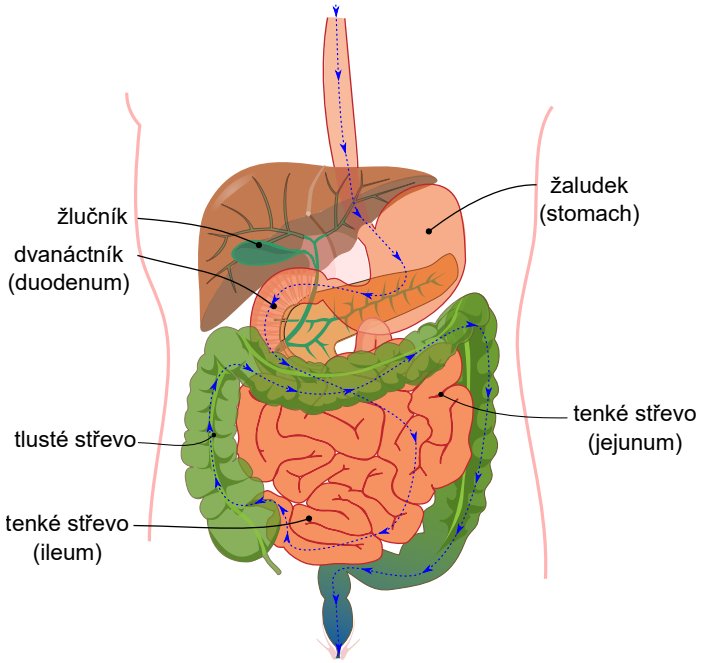
Žaludek je rozšířená část trávicí trubice, jejímž hlavním úkolem je mechanicko-chemické trávení pomocí kontrakcí žaludečního svalstva, působení kyseliny chlorovodíkové a trávicích enzymů. Pro ochranu před působením trávicích šťáv je vnitřní část žaludku potažena alkalickým hlenem. Po zpracování potravy obsah žaludku postupně prochází do dvanáctníku (duodenum). Podmínky v této části trávicího traktu popisuje tabulka 2.1.

Hodnota pH	2 při aktivním trávení, 4-5 v klidovém stavu
Chemické složení	HCl, NaCl, KCl, enzymy
Teplota	typicky 37°C, ale může být ovlivněna teplotou potravy

Tabulka 2.1: **Podmínky panující v žaludku** [17]

2.1.2 Dvanáctník (duodenum)

Dvanáctník je počáteční část tenkého střeva, která následuje bezprostředně za žaludkem. Vstupují do něj také vývody žlučníku a slinivky břišní, kterými jsou zde přiváděny trávicí šťávy. S jejich pomocí dochází k neutralizaci kyselého výstupu žaludku, enzymatickému



Obrázek 2.1: **Trávicí soustava člověka** – schéma trávicí soustavy s naznačeným směrem průchodu potravy (šipky s modrou přerušovanou čarou) jednotlivými orgány. (Původní verze obrázku byla převzata z https://bit.ly/39AuBOG)

štěpení a vstřebávání některých živin, zbytek obsahu je následně posunut do tenkého střeva. Podmínky v této části trávicího traktu popisuje tabulka 2.2.

Hodnota pH	5
Chemické složení	NaCl, KCl, enzymy, bikarbonátový pufr
Teplota	37°C

Tabulka 2.2: **Podmínky panující ve dvanáctníku** [17]

2.1.3 Tenké střevo a jeho části (jejunum, ileum)

Tenké střevo je dutá trubice o průměru 3-4 cm, jejímž hlavním účelem je enzymatické štěpení potravy a vstřebávání živin. Současně s živinami zde dochází k nejintenzivnějšímu vstřebávání účinných látek z léčiv. Tenké střevo se dělí do dvou částí, které mezi sebou přecházejí bez jasné hranice.

První z těchto částí je jejunum, ve kterém dochází k nejintenzivnějšímu vstřebávání látek díky vysokému počtu řas. Jejunum má větší šířku než ileum, ale je kratší. Chemické podmínky shrnuje tabulka 2.3.

Druhou částí tenkého střeva nazýváme ileum. Jeho délka oproti jejunu je sice větší, avšak počet řas postupně klesá, až nakonec mizí, což má za následek snižující se intenzitu vstřebávání živin. Po této části tenkého střeva dochází k napojení na tlusté střevo. Podmínky panující v ileu shrnuje tabulka 2.4.

Hodnota pH	6
Chemické složení	NaCl, KCl, bikarbonátový pufr
Teplota	37°C

Tabulka 2.3: **Podmínky panující v jejunum** [17]

Hodnota pH	7
Chemické složení	NaCl, KCl, bikarbonátový pufr
Teplota	37°C

Tabulka 2.4: **Podmínky panující v ileum** [17]

2.2 Lékové formy

Léková forma definuje základní fyzické, chemické a tvarové parametry léku. Nejběžnější lékovou formou jsou tablety o různých hmotnostech a tvarech, můžeme se však setkat také s gely, roztoky, mastmi, náplastmi, ale i mnoha dalšími formami, které jsou shrnuty v následujícím seznamu[12].

1. Pevné lékové formy
    - granuláty, zásypy, prášky pro injekce
    - tablety
    - tobolky
    - čípky a globule
2. Polotuhé lékové formy
    - masti
    - pasty
    - gely
3. Kapalné lékové formy
    - roztoky, kapky, sirupy pro vnitřní užití
    - kloktadla a spreje k aplikaci v ústech
    - roztoky, emulze, spreje, pěny na kůži
    - oční kapky a suspenze
    - parenterální lékové formy – injekce a infuze
4. Transdermální náplasti

Použitá léková forma do značné míry určuje způsob a rychlost vstřebání daného léčivého přípravku, což ji činí jedním z nejdůležitějších faktorů, který je potřeba při studiu uvolňování léčivých látek zohlednit.

Kapitola 3

Analýza stávajícího řešení

Pro vytvoření kvalitního uživatelského rozhraní je nezbytné detailně pochopit princip fungování daného přístroje a také postupy, které jeho obsluha používá při provádění experimentů. Z tohoto důvodu se v této kapitole zabývám detailní analýzou principů funkce, hardwarových a softwarových částí i způsobu, jakým obsluha přístroj používá při provádění experimentů. Výsledky analýzy budou následně použity v dalších kapitolách jako základ pro návrh nového uživatelského rozhraní.

Obrázek 3.1: **GOLEM 2** – fotografie přední strany laboratorního přístroje [24]

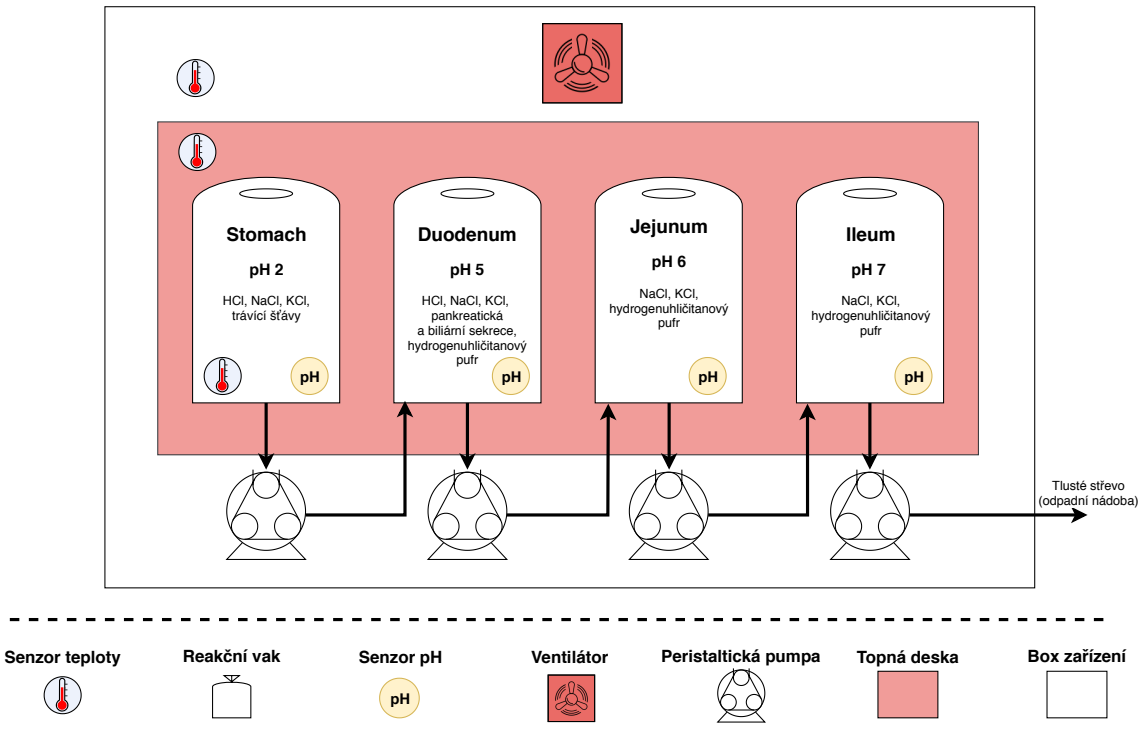
3.1 Analýza přístroje GOLEM 2

Přístroj GOLEM 2, jehož fotografii můžete vidět na obrázku 3.1, byl vyvinut v rámci grantu 319/2018/FaF [4] Veterinární a farmaceutické univerzity v Brně Ústavem organické chemie a biochemie Akademie věd České republiky. Původní verze byla vyvinuta v roce 2011, druhá verze přístroje pak byla vyvinuta v roce 2013. Kapitola vychází z dokumentace [17] poskytnuté výzkumnou skupinou, článků [23], [20], [26] a osobních konzultací.

Jedná se o disoluční přístroj, jehož účelem je biorelevantně napodobit způsob rozpouštění různých lékových forem v lidské trávicí soustavě a umožnit vědcům analýzu farmakokinetiky léčiv. Oproti jiným přístrojům pro analýzu disoluce se snaží věrně napodobit chemické prostředí i pochody v trávicí soustavě.[5]

3.1.1 Princip funkce

Princip funkce přístroje (viz obrázek 3.2) je odvozen od lidského trávicího traktu, jehož vybrané části simuluje.[17] Těmito částmi, které můžete vidět v kontextu lidského těla na obrázku 2.1, jsou žaludek (stomach), dvanáctník (duodenum), a dvě části tenkého střeva (jejunum a ileum). Tyto části jsou v přístroji zastoupeny čtyřmi reakčními vaky, které jsou při zahájení experimentu naplněny zvoleným množstvím roztoků, který napodobuje prostředí v dané části trávicího traktu. První z vaků (stomach) při provádění experimentu obsahuje tabletu, nebo jinou lékovou formu, jejíž disoluce má být studována.[26]



Obrázek 3.2: – **zjednodušené schéma funkce přístroje** Schéma popisuje základní části přístroje Golem, jejich umístění a standardní způsob propojení reakčních vaků

Simulace peristaltických pohybů

Ve vacích dochází k postupnému promíchávání jejich obsahu pomocí speciálních míchacích lamel ve tvaru písmene V, které imitují kontrakce svalů. Lamely jsou umístěny ve středu vaků a jejich rychlost je možné nastavit pro každý vak nezávisle. Detaily konstrukce míchacího systému jsou uvedeny v kapitole 3.2. Putování potravy trávicí soustavou je simulováno pomocí soustavy silikonových hadiček a peristaltických pump, které umožňují přesné přečerpávání zvoleného množství tekutiny.

Obvyklým způsobem propojení reakčních vaků je série odpovídající trávicímu traktu, avšak pro speciální experimenty mohou být zvoleny i jiná zapojení. Standardní schéma experimentu je znázorněno na obrázku 3.2. V průběhu experimentu jsou obsluhou přístroje odebírány vzorky tekutiny z jednotlivých vaků, jejichž následnou analýzou je možné studovat vývoj koncentrace účinných látek v závislosti na čase.

Řízení pH

Jelikož je v jednotlivých částech trávicího traktu rozdílná hodnota pH, dochází při přečerpávání tekutiny mezi reakčními vaky ke změnám této hodnoty. Z tohoto důvodu je nezbytné monitorovat pH ve všech vacích, aby bylo možné udržet v jednotlivých vacích konstantní podmínky přidáním korekčních roztoků pomocí pístových pump.

Simulace sekrece enzymů

Další součástí přístroje je pak systém otočných selekčních ventilů a pístových pump, které mohou do vaků vstřikovat enzymy, či jiné látky dle potřeby. Tímto způsobem je simulována sekrece trávicích šťáv.

Udržování teploty

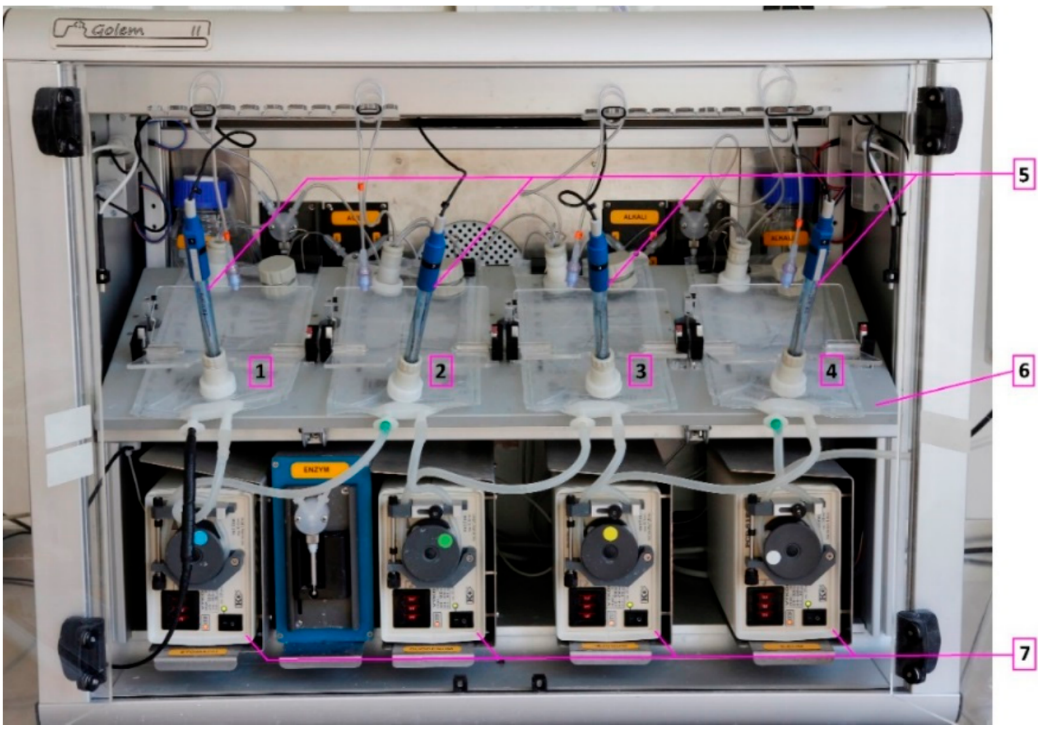
Pro dosažení a udržení konstantní teploty během experimentu jsou použity dva automatické systémy vytápění. Prvním z nich je topná deska, jejímž hlavním úkolem je udržovat konstantní teplotu tekutiny uvnitř reakčních vaků, druhým systémem je pak horkovzdušný ventilátor, jehož úkolem je udržovat teplotu vzduchu a všech částí uvnitř přístroje.

3.2 Technické provedení

Z hlediska technického provedení se jedná o box tvořený hliníkovým rámem s pláštěm vyrobeným z průhledného plexiskla. Výjimkou je zadní strana přístroje, která je dvojitá a je vyrobena z lakovaného plechu. První z těchto stěn je umístěna řídící elektronika a elektroinstalace, druhá pak tvoří zadní plášť zařízení. Přední stěna je tvořena dvěma křídly z plexiskla s madly, která jsou k rámu upevněny pomocí pantů. Tato otevíratelná přední stěna umožňuje přístup k vnitřním částem přístroje a zároveň brání úniku tepla a tím i nechtěné změně podmínek v průběhu experimentu.

Po stránce vestavěných systémů se jedná o zařízení složené z jednoho řídícího mikrokontroléru, kterým je zde Arduino UNO a několika podpůrných řídících desek. Mikrokontrolér obousměrně komunikuje po sériové lince s počítačem operátora pomocí USB sběrnice. Vyhodnocené povely poté mikrokontrolér distribuuje ke svým periferiím, které řídí buď přímo pomocí GPIO portů (spínání vytápění, peristaltická čerpadla), sériovou linkou RS485 (pístové pumpy a otočné selekční ventily), nebo s nimi komunikuje po I2C sběrnici (A/D pře-

vodník MCP3424 pro měření). Jednotlivé části a jejich rozmístění můžete vidět na obrázku 3.3. [17]



Obrázek 3.3: **Fotografie přední strany laboratorního přístroje GOLEM 2** – (1) stomach (žaludek), (2) duodenum (dvanáctník), (3) jejunum, (4) ileum, (5) pH sondy, (6) topná deska, a (7) peristaltické pumpy [23]

3.2.1 Části přístroje

Přístroj v sobě spojuje do funkčního celku několik různých typů zařízení a na míru vyrobených součástí, se kterými se v praxi běžně nesetkáme. Proto je tato kapitola věnována jejich popisu, mechanismu funkce, technickému provedení a vlastnostem.

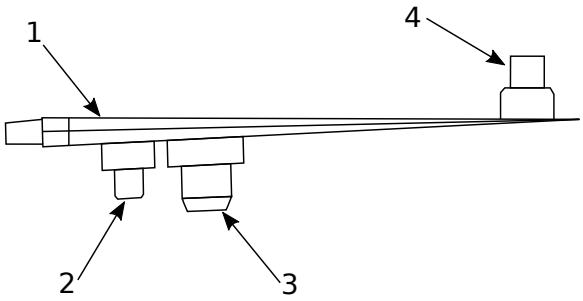
Reakční vak

Jedná se o průhledný plastový vak s konektory pro připojení vstupů, výstupů, titračních a vzorkovacích vývodů, pH sondy a teploměru. Všechny konektory jsou opatřeny plastovou zátkou, kterou mohou být uzavřeny v případě nevyužití daného portu. Vstupně-výstupní konektory mohou být použity jak pro tekutiny, tak pro zasunutí teploměru.[17]

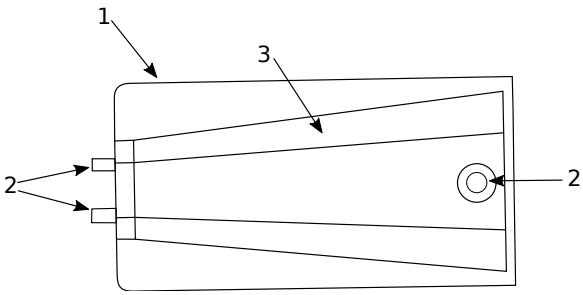
Vak může mít pomocí svarů horní a spodní strany vytvořeny různé tvary přepážek, které upravují proudění tekutiny při pohybu míchacího zařízení. Tímto způsobem je možné ovlivnit požadovaným způsobem rozpouštění léku.

Otočný selekční ventil CAVRO

Selekční ventil funguje na základě servomotoru umístěného v zadní části a otočeného mechanismu s trubičkami, které dokáže při otočení změnit směr průtoku kapalin. Selekční

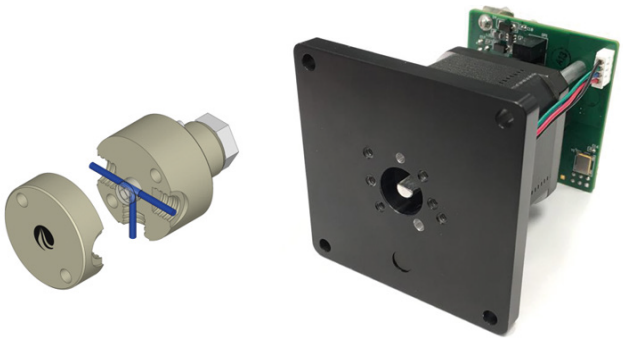


Obrázek 3.4: **Boční pohled na reakční vak** – (1) tělo vaku, (2) titrační a vzorkovací přívody, (3) port pH sondy (4) vstup / výstup



Obrázek 3.5: **Pohled shora na reakční vak** – (1) tělo vaku, (2) vstup / výstup (3) svařovaný spoj fungující jako přepážka

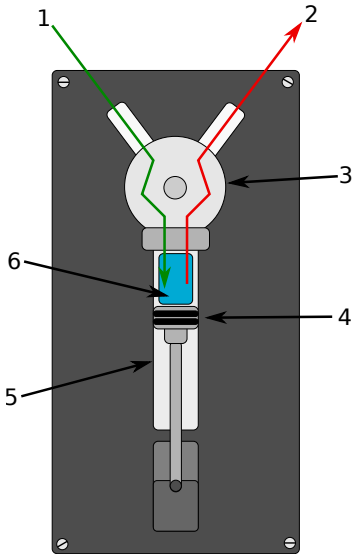
mechanismus je vyroben z chemicky odolných materiálů, aby nedocházelo k jeho porušení vlivem protékajících chemických látek. [14] [2]



Obrázek 3.6: **Otočný selekční ventil** – v levé části obrázku se nachází model selekčního ventilu se třemi porty, v pravé částí se pak nachází řídící jednotka s motorem [14]

Pístová pumpa CAVRO

Jedná se o specializované laboratorní zařízení tvořené krokovým motorem, selekčním ventilem, pístem a skleněným válcem. Zařízení je ovládáno pomocí zasílání zpráv skrz rozhraní RS232 a zpracovává pokyny pro načerpání daného množství tekutiny do válce ze vstupního portu a následně vstříknutí do výstupního portu přepnutím selekčního ventilu.

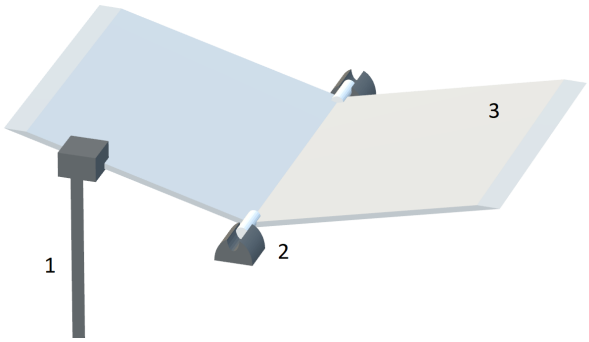


Obrázek 3.7: **Pístová pumpa** – (1) vstup, (2) výstup, (3) otočný selekční ventil, (4) píst, (5) válec (6), čerpaná kapalina

Míchací zařízení

Jedná se o na míru vyrobené míchací křídlo z průhledného plexiskla, které je uchyceno k topné desce pomocí západkového mechanismu. Samotný proces míchání je zajištěn díky servomotoru, který je umístěn pod topnou deskou a pohybuje s míchacím křídlem pomocí táhla.

Míchací křídlo je upevněno nad deskou v dostatečné výšce, aby byl při jeho pohybu umožněn průtok tekutiny mezi horní a spodní částí vaku. Míchací zařízení je ovládáno autonomní řídící jednotkou, která přijímá příkazy ve formě nastavené rychlosti míchání v rozsahu 0–7, kdy 7 znamená nejvyšší rychlost.[17]



Obrázek 3.8: **Boční pohled na model míchacího zařízení.** (1) táhlo míchacího zařízení, (2) uchycení míchacího křídla (3) míchací křídlo

Peristaltické čerpadlo

Jedná se o čerpadlo fungující na bázi periodického stlačování a uvolňování silikonové hadičky pomocí otáčejícího se kola s válečky. Při otáčení kola dochází díky válečkům ke stlačení silikonové hadičky na vstupu a postupnému vytlačování určitého množství tekutiny. Díky znalosti průměru hadičky a rychlosti otáčení lze přesně určit přečerpané množství za jednotku času.[3]

Obrázek 3.9: **Peristaltické čerpadlo PCD 81E** [24]

pH sonda

V zařízení jsou použity čtyři pH sondy výrobce Hamilton s typovým označením 238401[1]. Jedná se o standardní sondu o délce 120 mm s konektorem s označením S7. Její pracovní teplota je 0–60°C a její měřící rozsah je pH 0–14.

Teplotní senzor

Teplotní senzor s označením PT1000[6] je běžně používaný typ teplotního senzoru s platinovým rezistorem s pozitivním teplotním koeficientem. Jeho kabel je vyroben z chemicky odolného silikonu a plášť je tvořen nerezovou ocelí, díky čemuž je vhodný i pro použití v chemicky agresivním prostředí. Jeho přesnost je ±0.15°C. Pro udržení vysoké přesnosti je výrobcem doporučováno provést kalibraci senzoru každé tři roky.

Topná deska

Jedná se o na míru vyrobený lakovaný plechový díl, který je uchycen do rámu zařízení na pohyblivých čepech, aby bylo možné měnit sklon desky. Tento díl je možné vidět v kontextu zařízení na obrázku 3.3. Na spodní straně desky se nachází elektrická topná tělesa sloužící k ohřívání desky, která poté slouží jako tepelný zářič pro ohřev na ni ležících vaků.

Topná tělesa jsou řízena pomocí SSR (solid-state relay), tedy relé na bázi polovodičů, které umožňují galvanicky oddělit řídící elektroniku od silové části sloužící k vytápění. Jejich opakovaným spínáním pomocí PWM (pulzně šířková modulace) je možné regulovat intenzitu vytápění dle potřeby. Pro přesné ovládání je však potřeba zohlednit tepelnou setrvačnost topné desky.

Vytápění

Jedná se pravděpodobně o generický typ topného tělesa s ventilátorem v plastovém provedení. Bohužel není v dokumentaci přístroje, ani na zařízení uvedeno žádné typové označení. Nezbývá tedy než usuzovat podle jištění pojistkou o hodnotě 230V/6.3A, že vytápění má příkon menší než 1500W.

Řídící jednotka

Jako řídící jednotka (obrázek 3.10) je v zařízení použita vývojová platforma Arduino Uno založená na mikrokontroléru ATmega328. Technické parametry jsou uvedeny v následující tabulce.[13]

Provozní napětí	7–12V
GPIO	14 (6 umožňuje PWM výstup)
Digitálních PWM GPIO	6
Analogových vstupů	6
Flash paměť	32 KB
SRAM	2 KB
EEPROM	1 KB
Pracovní frekvence	16 MHz

Tabulka 3.1: **Specifikace Arduino Uno** [13]

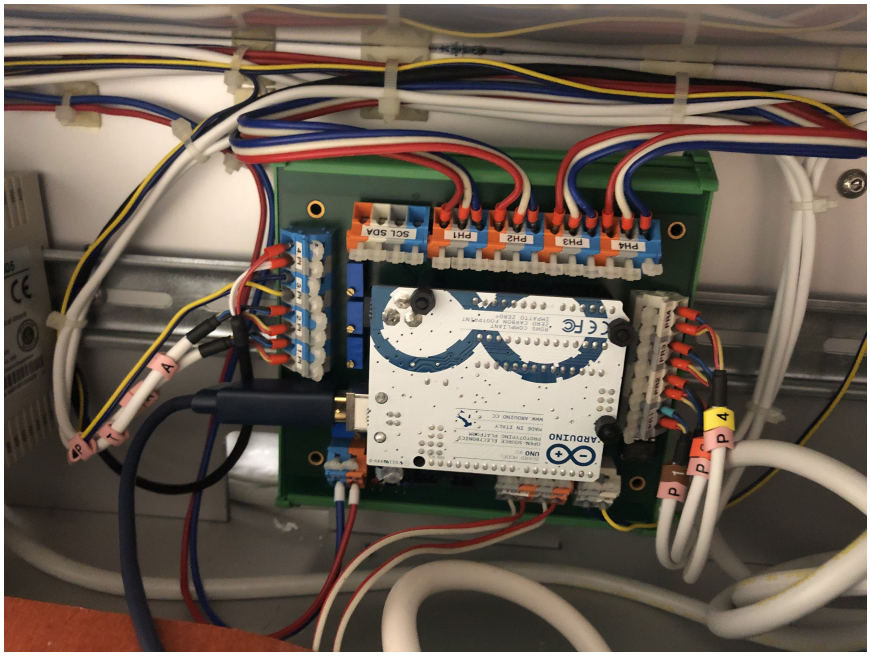
Řídící program této jednotky je napsán v jazyce C. Program může být do mikrokontroléru nahrán pomocí USB sériové linky.

Jednotka je skrz své konektory propojena s periferiím a podpůrnou řídící jednotkou pro ovládání míchacích zařízení. Pro měření teploty jsou využity analogově-digitální převodníky MPC-3424.

3.2.2 Komunikace a komunikační protokol

Komunikace mezi zařízením a uživatelským rozhraním probíhá po USB sběrnici ve formě sériové linky (UART). Příkazy jsou posílány ve formě textu ve formátu `/@X[P, ...]<CR>`. Popis komunikačního protokolu vychází z dokumentace přístroje GOLEM [17].

- / – Jedná se o prefix povelu který musí být vždy přítomný.
- @ – Adresa zařízení viz tabulka 3.2
- X – Určuje příkaz, který má být jednotkou vykonán. Dostupné příkazy jsou tyto:
  - D – Čtení dat ve formě čtyř čárkami oddělených hodnot ze senzorů teploty a čtyř z pH sond zakončené <CR>.



Obrázek 3.10: **Řídící jednotka GOLEM 2** – snímek ukazuje umístění a zapojení řídící jednoty spolu se zabudovaným Arduino Uno [24]

  - H[P] – Nastavení výkonu vytápění, kde hodnota P určuje požadovaný výkon v procentech.
  - h[P] – Nastavení výkonu topné desky, kde hodnota P určuje požadovaný výkon v procentech.
  - ? – Příkaz pro test komunikace, který vyvolá odpověď „GOLEM-2“ .
- [P, ...] – P Označuje parametr příkazu. Pokud příkaz přijímá více parametrů, jsou tyto odděleny čárkou. (Symboly [,] zde slouží pouze jako oddělovač a v příkazech se nevyskytují.)
- **<CR>** – Povinný sufix ukončující příkaz. Jedná o řídící symbol carriage return.

3.3 Uživatelské rozhraní

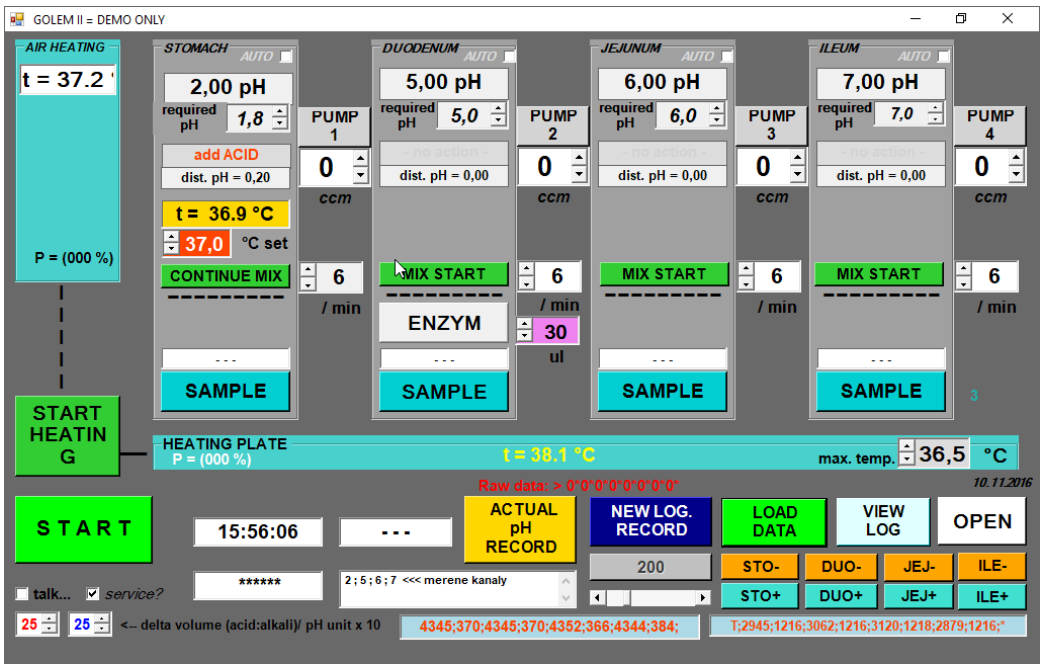
Uživatelské rozhraní využívá běžných prvků, jako jsou tlačítka, posuvníky a výběrové boxy. Z hlediska použitých technologií jde o standardní počítačovou aplikaci vytvořenou v prostředí jazyka Visual Basic.

Hlavní (a současně jediná) obrazovka (viz snímek obrazovky 3.11) obsahuje poměrně velké množství prvků, které umožňují řídit všechny dostupné parametry experimentu a také zobrazovat ladící informace a informační zprávy. Tyto zprávy mohou být doprovázeny i zvukovým upozorněním. Detailněji budou jednotlivé části rozhraní popsány v následujících částech této kapitoly.

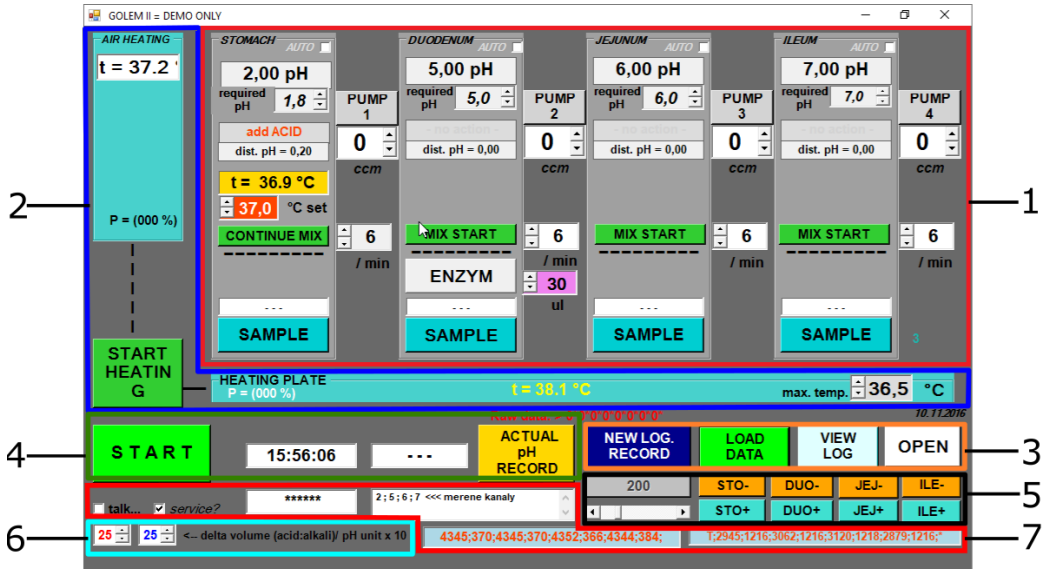
Pro usnadnění analýzy bylo rozhraní rozděleno na logicky oddělené celky, které budou analyzovány v podkapitolách. Rozdělení na části je vyznačeno na obrázku 3.12.

Adresa	Zařízení
1	Pístová pumpa kyseliny
2	Ventil kyseliny
3	Pístová pumpa zásady
4	Ventil zásady
5	Pístová pumpa enzymů
6	Modul míchání vaku 1
7	Modul míchání vaku 2
8	Modul míchání vaku 3
9	Modul míchání vaku 4
:	Modul Arduino

Tabulka 3.2: **Tabulka adres zařízení** – v této tabulce jsou uvedeny adresy jednotlivých zařízení.[17]



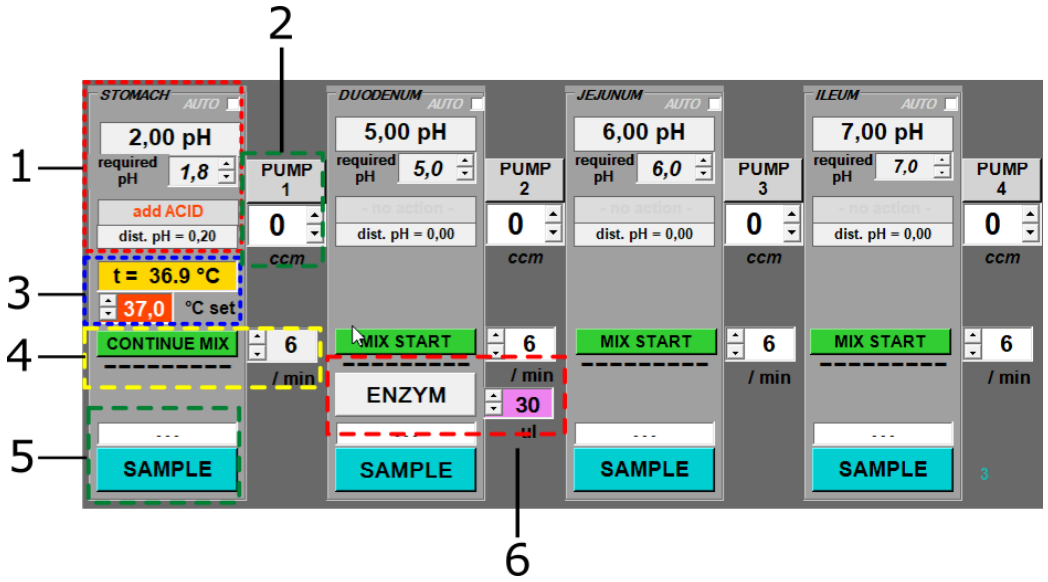
Obrázek 3.11: **Uživatelské rozhraní pro ovládání přístroje GOLEM** – snímek ukazuje původní verzi uživatelského rozhraní přístroje GOLEM, které bylo vytvořeno v jazyce Visual Basic a je tvořeno zejména standardními prvky jako jsou tlačítka, posuvníky a textová pole. Jak je ze snímku vidět, rozhraní má na poměrně malé ploše umístěno velké množství ovládacích prvků a místy není na první pohled zřejmý jejich účel



Obrázek 3.12: **Rozdělení uživatelského rozhraní na sekce** – (1) sekce rozhraní pro parametry zkoumané části trávicí soustavy, (2) rozhraní pro ovládání teploty, (3) rozhraní pro práci se soubory, (4) rozhraní pro řízení experimentu, (5) rozhraní pro manuální ovládání pH pump, (6) rozhraní pro nastavení pH roztoků v pumpách, (7) rozhraní pro ladění

3.3.1 Sekce rozhraní s parametry zkoumané části trávicí soustavy

V této části se nachází čtyři téměř identické bloky rozhraní, které představují jednotlivé části trávicího traktu. Bloky trávicí soustavy jsou ve stejném pořadí, v jakém jsou zapojeny při průběhu standardního experimentu, které bylo uvedeno na obrázku 3.2. Pro lepší orientaci jsou ovládací prvky rozděleny do skupin, které jsou vyznačeny na obrázku 3.13.



Obrázek 3.13: **Rozdělení ovládacích prvků do skupin** – (1) skupina prvků pro řízení pH, (2) ovládací prvky peristaltického čerpadla, (3) ovládání teploty, (4) ovládání míchacího zařízení, (5) ovládání vzorkování, (6) ovládání pístové pumpy enzymů

Řízení pH

Při bližším pohledu vidíme v horní části nadpis určující, o jakou část trávicího traktu se jedná. Bezprostředně pod nadpisem se v levé části nachází zaškrtávací políčko s názvem „AUTO“, který slouží k automatickému udržování stanovené hladiny pH. Další v pořadí je indikátor aktuální hodnoty pH ve formě textového pole, který je následován výběrovým boxem umožňujícím zadat požadovanou cílovou hodnotu pH. Posledními z prvků zaměřených na řízení pH je dvojice textových polí, které indikují doporučenou operaci pro korekci pH (přidat kyselinu / zásadu) a indikují odchylku od cílové hodnoty.

Ovládání peristaltického čerpadla

Napravo od předchozího bloku můžeme vidět tlačítko „Pump n“ kde n označuje číslo pumpy a pole pro určení množství tekutiny k přečerpání v ccm. Po zvolení hodnoty a stisknutí tlačítka „Pump“ dojde k okamžitému přečerpání zvoleného množství tekutiny do následujícího vaku, nebo do výlevky v případě posledního z vaků.

Ovládání teploty

Pod částí pro řízení pH následuje dvojice prvků věnovaná ovládání teploty. Prvním je textové pole zobrazující aktuální teplotu v prvním vaku, které je následováno polem pro určení cílové teploty. Tyto ovládací prvky se nachází pouze u první části (stomach), jelikož pouze tento vak má ve svém vstupním portu umístěný senzor teploty.

Ovládání míchacího zařízení

Další sekcí rozhraní je ovládání míchacího zařízení. V této části se nachází dva ovládací prvky, kde prvním je pole pro výběr rychlosti míchání a druhým je pak tlačítko pro spuštění, nebo zastavení míchání.

Ovládání odběru vzorku

V této části se nachází tlačítko pro potvrzení odebrání vzorku. Požadavek na odebrání vzorku v naplánovaném čase je signalizován blikáním tlačítka a zvukovým upozorněním. Po potvrzení odběru stisknutím tlačítka dojde k deaktivaci blikání.

Ovládání pístové pumpy enzymů

Ovládání je tvořeno polem pro zadání množství enzymů, které má být do vaku vstříknuto a tlačítkem pro provedení této operace. Tento ovládací prvek se nachází pouze u druhého vaku, jelikož je tento vak určen jako jediný zdroj enzymů v experimentální soustavě.

3.3.2 Sekce rozhraní pro ovládání teploty

Tato část rozhraní slouží k řízení teplotních parametrů experimentu. Umožňuje uživateli zvolit cílovou teplotu a aktivovat, či deaktivovat systém vytápění.

Rozhraní je rozděleno do dvou oddělených částí. Levý horní sloupec slouží k zobrazení informací o aktuálním procentuálním výkonu horkovzdušného ventilátoru a aktuální teplotě vzduchu uvnitř zařízení.

Druhou část rozhraní tvoří horizontální blok, který obsahuje informace o topné desce (aktuální teplota a výkon) a v levé části také textové pole pro zadání cílové teploty.

Posledním z obsažených prvků je tlačítko „Start heating“ sloužící k aktivaci a následně k deaktivaci topného systému.

3.3.3 Sekce rozhraní pro práci se soubory

Rozhraní pro práci se soubory je tvořeno čtyřmi tlačítky, které umožňují základní operace nezbytné pro načítání experimentu, nebo ukládání jeho výsledků.

Prvním z tlačítek zleva je „New log record“, které slouží k vynucení vytvoření nového záznamu v logovacím souboru. Následuje tlačítko „Load data“ které slouží k vyvolání dialogového okna pro výběr CSV souboru s parametry experimentu. Po kliknutí je zobrazeno standardní systémové okno pro výběr. Tlačítko „Wiev log“ slouží k otevření výchozího textového editoru systému se souborem obsahujícím záznamy o experimentu.

Posledním z tlačítek je „Open“ které slouží k odemknutí či uzamknutí tlačítka pro ukončení experimentu.

3.3.4 Sekce rozhraní pro řízení experimentu

Prvním z prvků v této části je tlačítko „Start“ sloužící ke spuštění experimentu a k jeho následnému zastavení. Za tímto účelem tlačítko po prvním kliknutí změní barvu na červenou a svůj text na „Stop session“. Použití tlačítka pro zastavení je možné až po kliknutí na tlačítko „Open“.

Druhým tlačítkem je „Actual pH record“ které slouží k vynucení nového měření hodnoty pH. Tlačítko slouží pro účely rychlého změření aktuální hodnoty například po manuálním přidání korekčního roztoku pH. Dále je zde dvojice textových polí v režimu pro čtení, které slouží jako hodiny a stopky měřící čas uběhlý od spuštění experimentu.

3.3.5 Sekce rozhraní pro manuální ovládání pH pump

Jak již jeho název napovídá, tlačítka v této oblasti slouží k aktivaci pístových pump pro vstříknutí kyseliny, nebo zásady do patřičného vaku. Tlačítka jsou pojmenována „STO+“ a „STO-“, kde první z nich slouží ke vstříknutí látek pro zvýšení pH ve vaku žaludku (stomach) a druhé pro snížení pH. Ostatní tlačítka jsou pojmenována analogicky dle dalších částí trávicího traktu.

Důležitým ovládacím prvkem je zde posuvník, který umožňuje výběr množství korekčního roztoku, které má být vstříknuto. Nad tímto posuvníkem se nachází textové pole, které zobrazuje aktuálně zvolenou hodnotu v mikrolitrech.

3.3.6 Sekce rozhraní pro nastavení pH roztoků v pumpách

V levé spodní části obrazovky programu se nachází dvojice polí pro výběr parametrů roztoků, které jsou k dispozici pro vstříknutí do vaků. Jejich hodnoty slouží pro výpočet množství korekčního roztoku, které má být vstříknuto při automatickém udržování pH.

3.3.7 Sekce rozhraní pro ladění

Posledním z prvků je zde zaškrtávací políčko s názvem „service?“, které po zaškrtnutí zobrazí textová pole obsahující ladící informace.

3.4 Typicky prováděné experimenty

Typicky prováděným experimentem je analýza rozpustnosti tablet, které jsou vhozeny do vaku představujícího žaludek. Poté následuje až několikahodinové promíchávání, postupné přečerpávání dle schématu na obrázku 3.2 a periodické odebírání vzorků a vyhodnocování. Výsledky experimentu jsou následně vyhodnoceny spektrální analýzou odebraných vzorků, která slouží k určení koncentrací účinných látek. Spojením výsledků spektrální analýzy, časů odebrání a zdrojů vzorků lze změřit vývoj koncentrace účinných látek v závislosti na čase. [26] Existují však i alternativní způsoby, jak tento systém použít.

3.4.1 Alternativní experimenty a zapojení přístroje

Jednou z možných alternativ je vícenásobné simulování částí trávicího traktu. V úvahu připadají například konfigurace 4x stomach, nebo 2x stomach a 2x duodenum. Tyto varianty experimentů mohou být použity například při experimentech s rychlostí míchání, popřípadě vlivu systému přepážek ve vacích.

Tímto způsobem pak mohou být například vyhodnocovány parametry až čtyř vaků při stejných podmínkách, nebo také testováno rozpouštění tablet ve čtyřech identických vacích s různou intenzitou míchání.

Kapitola 4

Uživatelská rozhraní na bázi webových technologií

Tato kapitola má za cíl poskytnout nezbytné informace o moderních webových technologiích, uživatelských rozhraních a jejich návrhu. Informace obsažené v této kapitole jsou použity v dalších částech této práce.

První podkapitola slouží jako teoretický základ shrnující základní principy, návrhové vzory a doporučené postupy pro vývoj moderních uživatelských rozhraní. Druhá podkapitola se pak věnuje zejména frameworku Vue, který byl použit pro realizaci této práce.

4.1 Návrh uživatelských rozhraní

Grafická uživatelská rozhraní (GUI) jsou v dnešní době jedním z nejčastěji používaných typů uživatelských rozhraní. Díky jejich rozšířenosti a oblíbenosti existuje velké množství nástrojů, návrhových vzorů a postupů, jak tato rozhraní vytvářet, tak, aby byla zajištěna jejich srozumitelnost a ergonomie. Cílem této podkapitoly je poskytnout stručný teoretický základ pro jejich realizaci. Tato podkapitola vychází z knih [19] a [21].

4.1.1 Proces návrhu a vývoje

Typicky proces návrhu nového uživatelského rozhraní začíná analýzou cílové skupiny a funkčních požadavků. Výsledky jsou poté formalizovány do podoby textových dokumentů a diagramů, které slouží jako základ pro další činnosti.

Po této fázi následuje návrh uživatelského rozhraní s použitím různých technik a nástrojů, které umožňují vytvoření prototypu. Prototypy mohou být buď statické, nebo interaktivní. Příkladem mohou být interaktivní mockupy, které simulují chování některých částí uživatelského rozhraní. Obvykle je interaktivní chování spojeno s ovládacími prvky, které slouží k vyvolání modálních oken, nebo navigaci v aplikaci.

Časté chyby návrhu

Jednou z častých chyb při návrhu nového uživatelského rozhraní je vytváření struktury rozhraní, která odpovídá implementačním mechanismům. Důsledkem tohoto přístupu jsou obvykle zbytečně složitá a málo ergonomická uživatelská rozhraní. Dalším typickým problémem je nesprávné pochopení požadavků, nebo jejich nepřesná specifikace, která vede k nekompletnímu nebo nesprávnému designu dané části rozhraní.

Doporučený postup

Pro vytvoření kvalitního a ergonomického uživatelského rozhraní je nezbytné analyzovat doménu úloh, které budou uživatelé vykonávat a navrhnout strukturu rozhraní tak, aby co nejlépe umožňovala jejich realizaci. Vhodnost a srozumitelnost jednotlivých částí je vhodné pravidelně ověřovat pomocí uživatelských testů a vyhodnocování jejich zpětné vazby. Rovněž není vhodné při návrhu používat nestandardní ovládací prvky, které by mohly být pro uživatele nesrozumitelné. Dále je také vhodné zachovat konzistenci významů barev a pozic ovládacích prvků napříč celou aplikací, aby nedocházelo k matení uživatele.

4.1.2 Návrhové vzory

Aby při návrhu nových uživatelských rozhraní nedocházelo k opakovanému vytváření standardních prvků a řešení běžných úkolů uživatelských rozhraní, bylo vytvořena rozsáhlá škála návrhových vzorů, které tyto typické prvky a činnosti realizují.

Návrhové vzory pro grafická uživatelská rozhraní můžeme rozdělit do následujících kategorií:

- Formuláře – textová pole, kalendáře, textové editory, ...
- Navigace – tlačítka, záložky, modální okna, mřížky, gesta, ...
- Data – tabulky, seznamy, galerie, formátovaný text, ...
- Společenské – hodnocení, komentáře, chat, ...
- Ostatní – produktové stránky, nákupní košíky, ...

Použití standardních návrhových vzorů umožňuje rychle a efektivně vytvářet uživatelská rozhraní, která jsou pro uživatele srozumitelná a neobsahují atypické ovládací prvky, které by mohly uživatelům působit problémy. [19]

4.2 Technologie webových aplikací

V současné době existuje velmi široká škála frameworků, knihoven a jazyků, které umožňují vytváření grafických uživatelských rozhraní. Mezi nejznámější z nich patří Swing, Qt a WPF, které slouží zejména pro vytváření uživatelských rozhraní klasických aplikací.

Alternativu poskytují webové technologie, založené na jazycích JavaScript, CSS a HTML, které umožňují vytváření multiplatformnách aplikací běžících ve webovém prohlížeči. Mezi nejznámější z nich patří frameworky Twitter Bootstrap a Material Design, které slouží jako designový základ a dále frameworky React, Angular a Vue, které poskytují realizační základ pro vytváření interaktivních uživatelských rozhraní.

V této práci byla použita kombinace frameworku Material Design implementovaného v knihovně komponent Vuetify a framework Vue. V následujících podkapitolách jsou uvedeny technické detaily a základy jejich použití. Detaily o procesu výběru technologií pro realizaci této práce naleznete v kapitole 5. Text této podkapitoly je založen zejména na webových dokumentacích frameworků Vue [15] a Vuetify [25] spolu s knihou o frameworku Vue [22].

4.2.1 Framework Vue.js

Jedná se o kompilovaný JavaScriptový framework pro vytváření uživatelských rozhraní a jednostránkových aplikací. Vue.js obsahuje oproti jiných frameworků pouze malé množství funkcí, které však mohou být velmi jednoduše rozšířeny použitím knihoven a modulů. Díky tomuto přístupu je možné vytvářet aplikace s minimem nadbytečných funkcí a závislostí, které by zvyšovaly jejich velikost a složitost.

Hlavním přínosem kompilovaných JavaScriptových frameworků je zejména multiplatformnost výsledného kódu, odstranění redundantních dat, komprese, jednodušší vývoj a údržba projektů díky možnosti generování HTML s použitím deklarativního jazyka.

Základní princip

Vue poskytuje nadstavbu nad klasickým jazykem HTML, která umožňuje využít sílu jazyka JavaScript pro automatické generování prvků uživatelského rozhraní a reakce na události. Typickým příkladem může být například generování tabulek, které by v klasickém HTML znamenalo velmi rozsáhlý statický kód, nebo použití jiného jazyka jako je například PHP pro jeho generování na straně serveru. V případě Vue.js odpadá nutnost generování HTML na straně serveru a tento úkol se naopak přesouvá na stranu klienta. Realizace tohoto úkolu je snadná, jelikož framework poskytuje širokou sadu direktiv a speciálních příkazů k realizaci těchto úkolů.

Architektura Vnitřní architektura Vue.js je založena na návrhovém vzoru MVVM, jehož grafickou podobu můžete vidět na obrázku 4.1. Vue realizuje zejména část ViewModel, která slouží k řízení vnitřního stavu a obousměrnému provázání vrstev View a Model. Vue následně reaguje na události přicházející z vrstvy View a reaguje buď změnou vnitřního stavu, nebo modifikací dat ve vrstvě Model. Změny ve vrstvě Model jsou naopak pomocí direktiv generujících jednotlivé části uživatelského rozhraní propagovány zpět do vrstvy View.

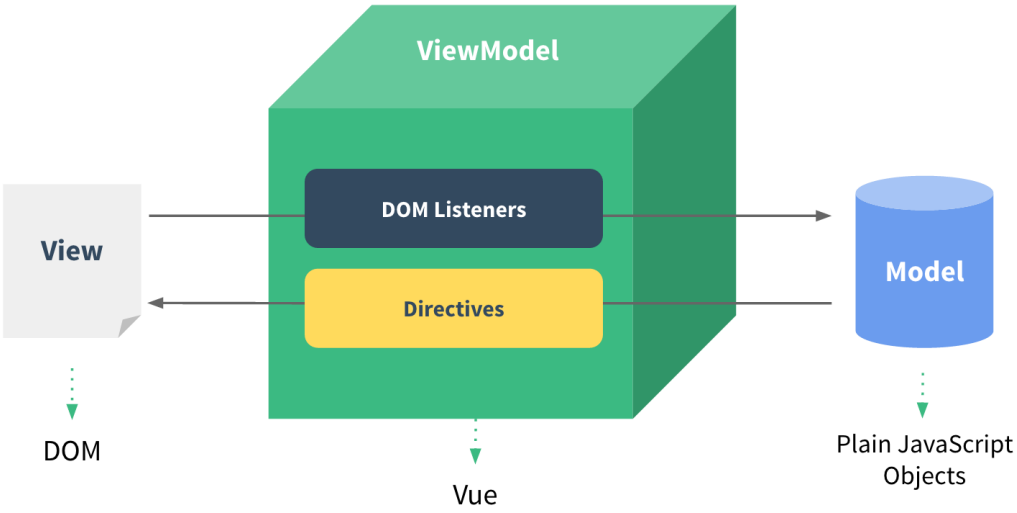
Komponenty

Komponenty jsou znovupoužitelné části, které deklarují novou HTML značku (tag), která může být v rámci aplikace použita. Obsahují typicky tři sekce. První je sekce `template`, tedy předloha, která obsahuje HTML značky sloužící ke generování prvků uživatelského rozhraní. Tato sekce je následovaná sekcí kódu v jazyce JavaScript a jednou, nebo dvěma sekcemi kaskádových stylů. Jedna ze sekcí stylů je obvykle vztažena k lokálním prvkům této komponenty (indikováno atributem `scoped`) a druhá ke globálním kaskádovým stylům.

Komponenty jsou realizovány buď jako oddělené soubory, nebo jako JavaScriptové objekty uvnitř jiných komponent, či pohledů. Komponenty nacházející se v oddělených souborech je potřeba před použitím importovat. Poté je možné komponentu použít jako vlastní HTML tag odpovídající názvu komponenty.

Direktivy

Pod pojmem direktivy jsou v tomto frameworku obsaženy různé specializované atributy, které mohou být použity při zápisu HTML značek. Direktivy jsou v tomto frameworku používány velmi často a můžete je jednoduše rozeznat od jiných atributů díky jejích před-



Obrázek 4.1: **Koncept návrhového vzoru MVVM** obrázek ukazuje obousměrné propojení vrstev View (pohled) a Model (data) které je realizováno skrz vrstvu ViewModel. Tato vrstva v reakci na události DOM (Document Object Model) modifikuje data a na základě změn dat provádí vykreslování pohledu pomocí direktiv. (Převzato z: https://012.vuejs.org/guide/)

poně `v-`. Direktivy je možné rozdělit do skupin podle jejich účelu na smyčky, podmínky, manipulace, datové direktivy a filtry.

Mezi nejčastěji používané patří například direktivy pro práci s daty, jako je `v-model`. Tato direktiva slouží k obousměrnému předávání dat mezi komponentami. Další často používanou direktivou je `v-for` která realizuje cyklus nad zvoleným polem. Díky této direktivě je možné pohodlně generovat například tabulky, či jiné struktury složené z prvků stejného typu.

Properties

Direktivy pro předávání dat ke svému fungovaní používají properties (rekvizity), což jsou atributy sloužící k předávání konkrétních dat mezi rodičem a cílovou komponentou. Účelem těchto nových atributů je vytvořit jednoznačné přiřazení mezi daty na straně rodiče a komponenty. Toto spojení je pouze jednosměrné, a to ve směru od rodiče k potomkovi. Pro zpětnou komunikaci je nezbytné zaslat aktualizovaná data rodiči ve formě zprávy.

Na straně komponenty jsou tyto rekvizity deklarovány v JavaScriptové části ve formě objektů. Objekty obvykle obsahují deklaraci přijímaného datového typu, výchozí hodnotu a příznak, zda je hodnota vyžadována. V případě, že tyto položky nejsou vyplněny není daná kontrola provedena a v případě výchozí hodnoty je použito `undefined`.

Události

Pro reakce na různé typy událostí jsou ve Vue používány speciální atributy začínající symbolem `@`, příkladem mohou být běžně známé události jako `@click`, nebo `@mouseover`. Vue

umožňuje i použití vlastních názvů událostí a jejich vyvolání pomocí metody `$emit()`. Reakcí na událost je typicky volání metody, což je zapsáno jako `@JménoUdálosti="metoda"`. Alternativně může být do uvozovek umístěn libovolný JavaScriptový kód, například přiřazení hodnoty proměnné.

Sloty

Slot je označení pro část komponenty, která může být nahrazena novým obsahem z rodičovské komponenty. Sloty mohou být pojmenovány pro jejich rozlišení a jejich využití z mateřské komponenty je možné použitím speciální direktivy `v-slot:jméno="data"`.

Typickým použitím je například vložení jednotky na konec textového pole u komponenty `v-text-field`, které může být realizováno pomocí slotu `append`. Jiné využití může být například vložení vlastního kódu do všech buněk určitého sloupce tabulky. Obsah těchto buněk může být nahrazen libovolným HTML kódem, jako například tlačítky, nebo ikonami.

4.2.2 Knihovna komponent Vuetify

Vuetify je knihovna komponent pro Vue.js, která přidává rozsáhlou škálu komponent, umožňující rychlý vývoj webových rozhraní tvořených standardními komponentami Material Design. Díky použití standardizovaných prvků a technologií je možné vytvořit unifikované rozhraní pro webové prohlížeče i mobilní aplikace.[25]

Material Design

Je designerský jazyk vyvinutý společností Google, která jej používá ve většině mobilních a webových aplikací, jako jsou YouTube, Google Drive a Google Maps. Je založen na systému karet, mřížek, responzivních animací, přechodů a mnoha dalších vizuálních prvků.[18]

V tomto projektu je požívána také knihovna Material Design Icons, která poskytuje rozsáhlou open-source sbírku ikon vytvářených komunitou. Ikony jsou dostupné jak ve formátu SVG, tak ve formě ikonového písma (font).

Kapitola 5

Inovace uživatelského rozhraní přístroje GOLEM 2

Tato kapitola je zaměřena na proces návrhu nového uživatelského rozhraní, které by výzkumníkům umožnilo jednoduše a efektivně řídit průběh experimentů. Jako základ pro analýzu požadavků byly použity výsledky předchozí kapitoly, dokumentace a konzultace s členy výzkumné skupiny. Během konzultací byly členy výzkumné skupiny demonstrovány typické činnosti a určena slabá i silná místa původního rozhraní. Zpracováním výsledků těchto konzultací vznikly neformální specifikace požadavků, které byly následně upřesněny a formalizovány.

Dalším krokem bylo vytvoření diagramů potřebných pro pochopení systému jako celku a také interakcí mezi jednotlivými částmi. Jako první byl vytvořen use-case diagram (diagram případů užití) a diagram aktivit. Ty posloužily pro rozdělení systému na logické celky a posléze i na oddělené obrazovky.

Návrh nového rozhraní byl realizován formou interaktivního mockupu vytvořeného pomocí online platformy Draw.io. V tomto interaktivním prototypu uživatelé zkoušeli vykonat typicky prováděné úkony a svými poznámkami umožnili během několika iterací vytvořit takový design, který obsahoval všechny požadované prvky a byl uživatelsky přívětivý.

Významným faktorem při návrhu nového uživatelského rozhraní bylo také to, že v rámci souběžně probíhající diplomové práce Bc. Zbyška Vody má být pro zařízeni GOLEM vytvořena nová řídící jednotka[24]. Díky tomu je možné přidat nové funkce a prvky, které původní řídící jednotka neumožňovala.

5.1 Analýza požadavků zadavatele

Pro určení požadavků zadavatele bylo nezbytné nejen analyzovat původní systém, ale také důkladně pochopit typicky prováděné činnosti a význam jednotlivých kroků z hlediska farmakokinetiky. Za tímto účelem se projevily jako nejužitečnější osobní konzultace s výzkumnou skupinou, při kterých byly ve formě poznámek určeny nejen typické činnosti a požadavky uživatelů na rozhraní, ale také navrženy nové rozšíření a nápady na celkové vylepšení systému. Při konzultacích byly rovněž konzultovány technologie, které by mohly být použity pro vytvoření nového uživatelského rozhraní.

5.1.1 Identifikace aktérů

Jako první krok při analýze logicky vyplynulo určení aktérů a jejích rolí. Původní systém nevyužíval žádné uživatelské účty, ani role. Ovládání přístroje tedy bylo přístupné každému uživateli, který měl přístup do laboratoře.

Z tohoto zjištění vyplynul návrh na vytvoření dvou uživatelských skupin s rozdílnou úrovní oprávnění, tedy uživatelé a administrátoři. Tento krok byl vhodný zejména pro omezení práv běžných uživatelů (studentů) tak, aby jim nebylo umožněno měnit kalibraci přístroje a zasahovat do jiných pokročilých nastavení, které by mohly narušit chod přístroje. Dalšími aktéry jsou „Systém“ a „Čas“. Oba aktéři mohou vyvolat upozornění pro uživatele v případě potřeby. Aktér „Systém“ může navíc změnit stav experimentu (uspěl, neuspěl) v případě selhání hardware, nebo nedodržení tolerancí.

5.1.2 Funkční požadavky

Prvním krokem pro návrh rozhraní je určit nezbytné funkce, které mají být uživateli poskytnuty. Z tohoto důvodu je žádoucí věnovat značnou péči analýze funkčních požadavků a intenzivně je se zúčastněnými stranami konzultovat. Výsledkem tohoto postupu by měl být seznam jasně definovaných úkonů, které musí aplikace umožnit vykonat. Tato sada požadavků byla spolu s výzkumnou skupinou stanovena jako funkční minimum, které musí aplikace pro akceptaci umožnit realizovat.

Přihlášeni uživatele

Nezbytným krokem pro vytváření nové uživatelské relace je přihlášení uživatele do systému. Jako nejvhodnější se zde jeví standardní rozhraní ve formě textových polí.

Rozhraní musí umožnit zadání uživatelského jména, hesla a obsahovat tlačítko pro odeslání přihlašovacích údajů k ověření. Textové pole pro heslo nesmí zobrazovat skutečně zadané symboly, aby nedošlo k vyzrazení hesla během zadávání. Po úspěšné autorizaci bude uživatel přesměrován na hlavní menu aplikace. V opačném případě bude upozorněn na neplatné přihlášení.

Zobrazení hlavní nabídky

Po přihlášení bude uživatel přesměrován na stránku aplikace, která bude sloužit jako hlavní menu. Zde bude mít uživatel dle úrovně svého oprávnění zobrazeny položky pro přístup k dalším částem systému. Běžný uživatel bude mít přístup pouze k experimentům, jejich vytváření a provádění. Administrátor bude mít navíc i přístup k nastavení zařízení.

Otevření existujícího experimentu

Pravděpodobně nejčastěji používanou akcí zde bude použití tlačítka pro zobrazení seznamu existujících experimentů a následný výběr některého experimentu ze seznamu. Po výběru experimentu bude uživatel přesměrován na stránku obsahující detaily zvoleného experimentu.

Vytvoření nového experimentu

Alternativou k otevření již existujícího experimentu je vytvoření nového experimentu. V původním rozhraní je projekt vytvářen pomocí tabulkového procesoru Microsoft Office Excel

a jako předloha jsou použity již existující projekty, ve kterých jsou pouze upraveny či přidány požadované řádky. Výstup je importován ve formátu CSV.

Nové uživatelské rozhraní by mělo uživateli umožnit pohodlně vytvářet nové experimenty a definovat všechny potřebné parametry bez nutnosti použití aplikací třetích stran. Avšak je vhodné zachovat tuto možnost importu experimentu. Ta bude využita v případě vytváření velmi rozsáhlých experimentů s mnoha událostmi.

Ze zadání rovněž vyplývá nutnost vytvořit systém pro vizuální specifikaci automatického řízení průběhu experimentu a systému, který bude toto řízení vykonávat. Systém má dle požadavků výzkumné skupiny umožnit automatické hlídání dodržení parametrů experimentu a upozornit uživatele na nevyhovující podmínky. Další funkce nejsou požadovány.

Úprava experimentu

Pro umožnění správy experimentů je nezbytné vytvořit rozhraní, které umožní editaci již existujících experimentů. Avšak je potřeba mít na paměti, že editací, nebo odstraněním experimentu nesmí být ovlivněna historie běhů experimentů, jejichž výsledky by takto mohly být znehodnoceny. Pro zabránění těmto nežádoucím vlivům se jeví jako vhodné řešení ukládat aktuální nastavení experimentu v okamžiku jeho provedení do datové struktury běhu.

Spuštění experimentu

Aby bylo možné systém reálně využít, musí rozhraní nabídnout uživateli možnost spuštění nového běhu experimentu. Při spuštění dojde k vytvoření nového běhu experimentu s aktuální kopií nastavení a přesměrování uživatele na obrazovku pro sledování a řízení průběhu experimentu.

Sledování a řízení průběhu experimentu

Tato obrazovka musí uživateli poskytnout ergonomické rozhraní, jak pro sledování průběhu experimentu, tak pro jeho manuální řízení v případě potřeby. Z této obrazovky by měly být dostupné všechny ovládací prvky pro řízení experimentu, které byly obsaženy v původním uživatelském rozhraní.

Ukončení experimentu

Ukončení experimentu může být provedeno buď předčasně uživatelem, nebo automaticky systémem při vypršení času běhu experimentu. Při předčasném ukončení se experiment považuje za neúspěšný. Toto chování je důsledkem nevykonání všech kroků experimentu, což vede k nekompletním nebo chybným výsledkům.

Správa systému

Alternativní volbou v hlavní nabídce, která bude dostupná pouze administrátorům, je použití tlačítka pro přístup k obrazovce pro správu nastavení přístroje. Z této obrazovky by měly být uživateli přístupné všechny následující funkce. Uživateli bez patřičného oprávnění bude při pokusu o přístup na tuto stránku i následující podstránky v tomto kroku zabráněno. Stejným způsobem musí reagovat i serverová část v případě pokusu o manuální odeslání požadavku.

Správa uživatelů

Pro umožnění správy uživatelů musí být vytvořeno rozhraní, které umožní všechny běžné operace, jako je přidání, editace a odstranění uživatelů. Jemné řízení uživatelských oprávnění není požadováno a bude tedy k řízení přístupu použita pouze uživatelská skupina.

Kalibrace přístroje

Jelikož přístroj obsahuje citlivé senzory pH, které je nezbytné pravidelně kalibrovat, musí nové rozhraní obsahovat obrazovku, která umožní administrátorům jejich jednoduchou vícebodovou kalibraci. Z požadavků zadavatele vyplynulo, že rozhraní by mělo umožnit jak manuální kalibraci, tak automatickou kalibraci s použitím průvodce, který operátora procesem kalibrace provede.

Detekce a oprava chyb přístroje

Aby bylo možné detekovat a následně opravit chyby přístroje, je nezbytné vytvořit rozhraní, které přehledně informuje administrátora o detekovaných chybách. Rozhraní by dále mělo umožnit zobrazit poslední prováděné experimenty a zda nedošlo k chybám při jejich provádění. Další z funkcí, které by mělo rozhraní umožnit, je obnovení stavu systému ze zálohy v případě výpadku napájení a poškození interní databáze.

Nastavení sítě a aktuálního času

Systém musí umožnit administrátorovi nastavit klíčové parametry pro provoz zařízení. Těmito parametry jsou zejména aktuální čas a nastavení Wifi sítě, která má být v nové verzi přístroje GOLEM použita k propojení uživatelského rozhraní se zařízením. V rozhraní musí být možné nastavit klíčové parametry jako je název Wifi sítě (SSID), heslo sítě a mód (klient, přístupový bod).

5.1.3 Výkonnostní požadavky

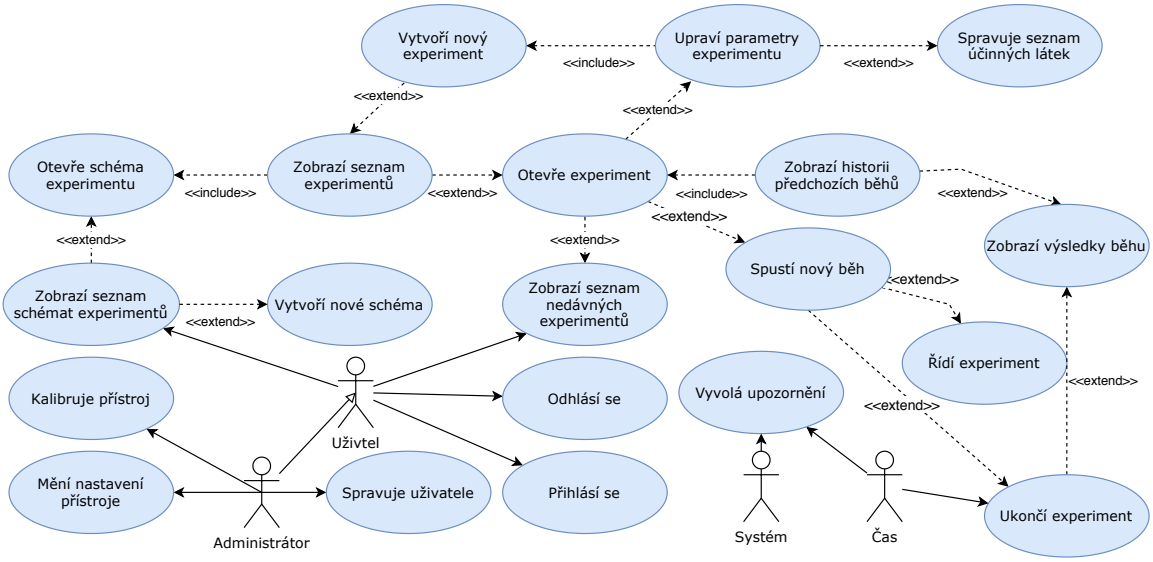
Výsledná aplikace musí dokázat v reálném čase monitorovat průběh experimentů a umožnit jeho řízení. Vzhledem k tomu, že zařízení neumožňuje průběh více experimentů současně, je dostatečné, když systém umožní pouze jednu aktivní uživatelskou relaci.

5.2 Vytvoření modelu systému

Před započetím návrhu nového uživatelského rozhraní byla vytvořena sada diagramů, které sloužily pro vytvoření ucelené představy o funkci systému a jako základ pro následující fázi návrhu. Tyto diagramy byly následně v jednotlivých iteracích upravovány a rozšířeny, až bylo dosaženo finálních diagramů, které jsou uvedeny v této podkapitole.

5.2.1 Případy užití

Prvním vytvořeným byl diagram případů užití, který má za úkol zejména poskytnout ucelený pohled na funkční požadavky a jejich vzájemné vazby. Tento diagram byl následně využit jako základ pro vytvoření diagramu aktivit, rozdělení a návrh pohledů aplikace. Výsledný diagram je zobrazen na obrázku 5.1.



Obrázek 5.1: **Diagram případů užití** – účelem tohoto diagramu je poskytnout představu o jednotlivých případech užití aplikace a jejich vzájemných vazbách na aktéry Administrátor, Uživatel, Čas a Sytém i vazby mezi jednotlivými případy užití.

5.2.2 Diagram aktivit

Dále byl vytvořen diagram aktivit (obrázek 5.2), jehož účelem je zachytit aktivity, které může uživatel v systému vykonávat a jejich vzájemnou návaznost. Tento diagram slouží zejména k dalšímu upřesnění představy o funkci systému a také jako výchozí bod pro návrh struktury směrovaní mezi jednotlivými pohledy v aplikaci.

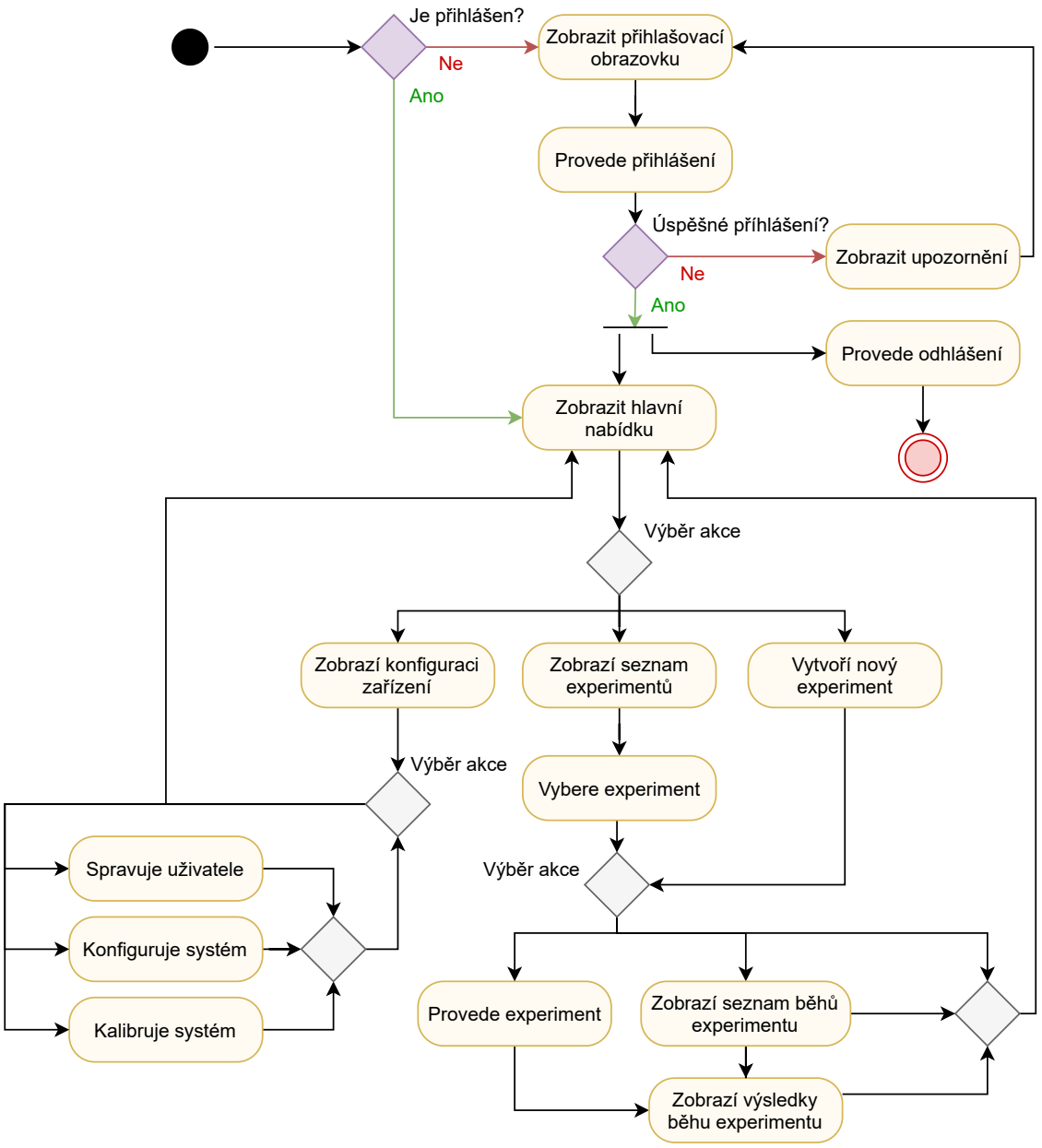
5.3 Návrh uživatelského rozhraní pomocí interaktivních mockupů

Pro jednoduchost návrhu nového uživatelského rozhraní a vyloučení nedorozumění v koncepci nového rozhraní bylo zvoleno použití interaktivních mockupů, které mají za úkol věrně napodobit podobu výsledné aplikace. Jejich další výhodou je to, že poskytnou uživateli možnost ovládání některých prvků rozhraní pro vyvolání dialogových oken a procházení jednotlivými obrazovkami aplikace, což lze výhodně využít při testování s cílovými uživateli. Výsledná aplikace by pak měla být pro uživatele srozumitelná a obsahovat všechny požadované prvky.

5.3.1 Draw.io

Pro vytváření prototypu uživatelského rozhraní bylo zvoleno Draw.io[1], což je online platforma pro vytváření mockupů, vektorové grafiky, diagramů a tabulek. Tato platforma byla zvolena zejména proto, že umožňuje zdarma vytvořit interaktivní online mockupy pomocí použití více vrstev a řízení jejich viditelnosti kliknutím. Díky této vlastnosti lze jednoduše vytvářet návrhy obrazovek, formulářů a dialogových oken v podobě, která je blízká finální

[1] https://draw.io/



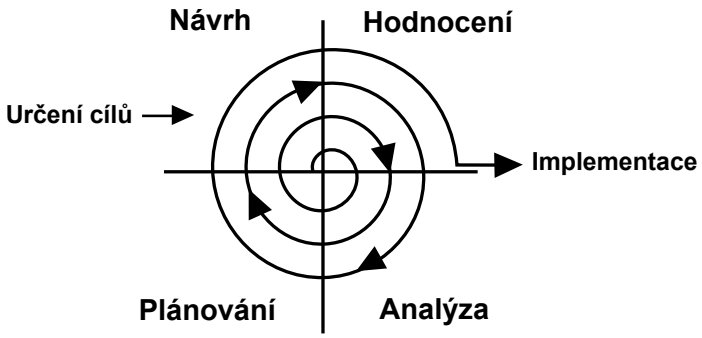
Obrázek 5.2: **Diagram aktivit** – účelem tohoto diagramu je znázornit akce, které může uživatel v dané části rozhraní vykonat a jejich vzájemnou návaznost.

implementaci. a dialogových oken v podobě, která je blízká finální implementaci. a dialogových oken v podobě, která je blízká finální implementaci.

5.3.2 Proces návrhu

Proces návrhu uživatelského rozhraní (viz obrázek 5.3) byl rozdělen do iterací, kde na začátku každé iterace byl upraven (v případě první iterace vytvořen) návrh uživatelského rozhraní ve formě mockupu. Následně byl tento návrh předám členům výzkumné skupiny k testování. Uživatelé jej následně procházeli a studovali jednotlivé prvky rozhraní, jejich rozmístění a účel. Po této fázi následovala diskuse a vyhodnocení zpětné vazby uživatelů. Výsledky byly shrnuty ve formě poznámek, které byly následně analyzovány. Výsledky analýzy byly použity k návrhu změn a následně zapracovány v další iteraci do aktualizované verze mockupu. Postupně se takto mockup rozšiřoval o nové funkce a obrazovky, až bylo dosaženo výsledného návrhu rozhraní, které výzkumná skupina odsouhlasila jako kompletní.

V této kapitole jsou kvůli velkým rozměrům a špatné čitelnosti uvedeny pouze některé snímky z výsledného mockupu, které zde slouží pro dokreslení idey těchto obrazovek. Snímky implementovaných obrazovek ve vysoké kvalitě jsou umístěny v příloze C.



Obrázek 5.3: **Spirálový model procesu návrhu uživatelského rozhraní** (Původní verze obrázku převzata z https://bit.ly/302Kfi5)

Přihlašovací obrazovka

První obrazovkou, která by měla být uživateli po vstupu do systému zobrazena, je přihlášení. Nepřihlášený uživatel nesmí mít přístup k žádné z částí systému, aby nedošlo k narušení probíhajícího experimentu, nebo nastavení přístroje.

Pokud se nepřihlášený uživatel pokusí přistoupit na jinou obrazovku, bude přesměrován právě na tuto. V případě neúspěšného přihlášení bude uživatel upozorněn červeným textem pod tlačítkem přihlášení. V případě, že uživatel již je přihlášen, bude automaticky přesměrován na následující hlavní obrazovku aplikace.

Hlavní obrazovka

Další obrazovkou v pořadí je hlavní stránka (viz obrázek 5.4), která funguje jako hlavní menu aplikace. Uživatel má zde možnost zvolit si požadovanou činnost, po jejímž výběru je přesměrován na další stánku.

Dostupné položky hlavní nabídky mají být zobrazeny na základě uživatelské skupiny, do které aktuálně přihlášený uživatel spadá. V případě administrátora je zobrazena položka pro přístup k nastavení zařízení, v případě běžného uživatele je tato položka skryta.

Obrázek 5.4: **Hlavní obrazovka s nabídkou**

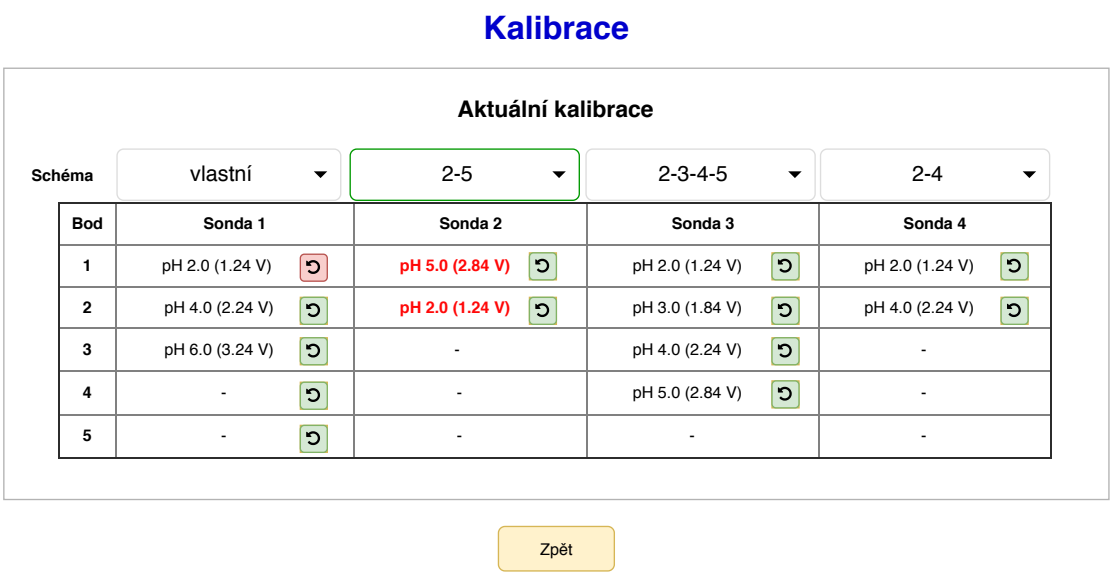
Nastavení

Pokud se administrátor rozhodl na předchozí obrazovce zvolit možnost „Nastavení“, je automaticky přesměrován na tuto stránku, která funguje jako další úroveň nabídky pro výběr konkrétní stránky nastavení. Návrh této stránky můžete vidět na obrázku 5.5.

Obrázek 5.5: **Obrazovka pro správu nastavení systému**

Proces návrhu této stránky byl poměrně intenzivní. V první verzi nebyla stránka tvořena pouze nabídkou, ale obsahovala prvky ze všech pod-obrazovek. Postupným přidáváním dalších funkcí však došlo k potřebě rozdělit tuto stránku na několik podstránek, které logicky oddělí dané operace a vytvoří tak přehlednější rozhraní.

Těmito novými stránkami byla správa uživatelů. Nastavení systému, obrazovka pro diagnostiku a obrazovka pro kalibraci přístroje, která je naznačena obrázkem 5.6.



Obrázek 5.6: **Obrazovka pro kalibraci přístroje**

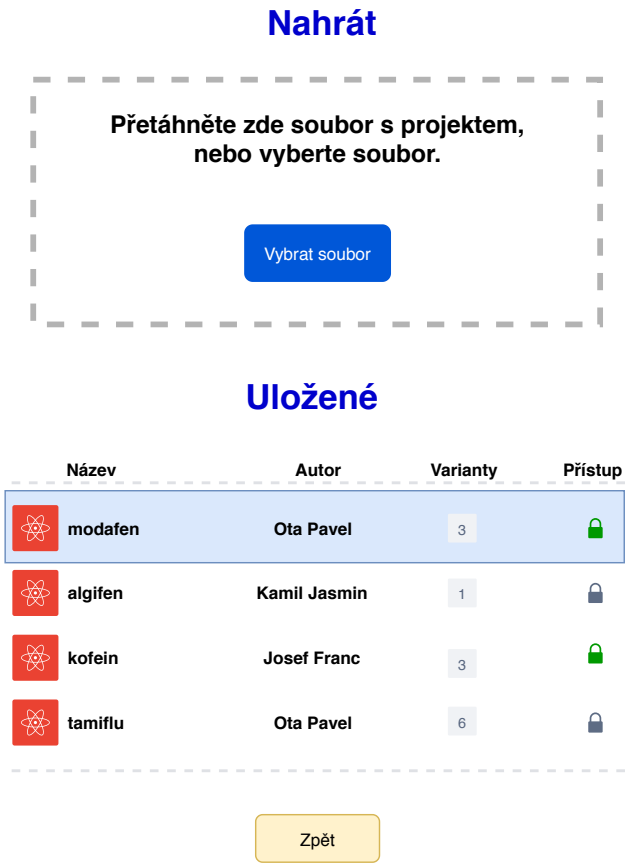
Nový experiment

Pokud uživatel na hlavní obrazovce použije tlačítko „Nový experiment“, je mu zobrazena obrazovka, která slouží k vytvoření nového experimentu a nastavení jeho základních údajů a parametrů. Uživatel má na této stránce možnost buď vyplnit všechny požadované parametry a vytvořit nový experiment použitím tlačítka pokračovat, které provede přesměrování na obrazovku pro vytvoření plánu vykonání experimentu. Alternativně může kdykoli opustit vytváření experimentu použitím tlačítka zpět. V případě, že uživatel vyplnil některé z polí na této stránce, bude při použití tlačítka zpět vyznán k potvrzení, že skutečně chce opustit tuto obrazovku. V případě kladné odpovědi je přesměrován zpět na hlavní obrazovku, jinak nebude přesměrování provedeno.

Proces návrhu této obrazovky byl proveden v několika iteracích, kdy byly postupně přidávány, přesouvány a měněny prvky rozhraní tak, aby poskytly úplné a přehledné rozhraní pro zadání základních parametrů nového experimentu. První verze obsahovala pouze textová pole pro zadání základních informací jako je jméno projektu, autor, obsahy vaků a podobně. Po konzultacích byly přidány oddělená textová pole pro nastavení pH jednotlivých vaků a následně i výběrové pole pro účinnou látku. S tímto polem bylo přidáno i modální okno, které slouží ke správě databáze účinných látek. Poslední modifikací bylo doplnění zaškrtávacího políčka pro určení přístupnosti experimentu (veřejný projekt je dostupný všem uživatelům, neveřejný pouze autorovi).

Existující experiment

Alternativní volbou k vytvoření nového experimentu je otevření již existujícího. Tato stránka (viz obrázek 5.7) má umožnit jak otevření projektu uloženého v rámci zařízení, tak nahrání projektu ze souboru. Nahrání projektu může být realizováno buď přetažením souboru s projektem do vyznačené oblasti, nebo použitím tlačítka pro výběr souboru, které vyvolá standardní dialogové okno prohlížeče.



Obrázek 5.7: **Obrazovka otevření existujícího experimentu**

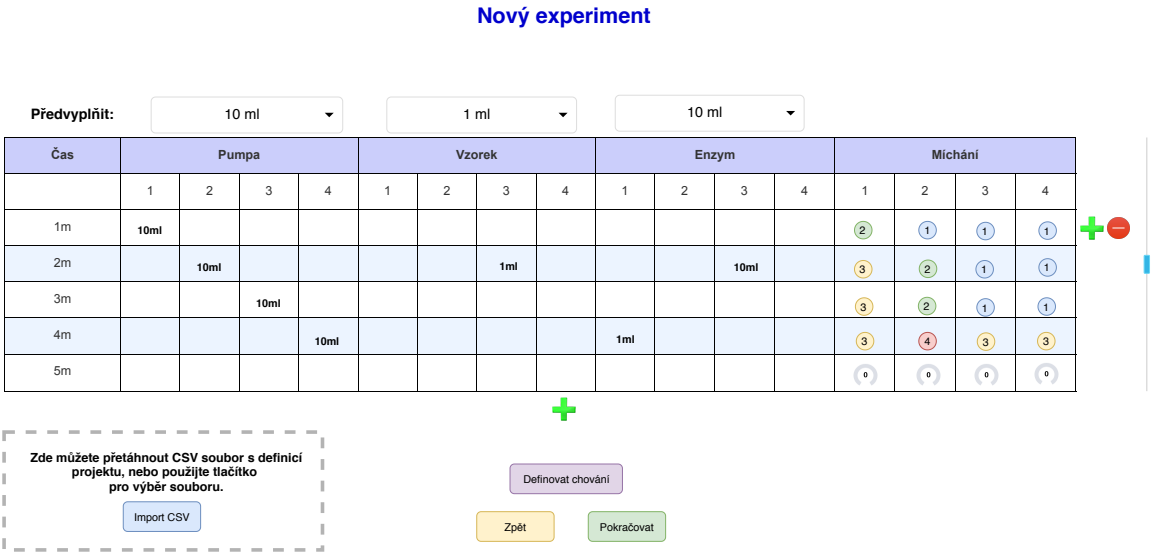
Návrh této stránky prošel dvěma fázemi, kdy nejprve byly jednotlivé projekty zobrazeny v prosté tabulce. Po konzultaci však došlo ke změně na zobrazení po skupinách dle účinné látky, jelikož se očekává více variant experimentu pro jednu účinnou látku. Seskupování pak povede na přehlednější zobrazení.

Po výběru konkrétního experimentu bude uživatel přesměrován na stránku se shodným rozhraním, jako má stránka vytvoření nového experimentu, avšak s tím rozdílem, že budou pole formuláře předvyplněny dle parametrů existujícího experimentu. Tato stránka má sloužit jak ke kontrole správnosti, tak k případné změně parametrů experimentu. V takovém případě bude následně uživatel upozorněn na změny v parametrech existujícího experimentu a v případě potvrzení bude takto změněný experiment uložen jako nová verze.

Plán experimentu

Účelem této stránky (viz obrázek 5.8) je možnost definovat časy a události, které mají v průběhu experimentu nastat. Těmito událostmi může být přečerpaní určitého objemu mezi vaky, odebrání vzorku, přidání enzymů, nebo změna rychlosti míchání. Plán experimentu je definován ve formě tabulky, kde jednotlivé řádky určují okamžik provedení a sloupce určují události a jejich parametry.

Nad jednotlivými sloupci se nachází pole pro výběr výchozí hodnoty, která má být po kliknutí do buňky tabulky umístěna. Na levé straně tabulky se pak nachází dvojice tlačítek



Obrázek 5.8: **Obrazovka s parametry experimentu.**

sloužících k přidání nového řádku pod aktuální řádek a tlačítko pro odstranění aktuálního řádku. Pod tabulkou se pak nachází tlačítko pro přidání nového řádku na konec tabulky.

Ve spodní části obrazovky se zde nachází trojice tlačítek, které umožňují návrat na předchozí stránku (po potvrzení uživatelem), pokračování na obrazovku pro spuštění experimentu, nebo zobrazení stránky pro definování řízení průběhu experimentu pomocí vizuální specifikace.

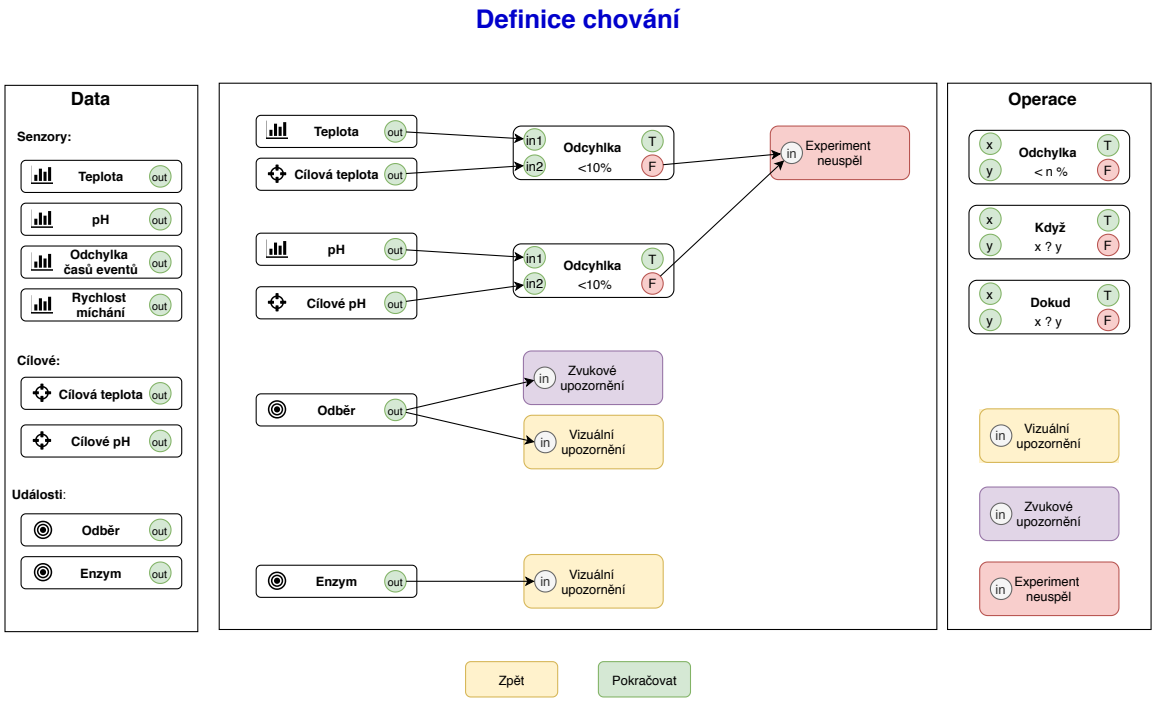
Během procesu návrhu této obrazovky bylo k původně jednoduché tabulce postupně přidáno rozhraní pro výběr automaticky doplňované hodnoty a možnost nahrát CSV soubor v případě projektů s velkým časovým rozsahem a množstvím událostí. Výběr souboru je realizován dialogovým oknem zobrazeným po kliknutí na tlačítko „Import CSV“, nebo přetažením souboru do vyznačené oblasti.

Vizuální specifikace řízení průběhu experimentu

Tato stránka umožňuje pomocí propojení bloků vizuálně specifikovat reakce na různé stavy experimentu. Bloky jsou trojího typu. Prvním typem je blok sloužící jako zdroj dat, druhým typem je blok pro vyhodnocení podmínek a třetím typem je blok akce. Spojováním těchto bloků do funkčních celků může být specifikováno pokročilé chování umožňující automatizovat reakce na určité stavy a parametry experimentů.

Stránka je rozdělena do tří částí, kdy v levé části obrazovky je umístěna nabídka bloků zdrojů dat, které obsahují název a výstupní port. Zdroje dat jsou buď sledované veličiny experimentu, nebo konstanty. V pravé části obrazovky se pak nachází nabídka obsahující bloky podmínek, které vyhodnocují hodnoty na svých vstupech a generují logický stav (pravda / nepravda) na svých výstupních portech. Ve spodní polovině pravé nabídky se pak nachází bloky akcí, kterými můžou být například nastavení experimentu do stavu neúspěšný, zobrazení upozornění uživateli, nebo zvukové upozornění. Prostřední část obrazovky pak slouží jako editor, do kterého mohou být bloky z nabídky přetaženy a kde mohou být spojovány do funkčních celků.

Stejně jako na předchozích stránkách může uživatel opustit návrh použitím tlačítka zpět a potvrzením, že si skutečně přeje odejít, nebo pokračováním, které změny ve vizuální specifikaci potvrdí.



Obrázek 5.9: **Obrazovka pro vizuální specifikaci řízení experimentu**

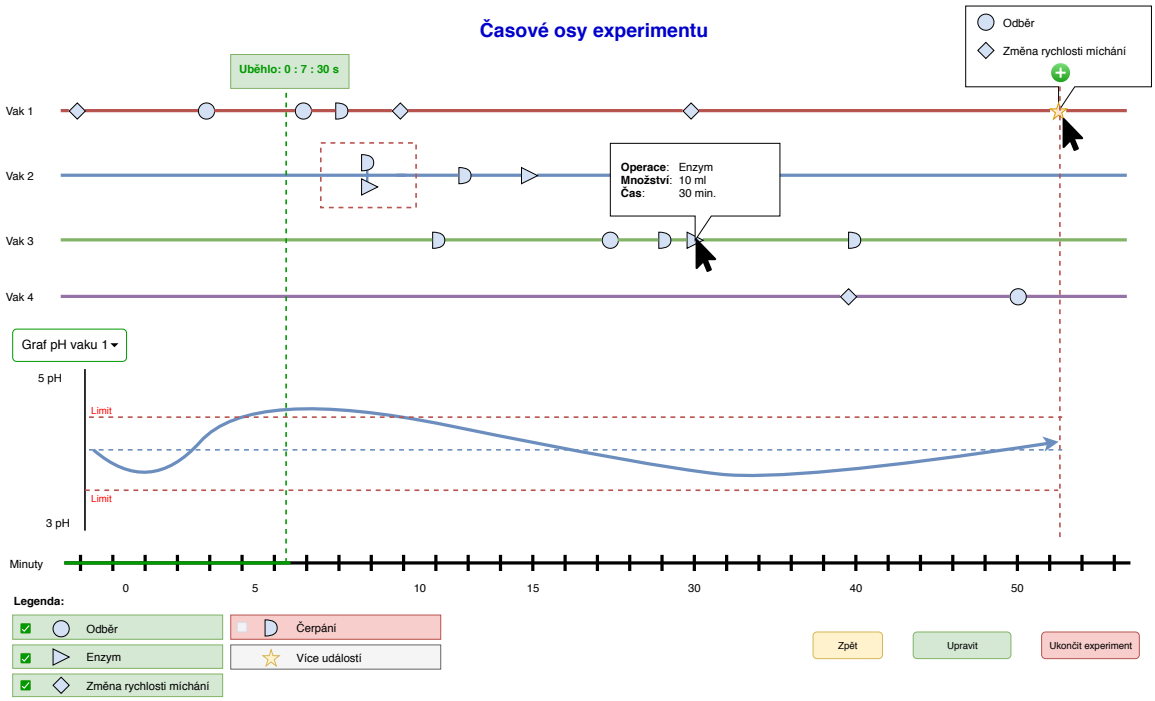
Průběh experimentu

Po stisknutí tlačítka pokračovat na stránce s parametry experimentu je uživatel přesměrován na tuto stránku, která slouží ke sledování průběhu experimentu. Průběh experimentů je vizualizován pomocí několika horizontálních os a časové osy pro jednotlivé vaky. Časový průběh experimentu je vizualizován posouváním zelené vertikální čáry, která znázorňuje aktuální časový bod na všech osách. Události jsou na osách vyznačeny pomocí značek dle typu operace.

Uživatel má možnost pomocí výběrového pole zvolit požadovaný graf, který si přeje v průběhu času sledovat a také pomocí legendy v levé spodní části obrazovky zvolit druhy událostí, které mají být na osách zobrazeny. Při umístění kurzoru na značku nacházející se na ose bude uživateli zobrazeno okno s parametry dané události. Ve spodní části obrazovky se pak nachází tlačítka pro návrat na předchozí obrazovku v případě neaktivního experimentu a tlačítko pro spuštění, popřípadě ukončení experimentu. Použitím tohoto tlačítka u probíhajícího experimentu dojde k upozornění uživatele, zda si přeje ukončit experiment a v případě potvrzení k přesměrování na stránku s výsledným protokolem experimentu.

Výsledky experimentu

Jedná se o závěrečnou stránku shrnující průběh experimentu a všechny důležité informace o jeho parametrech. Tato stránka bude sloužit zejména jako přehled, ale také k exportování



Obrázek 5.10: **Obrazovka pro sledování a řízení průběhu experimentu**

výsledků experimentu do různých formátů. Výstupem by měl být dokument ve formátu PDF a také datový soubor CSV obsahující výsledky měření.

5.4 Volba technologií

Výběr technologií je při realizaci každého softwarového projektu jedním z nejdůležitějších kroků, jelikož má zásadní vliv na téměř všechny aspekty výsledné aplikace. Z tohoto důvodu jsem při výběru vhodných technologií zvažoval všechny dostupné možnosti a vybíral zejména dle modularity, multiplatformnosti a podpory daných technologií. Z tohoto důvodu je v této kapitole nejdříve provedena přehledová studie dostupných technologií a knihoven, které by mohly být použity k realizaci aplikace. Výstupem této podkapitoly je pak volba konkrétních technologií a knihoven, na kterých bude aplikace založena.

Mezi vhodné kandidáty patřily například Qt, Java Swing, WPF, Django, Angular, React a Vue.js. Při výběru jsem zohlednil požadavek výzkumné skupiny na vytvoření aplikace, kterou by bylo možné jednoduše přenést při výměně počítače operátora. Druhým aspektem pak byla výše zmíněná modularita a multiplatformnost. Jako nejvhodnější se z těchto hledisek jevily webové technologie, které umožňují provoz bez nutnosti instalace a poskytují vhodný technologický základ pro realizaci modulárních uživatelských rozhraní.

Vzhledem k tomu, že tato kritéria splňují frameworky Angular, React a Vue.js, bylo nezbytné rozhodnout, který z těchto bude použit pro realizaci. Po důkladném seznámení s každým z nich jsem se rozhodl pro použití Vue.js, které představuje vhodný kompromis mezi velmi robustním frameworkem Angular a naopak minimalistickým frameworkem React.

Jako realizační základ jsem zvolil kombinaci frameworku Vue.js, knihovny komponent Vuetify a HTTP knihovny Axios.

5.4.1 Software a knihovny pro vizuální programování

V průběhu návrhu vyplynulo, že v novém uživatelském rozhraní bude obsažena obrazovka umožňující definovat automatické reakce na různé stavy v průběhu experimentu. Aby bylo dosaženo co možná největší flexibility a nejširších možností pro nastavení těchto událostí a následných reakcí, bylo zvoleno použití techniky vizuálního programování, které umožňuje spojováním funkčních bloků vytvářet jednoduché programy a větvení.

Za tímto účelem existuje rozsáhlá škála programů a knihoven, které mohou být pro realizaci použity. Následující podkapitoly se proto věnují rozboru několika z těchto programů a knihoven za cílem výběru nejvhodnějšího řešení pro použití v této práci.

Node Red

Prvním analyzovaným systémem byl Node Red. Jedná se o multiplatformní systém pro vizuální programování vytvořený v jazyce Node.js. Slouží zejména k propojování zařízení, API a online služeb do funkčních celků a obsahuje velké množství již definovaných bloků a nástrojů, které jej činí uživatelsky přívětivým. [9]

Nevýhodou tohoto systému je však to, že je nezbytné, aby byl instalován lokálně, na mikropočítači typu Raspberry Pi, nebo v cloudu. To komplikuje jeho nasazení v rámci webových aplikací a klade zvýšené nároky při použití.

Z hlediska návrhu nového GUI je tento systém až příliš robustní a jeho komplexnost nepřináší větší výhody.

Total.js Flow

Total.js je rozsáhlá platforma pro vývoj webových aplikací na bázi Node.js, která obsahuje komplexní moduly jako CMS, E-shop, modul administrace a mnohé menší moduly, mezi kterými je i modul Flow. Tento modul je určen k vytváření vizuálních programů pomocí spojování bloků a jejich převod do spustitelné podoby. Jedná se tedy o vhodného kandidáta pro použití v tomto projektu. [11]

Bohužel, stejně jako předchozí systém, vyžaduje instalaci na straně klienta, nebo na libovolné platformě, která bude sloužit jako server.

Laboratory Virtual Instrument Engineering Workbench (LabVIEW)

Jedná se o velice pokročilý systém pro řízení laboratorních experimentů, který dokáže pomocí techniky vizuálního programování vygenerovat programy spustitelné přímo pod hostitelským operačním systémem. [8]

Tato schopnost může být v mnoha případech velmi výhodná, jelikož poskytuje vyšší výkonnost než interpretované programy, avšak pro použití v kombinaci s jinými technologiemi (zejména webovými) je tento způsob nepraktický. Další nevýhodou je poměrně vysoká cena za licenci, která v době psaní této práce činila zhruba 153 000 Kč v případě plné verze.

jsPlumb

Jedná se o open-source knihovnu založenou na jazyce JavaScript, která je určena pro vizuální spojování objektů v prostředí webového prohlížeče. Funguje na základě manipulace s obrazovým formátem SVG. Jeho přímou vizualizací a propojením s JavaScriptem pomocí eventů je možné v rámci prohlížeče vytvářet pokročilé vývojové diagramy či jiné struktury. [7]

Největší výhodou této knihovny je to, že funguje přímo v rámci webového prohlížeče a nepožaduje instalaci. Díky této vlastnosti a také své malé velikosti se výborně hodí pro vytváření webových aplikací s podporou vizuálního programování. Další výhodou je poměrně rozsáhlá uživatelská základna a dostatek ukázkových projektů, ze kterých lze vycházet. Nevýhodou této knihovny je pak její poměrně velká komplexnost, která značně převyšuje potřeby této práce.

Rete.js

Rete.js je modulární webový framework, který umožňuje vytvořit v rámci webového prohlížeče editor pro vizuální programování s použitím propojitelných bloků. S jeho použitím je možné definovat bloky a propojení tak, aby byl uživateli poskytnuto srozumitelné rozhraní pro vytvoření vlastních programů. [10]

Tento framework je rovněž založen na jazyku JavaScript, což ho činí ideálním kandidátem pro použití v této práci. Bohužel má však mnohem menší uživatelskou základnu a také počet ukázkových kódů než jsPlumb.

Zvolené řešení

Po zvážení všech technických parametrů a vlastností jednotlivých knihoven se jako nejvhodnější kandidát pro použití v této práci jevila knihovna jsPlumb, avšak během pokusů o její začlenění do systému jsem došel k závěru, že knihovna je až příliš komplexní a její použití zbytečně zvyšuje složitost systému.

Jako alternativní řešení pak byla zvolena knihovna Simple Flowchart[2] (vue-simple-flowchart), která neslouží k vizuálnímu programování, ale pouze k vytváření jednoduchých vývojových diagramů. Knihovnu bylo možné díky její jednoduchosti upravit tak, aby přesně odpovídala potřebám tohoto projektu. Pro vykonávaní výstupu upravené knihovny byla implementována vlastní třída `Behavior`, která je popsána v podkapitole 6.1.7.

5.4.2 Knihovny, frameworky a podpůrný software

Pro realizaci tohoto projektu byly vybrány běžně využívané nástroje a knihovny, které mají dostatečně velkou uživatelskou základnu. Tato volba by měla zajistit jejich kvalitu a zejména dlouhodobou podporu, jelikož v dynamickém světě JavaScriptových frameworků dochází poměrně často ke vzniku nových technologií a jejich následnému opouštění.

Vue a Vuetify

Vue, spolu s knihovnou komponent Vuetify, zde slouží jako technologický základ, který spojuje všechny následující knihovny a vytváří prostředí pro jednoduchý a efektivní vývoj. [15] Detailní informace o těchto frameworcích a jejich použití jsou uvedeny v kapitole 4.

Součástí tohoto frameworku je také sada vývojářských nástrojů, které umožňují automatické spuštění webového serveru a kompilaci. Systém dále umožňuje i automatickou rekompilaci v případě detekovaných změn ve zdrojových souborech.

[2] https://github.com/Jeffreyrn/vue-simple-flowchart

Vuetify

Vuetify je knihovna komponent pro Vue.js, která přidává rozsáhlou škálu komponent, umožňující rychlý vývoj webových rozhraní tvořených standardními komponentami Material Design. Díky použití standardizovaných prvků a technologií je možné vytvořit unifikované rozhraní pro webové prohlížeče i mobilní aplikace. [25]

Axios

Tato knihovna[3] rozšiřuje Vue.js o možnost posílání a zpracování HTTP requestů založené na designovém patternu Promise. Díky tomuto rozšíření je možné jednoduše vytvářet asynchronní dotazy na REST API a reagovat na odpovědi.

Vue cookie

Knihovna Vue cookie[4] rozšiřuje Vue.js o možnost rychlé a jednoduché správy cookies přímo z komponent. Díky této knihovně je možné perzistentně ukládat uživatelská nastavení na straně prohlížeče, ale také identifikaci uživatelské relace použitou pro přihlášení do systému.

Eslint

Eslint[5] je statický kódový analyzátor, který umožňuje detekovat problémy a chyby v JavaScriptovém kódu. Tato funkce je velmi přínosná, jelikož pomáhá vyvíjet aplikace bez nadbytečných importů, nevyužitých proměnných a mrtvých větví programu, které by bez tohoto nástroje nemusely být odhaleny.

Nástroj dále varuje i před nestandardními operacemi provedenými nad datovými strukturami Vue.js, které by mohly vézt k záludným chybám vzniklým důsledky nepředpokládaných vedlejších účinků (side effect).

Babel

Babel[6] je kompilátor JavaScriptového kódu, který transformuje kód do podoby, která je zpětně kompatibilní s různými prohlížeči a jejich verzemi.

Díky tomuto nástroji není nutné optimalizovat výslednou aplikaci pro různé typy prohlížečů. Při použití standardních prvků a dodržení doporučených postupů je tato kompatibilita zajištěna automaticky.

TypeScript

Tato nadstavba[7] rozšiřuje jazyk JavaScript o možnost používání datových typů a nových jazykových konstrukcí jako například rozhraní tříd (interface). Typová kontrola a možnost deklarovat rozhraní je vhodná zejména k určení formátu komunikace API nebo komponent a jejich následnému dodržování. Tento přístup pomáhá eliminovat například chyby typu srovnávání nekompatibilních datových typů a mnohé jiné problémy.

[3] https://github.com/axios/axios
[4] https://www.npmjs.com/package/vue-cookie
[5] https://eslint.org/
[6] https://babeljs.io/
[7] https://www.typescriptlang.org/

i18n

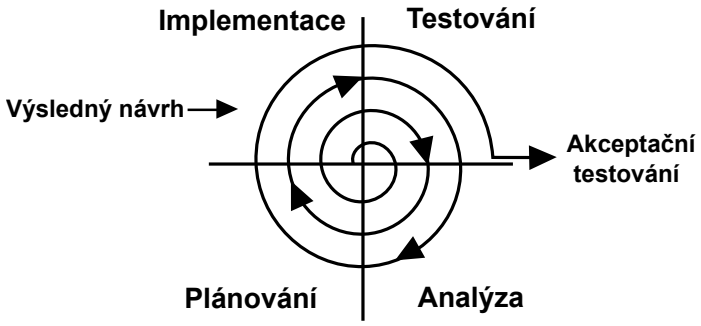
i18n[8] je univerzální internacionalizační framework sloužící k vytváření vícejazyčných aplikací. Systém je založen na použití zástupných řetězců, které jsou následně nahrazeny překladem dle zvolené jazykové verze. Texty překladů mohou obsahovat libovolné počty zástupných symbolů pro nahrazení proměnnými, mohou být zanořovány a doplněny podmínkami pro dosažení požadovaného chování například dle pohlaví uživatele.

[8] https://github.com/mashpie/i18n-node

Kapitola 6

Implementace

V dalších podkapitolách je věnována pozornost procesu vývoje uživatelského rozhraní, testovací API a třídy Behavior. Vývoj všech částí systému byl prováděn agilním iteračním přístupem, který je naznačen na obrázku 6.1.



Obrázek 6.1: **Model přístupu použitého k vývoji této práce** (Původní verze obrázku převzata z https://bit.ly/302Kfi5)

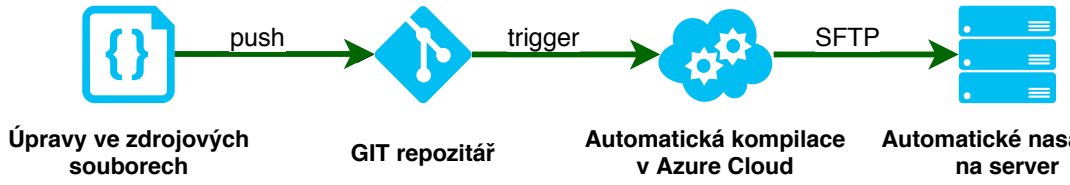
6.1 Klientská aplikace

6.1.1 Proces vývoje

Vývoj uživatelského rozhraní byl založen na agilní metodologii s inkrementálním přístupem, kdy v každé iteraci byl prováděn návrh, implementace, testování a vyhodnocení změn v projektu. K vývoji byl využit systém Azure DevOps obsahující GIT repozitář spolu s automatickou CI/CD pipeline[1] (definována souborem `azure-pipelines.yaml`, která zajišťovala kompilaci v Azure Cloud a nasazení nové verze systému na virtuální server. Ten sloužil jako hosting pro webovou aplikaci i testovací API. API zde, pro účely testování, simulovala chod reálného laboratorního zařízení.

Díky tomuto přístupu mohli členové výzkumné skupiny v reálném čase testovat změny v uživatelském rozhraní a poskytovat zpětnou vazbu pro další úpravy systému. Při provedení rozsáhlejších úprav systému bylo provedeno řízené uživatelské testování změněných částí. Zjištěné nedostatky byly následně analyzovány a zpracovány do návrhu změn uživatelského rozhraní. Po konzultaci byly změny do systému zapracovány.

[1] https://azure.microsoft.com/cs-cz/services/devops/pipelines/



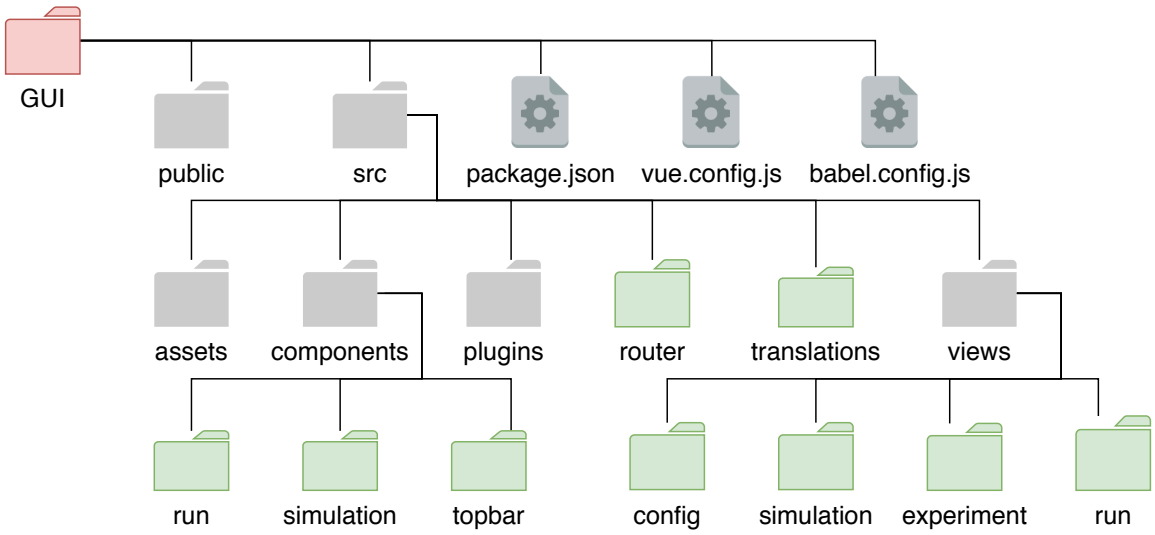
Obrázek 6.2: **CI/CD pipeline** – tento obrázek popisuje proces automatické kompilace a nasazení na server, který byl použit při vývoji a testování tohoto projektu. Proces začíná v okamžiku vložení změn do repositáře (push), které mají za následek spuštění automatického přeložení zdrojových kódů v rámci Azure Cloud (trigger v azure-pipelines.yaml) a následného nahrání na cílový server pomocí protokolu SFTP.

6.1.2 Architektura aplikace

Jako základ architektury aplikace posloužilo rozdělení na jednotlivé pohledy (view), které vycházely z obrazovek vytvořených ve fázi návrhu. Komplexnější části obrazovek jsou dále rozděleny na jednotlivé komponenty a jsou také vytvořeny komponenty pro opakující se části napříč všemi pohledy. Díky tomuto přístupu je dosažena vysoká modularita a omezuje se redundance opětovným využitím dílčích komponent.

Adresářová struktura

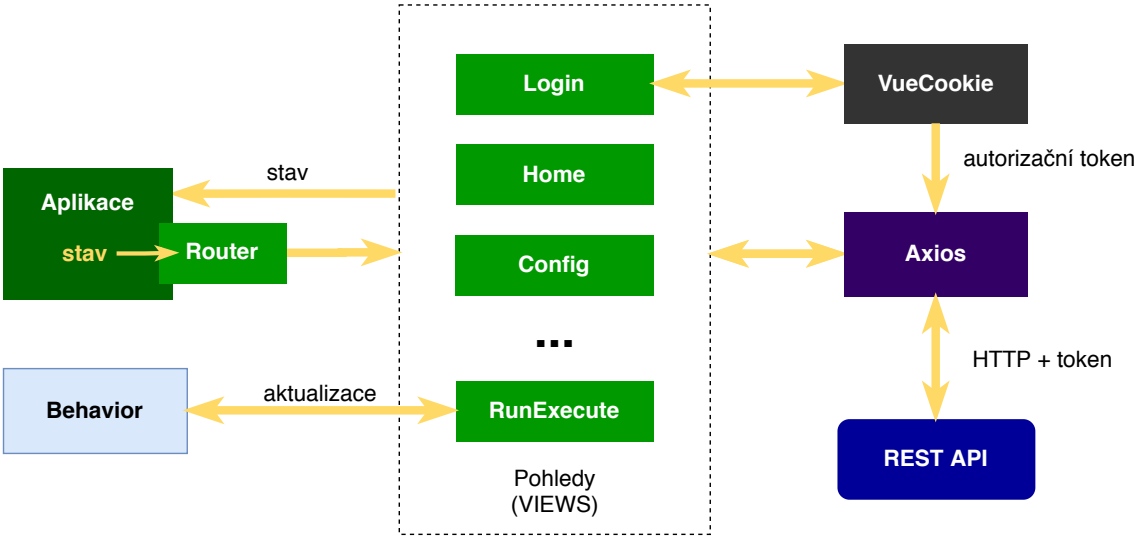
Základní architektura aplikace byla založena na standardním výchozím projektu pro Vuetify, který byl rozšířen o další adresáře pro umístění souborů knihoven a pro lepší dekompozici aplikace.



Obrázek 6.3: **Adresářová struktura a konfiguračními soubory** – zelené adresáře byly doplněny k výchozí adresářové struktuře projektu. Konfigurační soubor `package.json` slouží k instalaci všech nezbytných knihoven a softwarových balíčků a definici maker pro spouštění a kompilaci projektu. Soubory `vue.config.js` a `babel.config.js` pak slouží ke konfiguraci frameworku Vue a kompilátoru Babel.

Koncept systému

Pro vytvoření ucelené představy o vnitřním fungování systému byl vytvořen také tento diagram (6.4), který reprezentuje komunikaci mezi systémem Vue, pohledy, knihovnami a REST API. Diagram dále zachycuje způsob změn stavu systému a jejich propagaci.



Obrázek 6.4: **Koncept funkce aplikace** – Diagram znázorňuje způsob, jakým aplikace reaguje na změny stavu vyvolané událostmi v komponentách a jak komponenty komunikují s dalšími knihovnami a třídou Behavior. Diagram také zobrazuje způsob předávání autentizačního tokenu knihovně Axios z cookie, který realizuje knihovna VueCookie. Dále pak následují HTTP dotazy s tímto tokenem umístěným v hlavičce sessionId, které jsou zasílány na cílové REST API.

Interní datové struktury

Pro poskytnutí pohledu na interní reprezentace dat byl vytvořen následující diagram, který ve zjednodušené podobě reprezentuje interní datovou strukturu v objektové notaci JSON. Tyto datové typy, objekty a jejich části jsou využity pro interní reprezentaci na straně Vue, pro komunikaci pomocí REST API i pro perzistentní uložení na straně zařízení v NoSQL databázi NeDB[24]. Detailní popis jednotlivých datových typů, vzhledem k jejich velkému rozsahu, naleznete v příloze B.

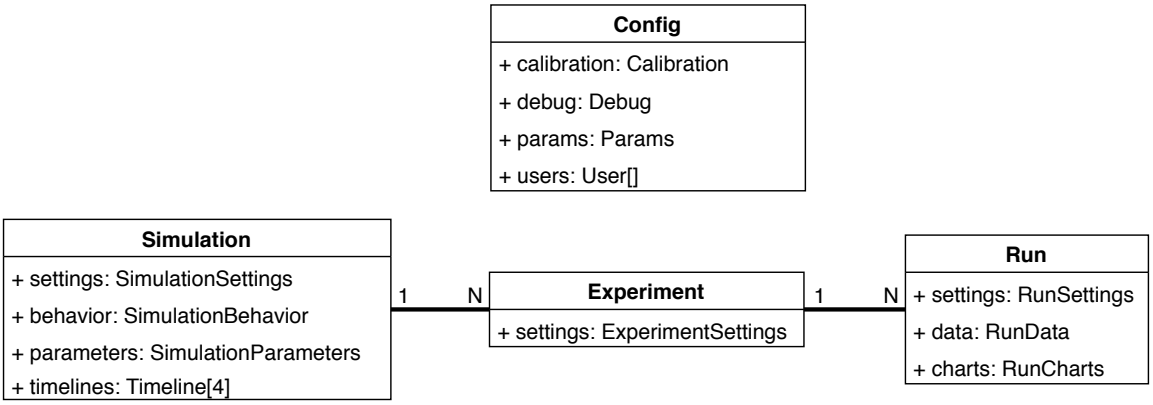
6.1.3 Komponenty uživatelského rozhraní

Při programování systému byly opakující se části, nebo části realizující specializovanou část uživatelského rozhraní odděleny do komponent, což umožňují vytvářet přehlednější modulární kód bez zbytečné redundance.

TopBar

Jedná se o základní komponentu celého systému, která realizuje horní lištu nacházející se na téměř všech obrazovkách aplikace. Výjimku tvoří pouze obrazovka pro přihlášení uživatele.

Komponenta je tvořena horizontálním panelem obsahujícím více částí, kde první zleva je ikonové tlačítko, sloužící jako vstup do postranní nabídky, která se po kliknutí zobrazí. Pro

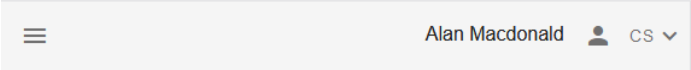


Obrázek 6.5: **Interní datová struktura** – diagram představuje zjednodušenou reprezentaci interní datové struktury a datové typy. Tento diagram vychází z informací uvedených v příloze B.2

její skrytí stačí kliknout kdekoli do prostoru stránky mimo nabídku. Obsahem nabídky jsou přepínače pro ovládání zvukových upozornění a skrytí nepřístupných položek v nabídkách a tabulkách pro usnadnění orientace v systému.

V pravé části panelu se pak nachází jméno a příjmení aktuálně přihlášeného uživatele, ikonové tlačítko pro přístup k uživatelské nabídce obsahující tlačítko pro odhlášení a tlačítko sloužící k zobrazení nabídky pro výběr jazyka systému.

Komponenta je vytvořena pomocí komponent obsažených v knihovně Vuetify, jako jsou například `v-menu` sloužící zde k realizaci všech nabídek, `v-list` pro vytvoření seznamu položek nabídek, tlačítek `v-btn` a ikon `v-icon`. Výjimku zde tvoří vlastní komponenta `Language`, která realizuje nabídku pro výběr jazyka.



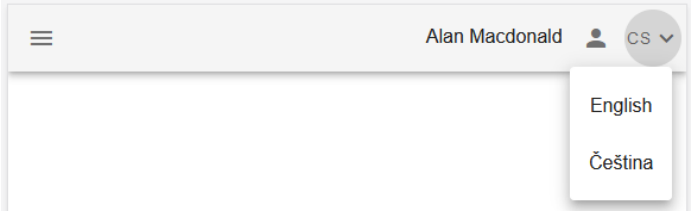
Obrázek 6.6: **Snímek komponenty TopBar**

Language

Tato komponenta byla vytvořena pro oddělené funkcí pro výběr jazyka od rodičovské komponenty `TopBar`.

Obsahem této komponenty je výběrové menu pro volbu požadovaného jazyka systému a metody pro provedení této operace. Zvolený jazyk je zde ukládán jak do interní proměnné překladového frameworku `i18n`, tak do cookie uložené na straně prohlížeče, aby byla zajištěna persistence volby jazyka například při obnovení stránky. Za tímto účelem byla využita knihovna `vue-cookie`, která rozšiřuje systém Vue.js o možnost práce s HTTP cookie.

Komponenta ke svému fungování využívá soubory v adresáři `translations`, kde `language.js` slouží jako seznam dostupných jazyků a ostatní soubory (například `cs.js`, `en.js`) slouží jako překladové soubory obsahující objekty typu JSON se zástupnými řetězci a překlady. Detailní popis překladového systému naleznete v podkapitole 6.1.6.



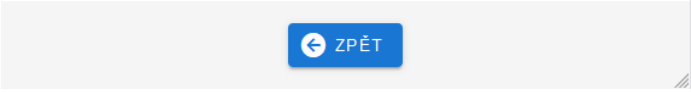
Obrázek 6.7: **Snímek komponenty Language**

Footer

Druhou základní komponentou je Footer, tedy zápatí, které se podobně jako předchozí komponenta nachází ve všech pohledech systému, kromě výjimek tvořených obrazovkami pro přihlášení a hlavního menu.

Komponenta je taktéž tvořena horizontálním panelem a pouze jediným tlačítkem nacházejícím se uprostřed panelu. Tímto tlačítkem je tlačítko zpět, které slouží k návratu na předchozí obrazovku.

Technická realizace je velmi přímočará. Základ tvoří komponenta `v-footer` realizující panel zápatí a tlačítko `v-btn` obsahující ikonu `v-icon`. Komponenta dále zachytává události oznamující změny ve formuláři dané stránky a implementuje systém pro upozornění uživatele na neuložené změny při pokusu o použití tlačítka zpět.



Obrázek 6.8: **Snímek komponenty Footer**

Tile

Jednou z prvních vytvořených komponent byla komponenta Tile, tedy dlaždice, která slouží pro realizaci položek v hlavním menu aplikace a také v menu konfigurace zařízení.

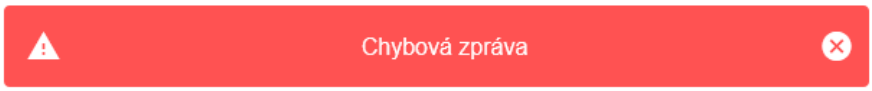
Konstrukce této komponenty je velice jednoduchá, jelikož ke své funkci používá pouze komponenty `v-card` realizující samotnou dlaždici a ikonovou komponentu `v-icon`. Data jsou této komponentě předávána pomocí property s názvem `tile` a datovým typem Object obsahující položky `link`, `color`, `icon` a `title`.

Obrázek 6.9: **Snímek komponenty Tile**

ErrorAlert

Další nezbytnou komponentou je `ErrorAlert`, která slouží k zobrazování upozornění na chyby. Její využití spočívá zejména v oznamování chyb komunikace s API a případných jiných selhání systému.

Komponenta je vytvořena s pomocí komponenty `v-alert`, která se zobrazuje jako absolutně umístěný prvek položený nad všemi ostatními prvky uživatelského rozhraní pomocí CSS parametru `z-index`. Komponenta je importována v hlavním souboru aplikace, aby bylo možné její vyvolání na všech obrazovkách. Její vyvolání je možné umístěním zprávy pro uživatele do globálně přístupného objektu `$root.error.errorString` a nastavením `$root.error.active` na hodnotu `true`.



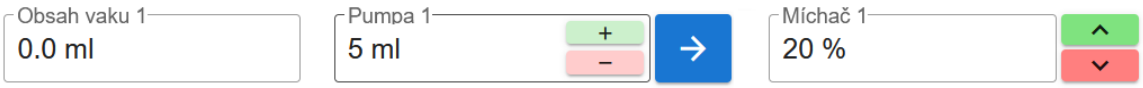
Obrázek 6.10: **Snímek komponenty ErrorAlert**

NumberField

Komponenta `NumberField` byla vytvořena pro usnadnění vytváření polí pro zadávání číselných hodnot a jejich ověřování. Jejím základem je komponenta `v-text-field`, která byla rozšířena o properties `min` a `max`, které slouží k automatické validaci, zda je hodnota v povoleném rozsahu. Komponenta rovněž neumožňuje uložit jinou než číselnou hodnotu. Povolenými vstupy jsou pouze celá a desetinná čísla.

ControlField

Tato komponenta byla vytvořena jako rozšíření standardních textových polí `v-text-field` v rozhraní pro řízení průběhu experimentu, kde poskytuje možnost inkrementace, dekrementace a odeslání hodnoty.



Obrázek 6.11: **Snímek komponenty ControlField** – v levé části je komponenta v běžném režimu, ve středu je komponenta v režimu `controls` a `submit`, poslední snímek ukazuje komponentu v aktivním režimu `controlEvents`

Komponenta umožňuje zobrazení přídavných tlačítek pro inkrementaci a dekrementaci, která mohou být aktivována použitím property `controls`. Alternativní typ jejich zobrazení může být specifikován pomocí property `controlsAppend`, která způsobí umístění tlačítek pro inkrementaci vně textového pole a změní jejich symboly na šipky. Všechny properties a jejich použití jsou uvedeny v následujícím seznamu:

- `controls: bool` – Zobrazí uvnitř textového pole dvojici tlačítek pro inkrementaci a dekrementaci.

- `controlsAppend: bool` – Tlačítka pro změnu hodnoty budou umístěna vně textové pole.
- `controlEvents: bool` – Tlačítka pro změnu hodnoty hodnotu nemění, ale pouze vytváří události `@increment` a `@decrement`.
- `submit: bool` – Zobrazí na pravé vnější straně textového pole tlačítko, jehož použití vyvolá událost `@submit`.
- `placeholder: string` – Slouží k nastavení obsahu prázdného textového pole.
- `readonly: bool` – Slouží k zablokování změny obsahu textového pole uživatelem a skryje tlačítka.
- `disabled: bool` – Deaktivuje všechna tlačítka a textové pole.
- `error: bool` – Nastaví tlačítko do chybového stavu a zobrazí červené orámování.
- `unit: string` – Zobrazí zadaný textový řetězec jako jednotku.
- `label: string` – Slouží k nastavení nadpisu textového pole.
- `value: number` – Slouží k nastavení hodnoty textového pole.

Knob

Komponenta `Knob`, slouží k manipulaci číselných hodnot z určitého rozsahu s vizuálním zobrazením aktuální hodnoty. Nejvýznamnější části je v této komponentě disk umístěný nad číselnou hodnotou, který pomocí své barvy a zvýrazněné oblasti indikuje aktuální hodnotu. Největší přínos této komponenty je zvýšení čitelnosti řad číselných hodnot.

Komponenta je postavena na základě open-source komponenty `vue-knob-control`, která byla modifikována pro účely tohoto projektu. Modifikace spočívá zejména v úpravě disku zobrazovaného nad hodnotou tak, aby se jeho barva měnila od modré po červenou podle aktuální hodnoty. Byla také doplněna property `readonly`, která umožňuje deaktivaci změny hodnoty pomocí kliknutí na disk.

Obrázek 6.12: **Snímek vzhledu komponenty Knob**

Timeline

Timeline je komponenta pro zobrazování časových os experimentů s vyznačenými událostmi. Komponenta je použita pouze na obrazovce pro řízení běhu experimentu.

Její realizace je založena na HTML značce `<ol>` sloužící k vytváření seřazených seznamů. Zobrazení seznamu je provedeno pouze horizontálně a prvky seznamu jsou umístěny na konkrétní pozici dle času, ve kterém mají být provedeny. Pro zobrazení podkladové osy byl použit pseoudoprvek `:before`, který se zobrazuje šedou barvou. K vyznačení aktuální

Obrázek 6.13: **Snímek komponenty Timeline s událostmi**

pozice byl použit pseudoprvek `:after`, který překrývá zvolenou barvou část osy, která již byla provedena.

Jednotlivé prvky této komponenty jsou vytvořeny komponentou `TimelineEvent`, nebo `TimelineEventGroup` v případě překrývajících se událostí. Následující seznam uvádí dostupné properties a jejich význam.

- `id: string` – identifikátor časové osy
- `name: string` – jméno časové osy, které bude před ní zobrazeno
- `height: string` – výška časové osy
- `width: string` – délka časové osy
- `lineColor: string` – podkladová barva časové osy
- `lineColorPassed: string` – barva pro již proběhlou část osy
- `backgroundColor: string` – barva pozadí časové osy
- `duration: number` – délka experimentu
- `eventSize: string` – velikost události na časové ose
- `items: Timeline (T.26)` – pole událostí časové osy

TimelineEvent

TimelineEvent je komponenta sloužící k zobrazení konkrétní události na časové ose komponenty `Timeline` a je tvořena komponentou tlačítka `v-btn` v režimu kruhu (fab), ikony `v-icon` a `v-tooltip`, která slouží k vytvoření okna popisku události. Popisek události je aktivován pomocí direktivy `v-on` při umístění kurzoru myši nad událost. Typy událostí jsou stejně jako v původním návrhu rozlišeny pomocí grafických ikon z knihovny Material Design Icons a barev pro jednoduché určení jejich významu. Komponenta přijímá properties uvedené v následujícím seznamu:

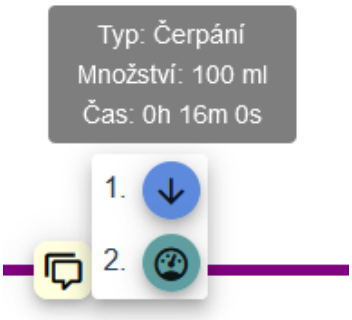
Obrázek 6.14: **Podoby komponenty TimelineEvent dle typu události**

- `item: TimelineEvent (T.27)` – Objekt obsahující všechny informace o události.
- `size: string` – Rozměry události na časové ose.

- `eventTypes: TimelineEventType[5]` (T.28) – Pole obsahující typy událostí, jejich ikony a barvy.
- `display: bool` – Zobrazí či skryje událost na časové ose.

TimelineEventGroup

Jak její název napovídá, tato komponenta slouží k shlukování událostí do skupin v případě, kdy se dvě události na časové ose nachází natolik blízko, že se překrývají. Překrývaní událostí je nežádoucí, jelikož by mohlo vést k přehlédnutí, nebo až neviditelnosti události.



Obrázek 6.15: **Snímek komponenty TimelineEventGroup s událostmi**

Komponenta ke své funkci využívá předchozí komponentu `TimelineEvent` dvojím způsobem. Jednak pro zobrazení vlastního symbolu skupiny událostí na časové ose a také k vytvoření položek událostí v okně komponenty `v-menu`, která slouží k zobrazení událostí ve skupině. Okno je stejně jako v případě `TimelineEvent` zobrazeno po najetí kurzorem nad skupinu. Properties této komponenty jsou následující:

- `index: string` – identifikátor v rámci pole událostí
- `items: TimelineEvent[]` (T.27) – události obsažené v této skupinně
- `size: string` – velikost na časové ose
- `eventTypes: TimelineEventType[5]` (T.28) – Pole obsahující typy událostí, jejich ikony a barvy.
- `displayEvent: bool[5]` – pole booleovských hodnot s aktuálně povolenými typy událostí

CustomList

Tato komponenta byla vytvořena pro zjednodušení zobrazování seznamů na stránce s výsledky běhu experimentu. Komponenta je založena na komponentě `v-list`, která je použita společně s `v-list-item` a direktivou v-for k vygenerování kompletního seznamu obsaženého v property `items`. Barva pozadí seznamu může být nastavena pomocí property `color`. Struktura položky pole objektu předávaného pomocí property `items` je následující:

- `name: string` – Jméno položky
- `content: string` – HTML kód nebo text položky

- `style: string` – Stylové předpisy CSS
- `unit: string` – definující jednotku zobrazované položky
  - `text: string` – řetězec zobrazený jako jednotka
  - `position: "prepend"|"append"` – pozice jednoty (prepend pro jednotku před, append pro jednotu za)

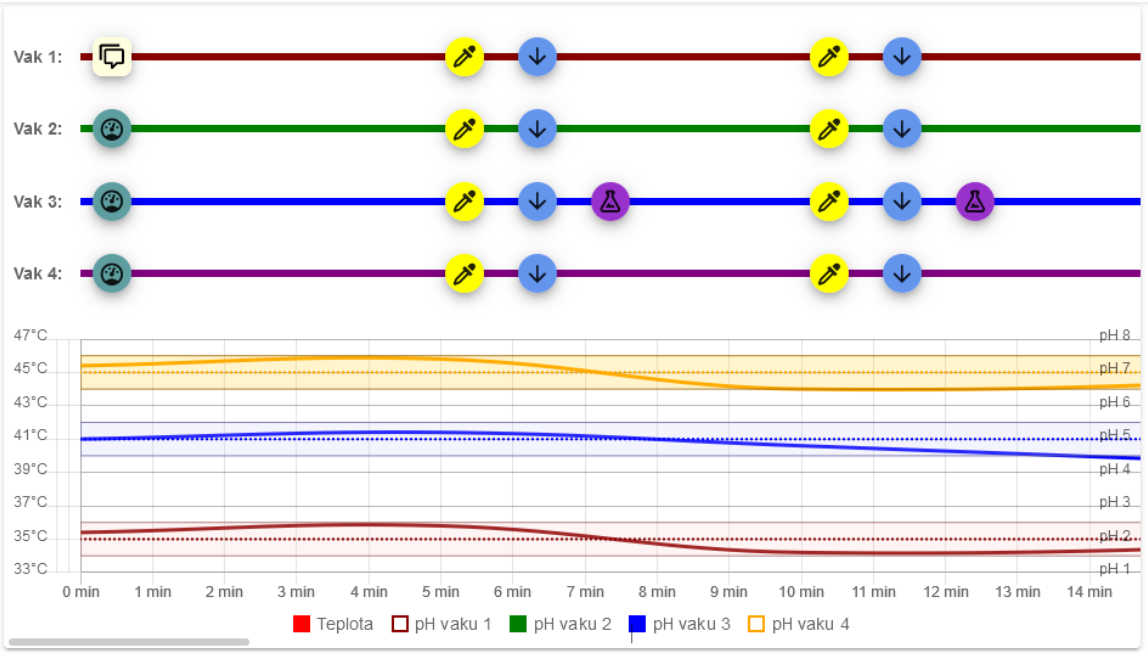
ProjectBehaviorDialog

Tato komponenta byla vytvořena pro oddělení poměrně rozsáhlého dialogového okna pro editaci parametrů funkčních bloků od stránky pro vizuální specifikaci průběhu řízení experimentu.

Komponenta je realizována pomocí komponenty `v-dialog`, která slouží k zobrazení dialogu, komponenty `v-card` pro vytvoření plochy okna a textových polí `v-text-field`. Textová pole jsou zobrazována, nebo skrývána dle typu editované události s pomocí direktivy `v-if`. Data jsou komponentě předávána skrz property `dialog` která má formát booleovské hodnoty sloužící k zobrazení, či skrytí komponenty a property `data: SimulationBehaviorNode` (T.19), která přijímá objekt zvoleného funkčního bloku.

ProgressSection

Účelem této komponenty je oddělení části funkcionality obrazovky pro sledování a řízení průběhu experimentu. Komponenta zapudruje části rozhraní, které slouží ke sledování průběhu experimentu, tedy graf a časové osy. Komponenta ke své činnosti využívá kontejner se skrytými přesahy a posunováním v horizontální ose (CSS vlastnost `overflow: scroll-x`), aby bylo možné zobrazit i velice dlouhé časové osy a grafy.



Obrázek 6.16: **Snímek sekce tvořené komponentou ProgressSection**

Sekce pro zobrazení časových os je tvořena komponentou `Timeline`, sekce grafu pak komponentou `ChartSection`. Komponenta přijímá tyto properties:

- `axisMaxPh: number` – maximální hodnota pH na ose grafu
- `axisMaxTemperature: number` – maximální hodnota teploty na ose grafu
- `eventTypes: TimelineEventType[5]` (T.28) – Pole obsahující typy událostí, jejich ikony a barvy.
- `timelines: Timelines` (T.25) – pole časových os
- `charts: RunCharts` (T.35) – objekt grafů
- `legendWidth: string` – délka legendy
- `progress: number` – počet sekund uplynulých od začátku experimentu
- `duration: number` – celková délka experimentu
- `settings: object` – objekt aktuální nastavení zobrazení GUI
- `showSidebar: bool` – aktivuje zobrazení postranního panelu
- `temperatureChartId: number` – identifikátor grafu teploty
- `updateKey: number` – klíč pro detekci aktualizace

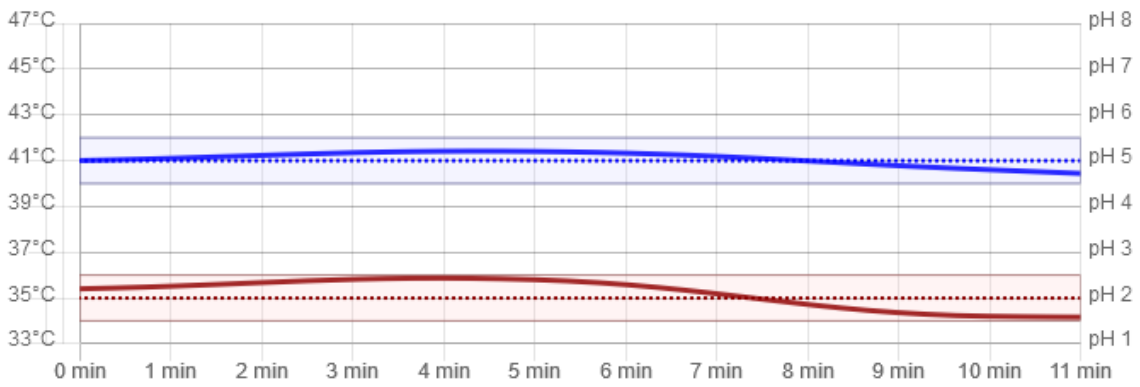
ChartSection

Účelem této komponenty je vytvořit graf, staticky umístěné popisky os a legendu nezávislé na posunu po horizontální ose v rámci komponenty `ProgressSection`. Statické umístění bylo dosaženo pomocí použití absolutní pozice těchto prvků a vypočítávání jejich nové pozice při posunu, či změně velikosti okna aplikace. Legenda zde slouží jak k identifikaci datových řad, tak k výběru veličin k zobrazení. Výběr, či zrušení výběru je realizováno pomocí kliknutí na příslušnou položku legendy, která funguje jako zaškrtávací pole. Tato změna oproti původnímu návrhu, ve kterém byl výběr grafů realizován pomocí výběrového pole, byla doplněna na základě uživatelských testů, které odhalily nižší uživatelskou přívětivost původního způsobu výběru. Přínosem této změny je zpřehlednění uživatelského rozhraní a vyšší uživatelská přívětivost.

K posunování v rámci časových os a grafu dochází automaticky během průběhu experimentu tak, aby byla aktuální pozice na časové ose a grafu vždy ve středu obrazovky, nebo v její levé části. Toto chování je možné deaktivovat buď použitím zaškrtávacího pole obsaženého v nabídce v komponentě **ControlSection**, nebo manuálním posunem použitím scrollbaru. Opětovná aktivace je možná pomocí stejného zaškrtávacího pole, nebo dvojklikem na scrollbar.

Komponenta ke svému fungování využívá zejména klasické HTML značky a reakci na událost `@scroll`. Díky zachycení této události a určení posunu oproti původní pozici je možné vypočítat nové umístění položek tak, aby zůstala zachována jejich původní pozice. K zachytávání použití scrollbaru je využita událost `@click`. Její properties jsou následující:

- `axisMaxPh: number` – maximální hodnota pH na ose grafu



Obrázek 6.17: **Snímek komponenty ChartsSection**

- `axisMaxTemperature: number` – maximální hodnota teploty na ose grafu
- `charts: RunCharts` (T.35) – objekt grafů
- `legendWidth: string` – délka legendy
- `progress: number` – počet sekund uplynulých od začátku experimentu
- `duration: number` – celková délka experimentu
- `scrollTo: number` – pozice, do které má být provedeno posunutí po ose x
- `updateKey: number` – klíč pro detekci aktualizace

LineChart

Účelem této komponenty je propojení knihovny `vue-chartjs` a pluginu `chartjs-plugin-annotation`. Díky jejich spojení je možné využívat novou HTML značku `line-chart`, která slouží k vytvoření instance XY čárového grafu s podporou anotací. Podpora anotací byla přidána, aby bylo možné v rámci grafu zobrazovat přerušované čáry představující cílové hodnoty a obdélníkové oblasti pro vyznačení povolených rozsahů hodnot.

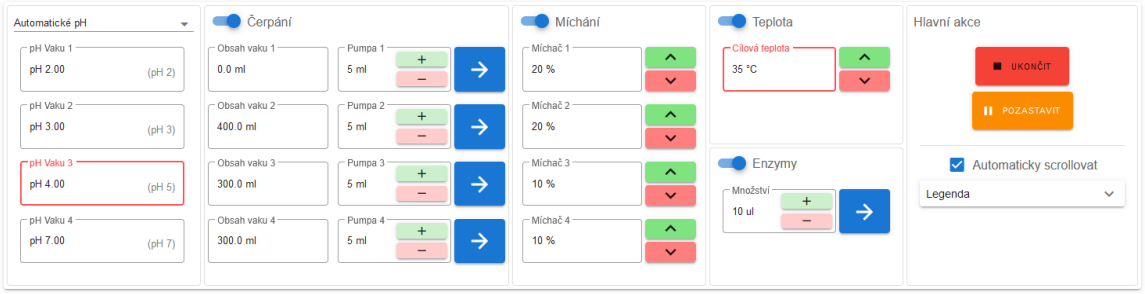
Technická realizace je poměrně minimalistická, jedná se zejména o importování obou knihoven a použití metody `mounted()` pro jejich inicializaci v okamžiku načtení stránky. Data spolu s konfigurací grafu jsou předána pomocí property `chartData: RunCharts` (T.35). Aktualizaci grafu je možné vykonat změnou hodnoty property `updateKey`.

ControlSection

Komponenta ControlSection byla vytvořena pro oddělení sekce pro řízení parametrů experimentu na stránce pro vykonání běhu experimentu. K tomu kroku bylo přistoupeno pro zjednodušení této stránky, jelikož je sekce pro řízení parametrů velmi obsáhlá.

Komponenta je tvořena sedmi kartami tvořenými komponentou `v-card`, hlavní z nich slouží jako obálka celé sekce obsahující jednotlivé podsekce. Jednotlivé podsekce jsou tvořeny zejména komponentami `ControlField` a několika dalšími komponentami, jako jsou `v-switch`, `v-btn` a `v-expansion-panel`.

Komponenty `v-switch` jsou použity v záhlaví jednotlivých podsekcí a slouží k jejich aktivaci a deaktivaci. V případě karty pro ovládání pH je tento přepínač nahrazen výběrovým



Obrázek 6.18: **Snímek komponenty ControlSection**

boxem pro volbu konkrétního režimu řízení pH. Na výběr je zde z možností automatického, poloautomatického a manuálního režimu řízení.

Karta s hlavními akcemi, která je umístěna v pravé části panelu obsahuje tlačítka pro ovládání běhu experimentu a dvě komponenty `v-expasnion-panel`, kde první obsahuje zaškrtávací pole pro aktivaci automatického posunování grafů a časových os při průběhu experimentu. Druhá pak poskytuje nabídku pro zobrazení, nebo skrytí typů událostí na časových osách. Tato část systému využívá ke svému fungování dotazy na endpoint `execute` (B.24) a následující properties:

- `targetConditions: TargetConditions` (T.33) – cílové podmínky experimentu
- `settings: object` – objekt aktuální nastavení zobrazení GUI
- `inputs: Experiment` (T.29) – objekt s daty experimentu
- `eventTypes: TimelineEventType[5]` (T.28) – Pole obsahující typy událostí, jejich ikony a barvy.
- `scroll: bool` – aktivuje, nebo deaktivuje automatické scrollování při načtení nových dat
- `progress: number` – počet sekund uplynulých od začátku experimentu

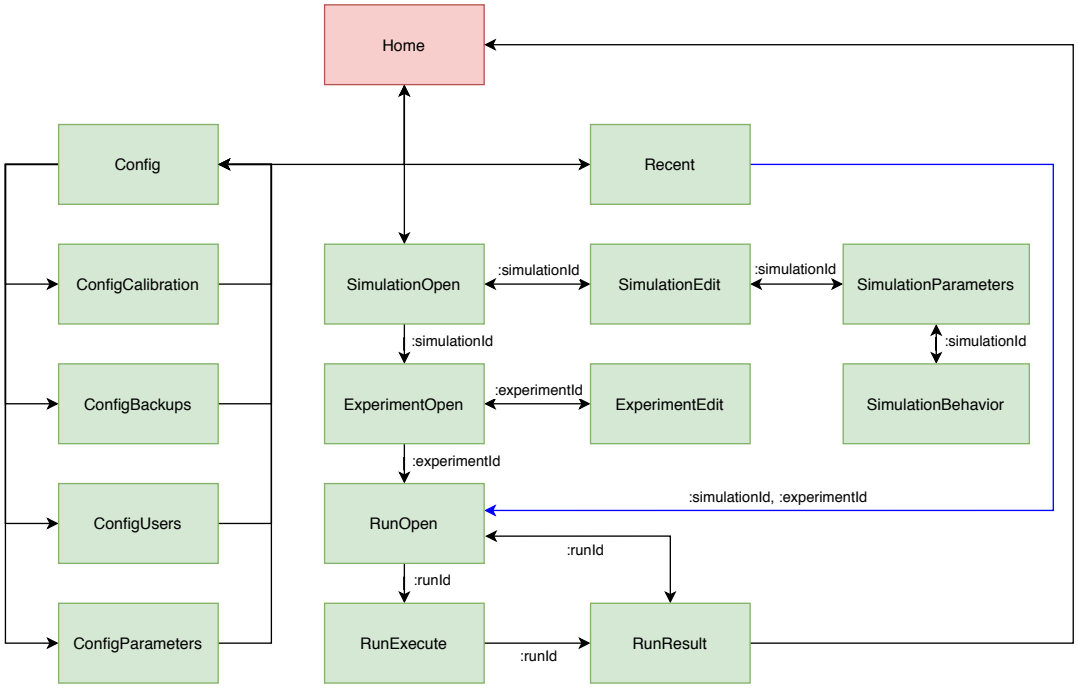
6.1.4 Směrování

Pro umožnění efektivní navigace mezi jednotlivými pohledy (views) byl v tomto projektu použit plugin `vue-router`, který je oficiálním modulem pro směrování ve frameworku Vue.

Pro směrování byl vytvořen systém cest v uživatelském rozhraní, které identifikují jak konkrétní pohledy, které mají být zobrazeny, tak i nesou identifikátory požadovaných simulací, experimentů a běhů. Jako příklad lze použít například nejdelší cestu v systému která má následující podobu: `/simulation/:simulationId/experiment/:experimentId/run/:runId/result`. Reální adresa by měla všechny pozice parametrů URL označené symbolem dvojtečky vyplněné identifikátorem ve formě celého čísla, nebo řetězce `new`, který signalizuje vytváření nové položky.

6.1.5 Pohledy

Pohledy jsou umístěny v adresáři views, který můžete vidět v kontextu celé aplikace na obrázku 6.3. Obrázek 6.20 pak ukazuje obsah adresáře views a jeho podadresářů. Nadpisy



Obrázek 6.19: **Diagram tras mezi pohledy rozhraní** – v diagramu můžete vidět vyznačené směrování mezi jednotlivými pohledy realizované modulem Router. Dále jsou zde vyznačeny parametry URL, které jsou použity k identifikaci. Modrou čarou je zde vyznačena zkratka, která byla přidána v důsledku uživatelských testů.

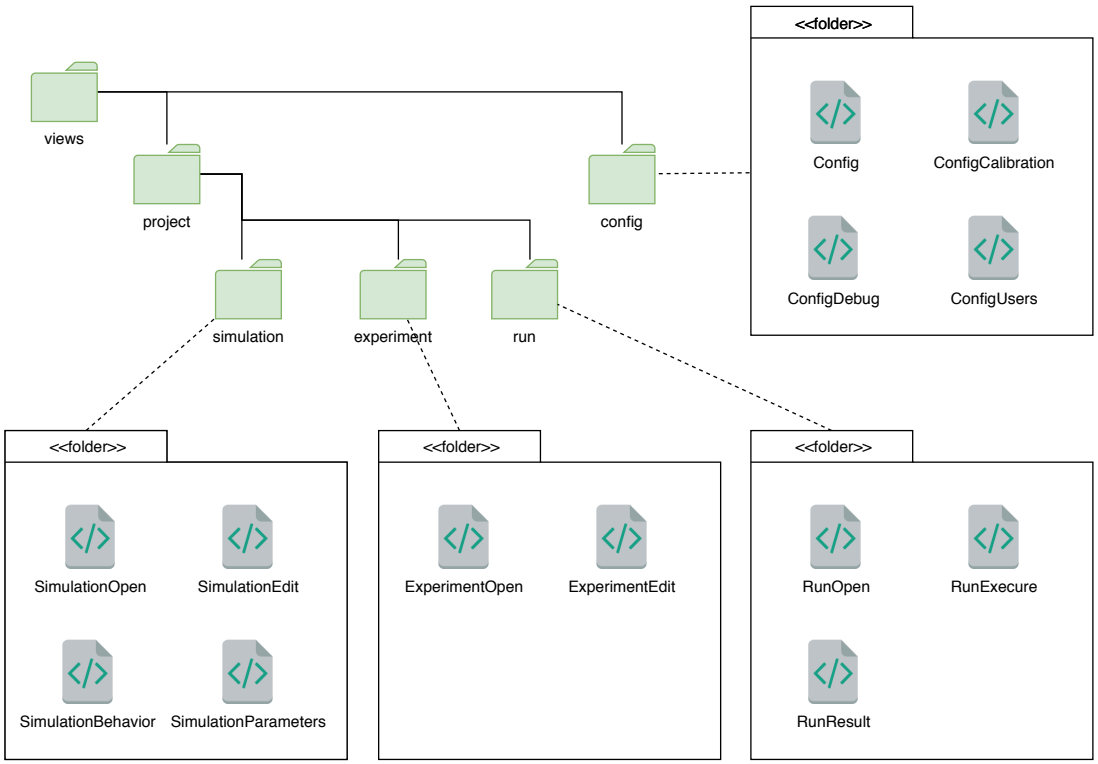
jednotlivých pohledů obsahují v závorce jméno pohledu (i souboru), které bylo využito při implementaci. Tato informace byla do nadpisů přidána pro zjednodušení orientace ve zdrojovém kódu a také dohledávání pohledu dle názvu ve zdrojovém kódu.

- **Project** – obsahuje skupinu obrazovek pro provádění a správu pokusů
  - Simulation – obsahuje skupinu obrazovek správu simulací
  - Experiment – obsahuje skupinu obrazovek správu experimentů
  - Run – obsahuje skupinu obrazovek pro provádění a správu běhů experimentů
- **Config** – obsahuje skupinu obrazovek pro správu zařízení

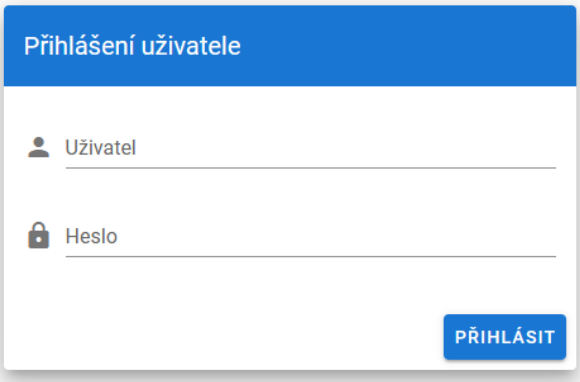
Přihlášení (`Login`)

Jedná se o úvodní obrazovku při vstupu nepřihlášeného uživatele do uživatelského rozhraní. Tato obrazovka je uživateli zobrazena i v případě vypršení uživatelské relace. V případě, že uživatel již má platnou uživatelskou relaci, je přesměrován do hlavní nabídky. Tento pohled využívá ke svému fungování api endpointy `login` (B.1) a `getUserData` (B.2).

Po technické stránce se jedná o jednoduchý formulář složený z komponent `v-text-input`, který umožňuje metodou POST odeslat požadavek na přihlášení na API endpoint `login`. Následná reakce se odvíjí od odpovědi API, která může být buď kladná (signalizováno HTTP kódem 200 OK), nebo záporná (signalizováno HTTP 401 Unauthorized). V případě záporné odpovědi je uživatel upozorněn na neplatné přihlášení, v opačném případě přesměrován na stránku hlavní nabídky.



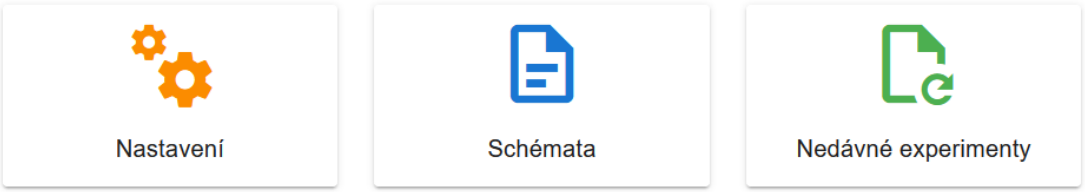
Obrázek 6.20: **Obsah adresáře views a jeho podadresářů** – jednotlivé podskupiny připojené k adresářům přerušovanou čarou zde představují obsah adresářů, ke kterým jsou připojeny



Obrázek 6.21: **Úvodní přihlašovací obrazovka**

Hlavní nabídka (`Home`)

Hlavní nabídka je realizovaná formou dlaždic sloužících jako odkazy na další stránky. Obsah nabídky se různí dle oprávnění uživatele. V případě běžného uživatele není dlaždice nastavení zobrazena.



Obrázek 6.22: **Hlavní nabídka**

Při realizaci obrazovky byla vícenásobně využita komponenta `Tile`, která slouží k vytvoření dlaždice na základě objektu obsahujícího její parametry (barva, ikona, text). Karty jsou automaticky vygenerovány na základě pole karet s použitím direktivy `v-for`.

Nabídka konfigurace (`Config`)

Jak název tohoto pohledu napovídá, jeho hlavním účelem je poskytnout administrátorovi nabídku pro výběr požadovaného konfiguračního panelu, který potřebuje zobrazit. Konstrukce tohoto pohledu vychází z konstrukce hlavní nabídky (`Home`), kde je využita komponenta `Tile`. Obsah této obrazovky je opět vygenerován pomocí direktivy `v-for` na základě pole obsahujícího data jednotlivých tlačítek.

Správa uživatelů (`ConfigUsers`)

Tento pohled slouží ke správě databáze uživatelů přístroje Golem, k čemuž využívá upravenou komponentu tabulky `v-data-table`, která je rozšířena o sloupec s tlačítky pro editaci konkrétního uživatele. Dále je pro realizaci dialogového okna využita komponenta `v-dialog` a standardní textová pole `v-text-field`. Upravená data uživatele jsou po uložení odeslána k uložení na server. Tento pohled využívá endpointy `user` (B.10), `user/save` (B.11) a `user/remove` (B.12).

Kalibrace (`ConfigCalibration`)

Pohled slouží ke kalibraci pH sond zařízení, a to ve dvou režimech. Prvním, je manuální mód, který je aktivován v případě, že není vybráno žádné kalibrační schéma v záhlaví tabulky. Tento mód umožňuje uživateli manuálně kalibrovat jednotlivé body a tím vytvořit kalibraci na míru konkrétním požadavkům. Druhou variantou jsou předdefinovaná kalibrační schémata, která umožňují kalibraci s použitím průvodce pro přidání jednotlivých bodů. Tento mód slouží zejména ke kalibraci na obvyklý pracovní rozsah hodnot pH.

Z technického hlediska se jedná o tabulku `v-data-table`, která je pomocí slotů upravena pro zobrazení buněk dle zvoleného módu. V obou režimech jsou zobrazeny hodnoty pH a napětí pro každý kalibrační bod. Avšak v případě manuálního módu je v daném sloupci u všech buněk zobrazeno i tlačítko pro kalibraci. Pohled využívá endpointy `calibration` (B.5 a `calibration/getRaw` (B.6).

Ladění (`ConfigDebug`)

Účelem této obrazovky je zobrazení ladících informací pro identifikaci problémů v zařízení. Obrazovka je rozdělena do tří částí obsahujících tabulky `v-data-table`, kde v první části jsou zobrazeny informace o stavech jednotlivých pump, senzorů teploty a pH. Ve druhé části obrazovky je pak zobrazena tabulka posledních běhů experimentů (jak privátních, tak veřejných) a jejich výsledků, aby bylo možné detekovat případný problém vzniklý při provádění experimentu. Poslední části je pak sekce pro zálohování a obnovení záloh, která je také realizována pomocí tabulky. Tento pohled využívá endpointy `debug` (B.7), `debug/backupCreate` (B.8) a `debug/backupRestore` (B.9).

Nastavení zařízení (`ConfigParams`)

Tato stránka slouží k zobrazení formulářů sloužících ke konfiguraci připojení přístroje Golem k Wifi síti a k synchronizaci času se zvoleným NTP serverem. Konstruce této stránky je poměrně přímočará, jelikož obsahuje pouze dvě komponenty `v-card`, skupiny textových polí `v-text-field` a tlačítka. Tento pohled využívá endpointy `params` (B.13), `params/sync` (B.14) a `params/Wifi`. (B.15).

Nedávné experimenty (`Recent`)

Pohled byl do systému přidán jako součást řešení problému, který byl odhalen při uživatelských testech. Tento problém spočíval v dlouhé trase skrz uživatelské rozhraní, kterou bylo nezbytné absolvovat před spuštěním běhu experimentu. Tento pohled poskytuje řešení tohoto problému tím, že zobrazuje nedávno použité experimenty v přehledné tabulce a tím zjednodušuje jejich výběr a zkracuje cestu pro spuštění nového běhu.

Jeho konstrukce je založena na tabulce `v-data-table`, která zobrazuje nedávné experimenty a seskupuje je dle účinné látky pomocí property `groupby`. Tabulka reaguje při kliknutí na řádek zachycením eventu `@click` a přesměrováním uživatele na stránku se zvoleným experimentem. Pohled využívá endpoint `recent` (B.4).

Seznam simulací (`SimulationOpen`)

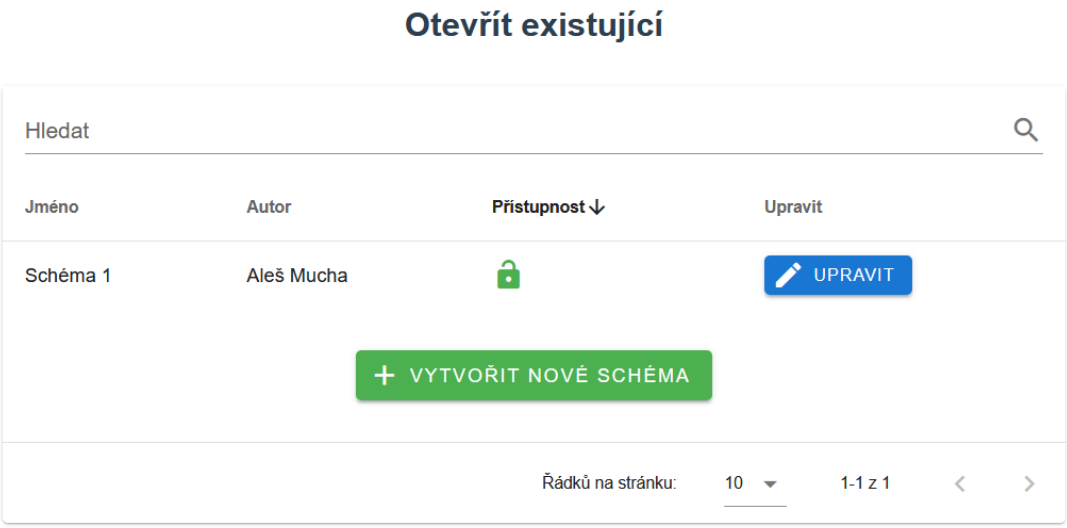
Obrazovka uvedená na obrázku 6.23 slouží k výběru konkrétního schématu, nebo vytvoření nového. Jejím základem je modifikovaná komponenta `v-data-table`, která je obsažena v knihovně Vuetify.

Modifikace spočívají například v nahrazení výchozího obsahu sloupce úprav za tlačítko pro editaci. Tohoto bylo dosaženo použitím tagu `template` a direktivy `v-slot:item.edit`, díky které byl dosazen vlastní kód pro tlačítko do všech buněk sloupce `edit`. Obdobným způsobem byl upraven i sloupec s indikací přístupnosti a přidáno tlačítko pro vytvoření nového schématu. Reakce na kliknutí na řádek tabulky je realizována pomocí zachytávání události `@click`, po kterém následuje přesměrování na seznam experimentů dané simulace (pohled `ExperimentOpen`). Pohled využívá endpoint `simulation/open` (B.16).

Editace simulace (`SimulationEdit`)

Stránka editace simulací slouží k zadávání a editaci klíčových parametrů simulace. Tato stránka je využita jak pro editaci, tak pro vytváření nových simulací.

Její realizace byla poměrně přímočará, jedná se zejména o vhodně rozmístěné komponenty `v-text-field`, kterým byla pomocí slotu `v-slot:append-inner` přidána podpora



Obrázek 6.23: **Obrazovka pro výběr a vytvoření simulace**

pro zobrazení jednoty potřebných veličin. Dále je použita komponenta `v-select`, která zde realizuje výběrové pole, `v-textarea`, která vytváří textové pole ve spodní kartě, `v-card` vytvářející jednotlivé sekce formuláře a komponenta `v-button` sloužící jako tlačítko pro uložení, reagující na event `@click`. Pohled využívá endpoint `simulation/edit` (B.17).

Parametry simulace (`SimulationParams`)

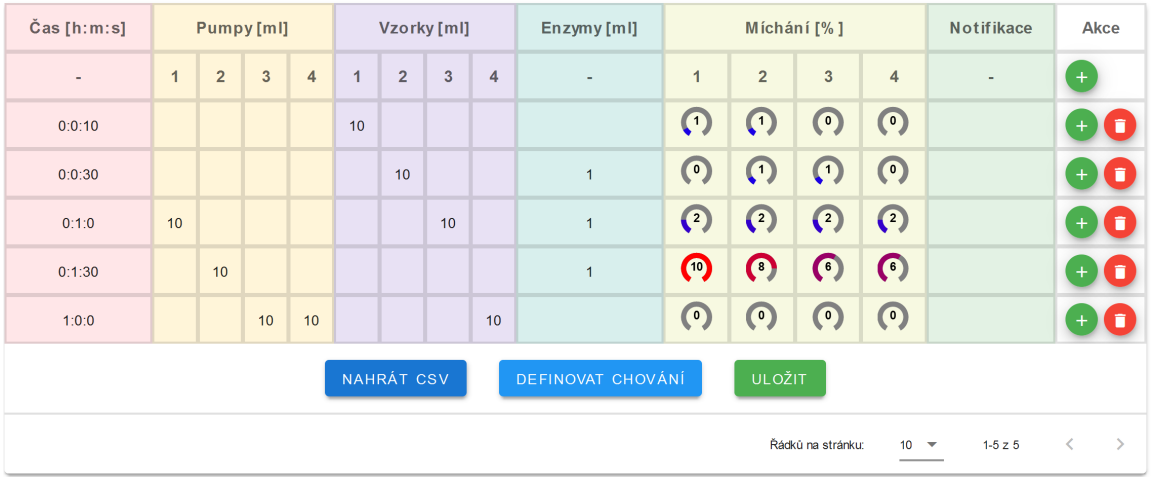
Tato stránka slouží k vytvoření seznamu akcí, které mají být v daných časech při běhu experimentu vykonány. Těmito událostmi mohou být čerpání, vstřikování enzymů, změny v rychlosti míchání a textová upozornění pro operátora.

Hlavním obsahem této stránky je tabulka vytvořená komponentou `v-data-table`, která je výrazně modifikována pomocí slotů. Sloty jsou zde využity k modifikaci všech sloupců tak, aby bylo možné do sloupců po kliknutí zapisovat data. Tohoto bylo docíleno dosazením komponenty `v-edit-dialog` do všech požadovaných buněk a propojení této komponenty s hodnotou dané buňky direktivou `return-value.sync`. Do komponenty `v-edit-dialog` je zanořena komponenta `v-text-field` využívající ke čtení a modifikacím hodnoty buňky direktivu `v-model`. Pohled využívá endpoint `simulation/parameters` (B.18).

Sloupec akcí obsahuje tlačítka vytvořená pomocí `v-btn` s reakcí na event `@click`. Tlačítka slouží k vytváření nových řádku a jejich mazání.

V zápatí tabulky se nachází další skupina tlačítek, které slouží k nahrávání předlohy ve formátu CSV, přístupu na stránku pro definici chování pomocí vizuálního programování a uložení.

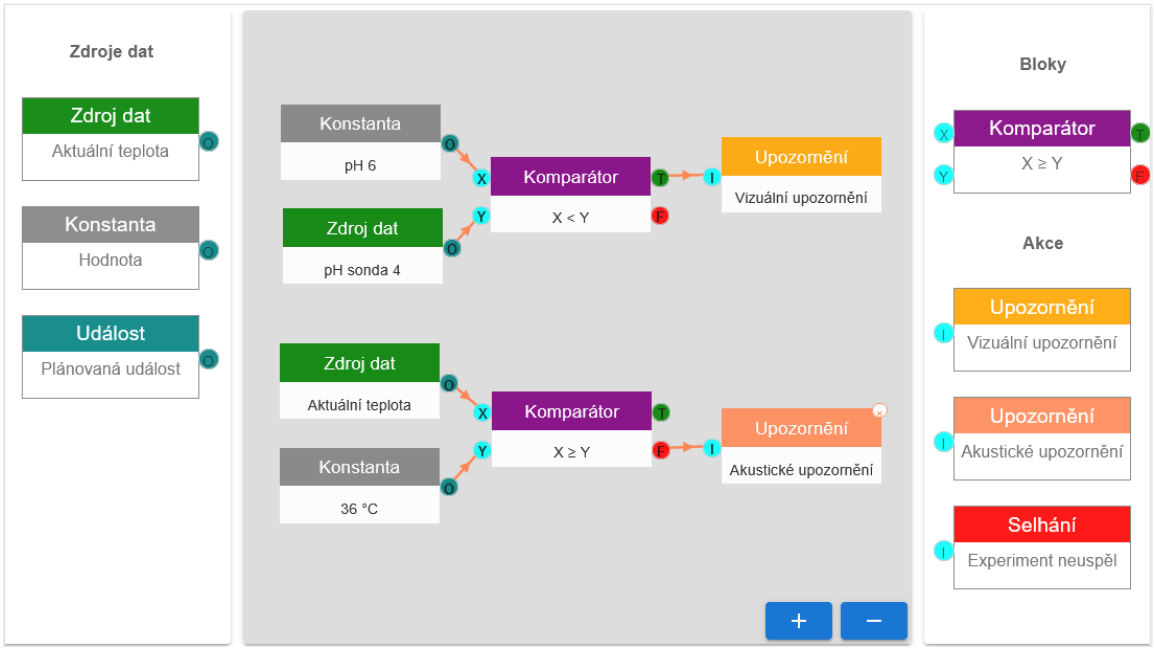
Oproti původnímu návrhu byla pro sloupce určené k nastavení rychlosti míchání použita komponenta `Knob`, která k zobrazení číselné hodnoty přidává i grafický prvek indikující rychlost míchání. Tento prvek byl přidán pro jednodušší orientaci v hodnotách sloupců.



Obrázek 6.24: **Obrazovka pro definici parametrů simulace**

Definice chování simulace (`SimulationBehavior`)

Jednou z nedůležitějších obrazovek celé aplikace je obrazovka pro vizuální specifikaci průběhu experimentu. Na této stránce, kterou můžete vidět na obrázku 6.25, je možné z připravených bloků vytvořit různá zapojení, která mohou být použita k automatickému hlídání průběhu experimentu a varování operátora na nevyhovující podmínky. Pohled využívá endpoint `simulation/behavior` (B.19).



Obrázek 6.25: **Obrazovka pro definici chování simulace**

Stránka je tvořena třemi sloupci, kdy v postranních sloupcích, sloužících jako nabídka, jsou umístěny funkční bloky a zdroje dat. Tyto bloky mohou být kliknutím přidány do středového sloupce, který slouží jako drag&drop editor. V editoru je možné pomocí metody táhni a pusť propojovat funkční bloky do větších celků a dvojklikem na funkční blok zobrazit

dialogové okno pro editace jejich parametrů. Ve spodní části editoru se nachází tlačítka pro zoomování editoru a ve spodní liště také tlačítko pro výběr existujícího diagramu chování z jiné simulace a tlačítko pro uložení.

Pro realizaci této stránky byla použita open-source knihovna vue-simple-flowchart, která umožňuje vytváření jednoduchých diagramů. Tato knihovna byla pro účely použití v tomto projektu značně modifikována přímou úpravou zdrojového kódu. Touto modifikací byla přidána podpora pro více vstupních a výstupních uzlů, konfigurace vlastností funkčních bloků pomocí dvojkliku, podpora pro spojování bloků v horizontálním módu a zoomování.

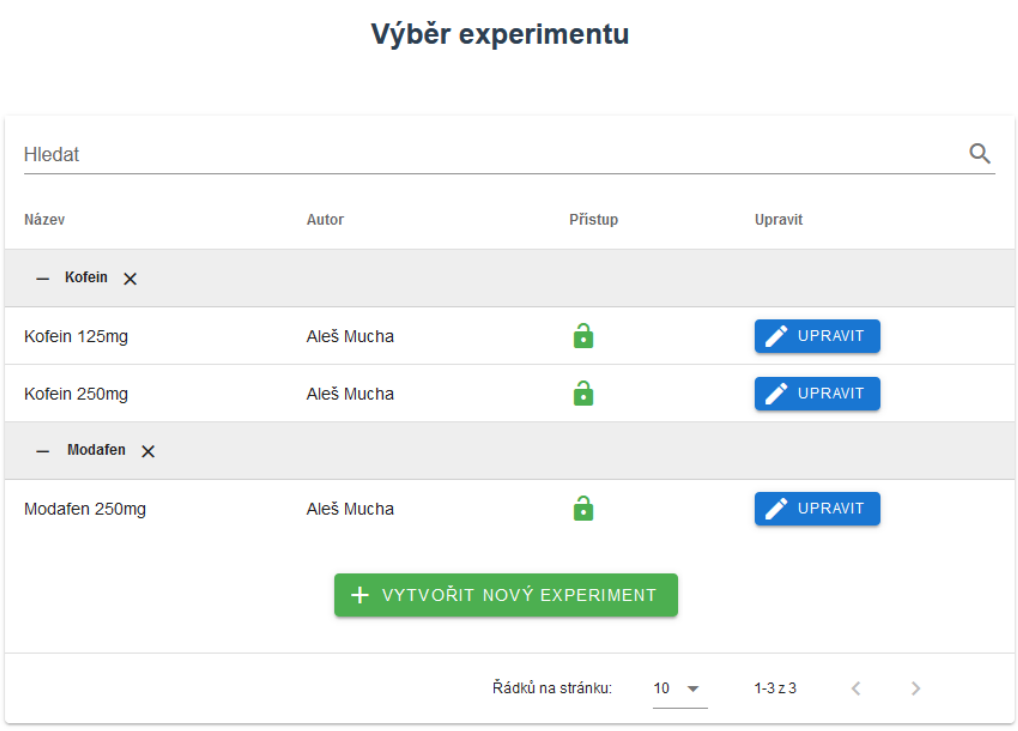
Většina úprav byla založena na modifikaci vnitřních datových struktur knihovny a přepsání metod obsluhujících danou část aplikace. Pro přidání reakcí na speciální události jako zoomování byly přidány nové eventy a handlery pro jejich zachytávání. Zoomování je realizováno pomoc obrazové transformace škálování CSS `transform: scale(x)`.

Výsledná modifikovaná knihovna byla veřejně zpřístupněna v repositáři[2] GitHub.

Seznam experimentů (`ExperimentOpen`)

Obrazovka seznamu experimentů (obrázek 6.26) slouží k výběru požadovaného experimentu, přidávání nových experimentů nebo přístupu k jejich editaci. Stejně jako stránka se seznamem simulací (6.1.5) využívá i tato téměř identicky modifikovanou komponentu `v-data-table`. Rozdílem je pouze seskupování experimentů podle účinné látky, které zjednodušuje orientaci.

Za tímto účelem byl použit parametr `group-by`, kterým se v komponentě `v-data-table` určuje sloupec sloužící jako klíč pro seskupování. Pohled využívá endpoint `experiment/edit` (B.20).



Obrázek 6.26: **Obrazovka pro výběr experimentu**

[2] https://github.com/xtruhl05/vue-simple-flowchart

Editace experimentu (`ExperimentEdit`)

Obdobně jako v případě editace schémat je tato obrazovka použita jak pro editaci experimentu, tak pro vytváření nového. Chování se odvíjí stejně jako v předchozím případě podle identifikátoru experimentu obsaženém v URL, který může nabývat pouze číselných hodnot v případě existujících experimentů a hodnoty `new` v případě nového.

V případě otevření existujícího experimentu je odeslán dotaz na API endpoint `/experiment/edit` (B.21) a je očekávána odpověď s daty formuláře, které jsou následně zobrazeny. V případě nového experimentu není tento dotaz proveden. Pro uložení dat je tento endpoint využit jako cíl POST dotazu s daty formuláře.

Seznam běhů (`RunOpen`)

Tato obrazovka slouží jako seznam všech již vykonaných běhů experimentu. Kliknutím na konkrétní řádek dojde k přesměrování na detaily výsledků zvoleného běhu, nebo je možné zahájit nový běh experimentu použitím tlačítka v zápatí tabulky.

Z technického hlediska se jedná, jako u předchozích stránek pro výběr simulace a experimentu, o modifikovanou komponentu `v-data-table`. Pohled využívá endpoint `run/open` (B.22).

Obrazovka pro sledování a řízení běhu experimentu (`RunExecute`)

Obrazovka pro řízení a sledování běhu experimentu je nejdůležitější obrazovkou celé aplikace a spojují se v ní data z většiny předchozích obrazovek. Jak její název napovídá, jejím účelem je dle konkrétních nastavení simulace a experimentu vykonat běh experimentu a umožnit uživateli během tohoto běhu kontrolovat a řídit průběh experimentu a jeho parametry. Komponenta využívá

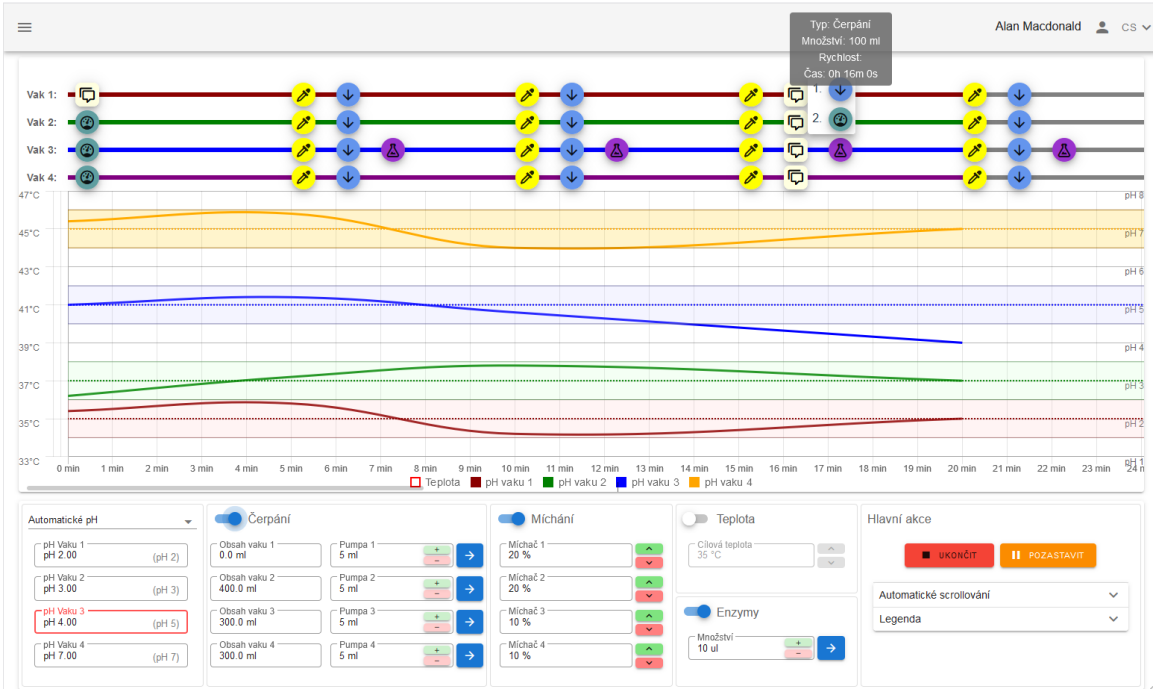
Tato obrazovka byla oproti původnímu návrhu značně změněna, jelikož uživatelské testy prováděné během jednotlivých iterací odhalily nedostatky původního návrhu. Výsledkem opakovaného testování a úprav je obrazovka, kterou můžete vidět na obrázku 6.27. Detailní informace o průběhu testování a jejich výstupech naleznete v kapitole 7.

Nové rozhraní je členěno do tří částí, kde první je sekce pro zobrazování časových os (timeline) a grafů nacházející se v levé horní části. Sekce potvrzení a notifikací je realizována jako postranní panel nacházející se na pravé straně. Poslední částí je panel ve spodní části obrazovky, který seskupuje všechny ovládací prvky pro řízení běhu experimentu a nastavení zobrazení.

Horní panel Sekce časových os a grafů, kterou můžete vidět na obrázku 6.16, slouží k vizuálnímu zobrazení klíčových veličin a událostí v průběhu experimentu. Tato část obrazovky je založena komponentě `ProgressSection`

Postranní panel Postranní panel byl doplněn jako náhrada za modální okna, která byla v původním návrhu použita pro potvrzovaní odběrů, enzymů a notifikací. Uživatelské testy (viz podkapitola 7.1) zde opět odhalily nedostatky původního návrhu, spočívající v překrývání části obrazovky a tím narušováním možnosti plnohodnotně řídit experiment. Z tohoto důvodu bylo vytvořeno alternativní rozhraní, které tyto nedostatky eliminuje.

Panel je tvořen v horní části tlačítky po potvrzování odběrů a vstřikování enzymů. Tlačítka jsou umístěna do mřížky a jejich aktivace událostí je doprovázena akustickým signálem



Obrázek 6.27: **Obrazovka pro řízení a sledování běhu experimentu**

a blikáním tlačítka. Akustický signál je vytvořen periodickým přehráváním zvukového souboru pomocí jazyka JavaScript, blikání je realizováno jako CSS animace umístěna v CSS třídě `blink`. Tato třída je při aktivaci tlačítku přidána a v případě potvrzení opět odebrána. Při potvrzení je rovněž provedena deaktivace zvukového upozornění a odesláno potvrzení na API endpoint `/run/execute` (B.24) metodou POST. Spodní část postranního panelu pak obsahuje prostý seznam notifikací. Každá notifikace je zobrazena jako oddělený řádek a obsahuje tlačítko pro odstranění.

Spodní panel Spodní panel pro řízení experimentu byl rovněž doplněn jako náhrada za modální okno obsahující prvky pro řízení průběhu experimentu. Hlavními důvody byly již zmíněné problémy s překrýváním obrazovky a také komplikovaný přístup k ovládacím prvkům. Panel je tvořen šesti sekcemi, kde každá poskytuje logicky oddělenou skupinu ovládacích prvků pro řízení určitého zařízení v přístroj. Ovládací prvky pro jednotlivé skupiny je potřeba před použitím aktivovat pomocí přepínače, nebo výběrového boxu v horní části. Ovládací prvky jsou skupiny textových polí (komponenta `ControlField`) a tlačítek (`v-btn`), které slouží buď ke změně hodnoty (tlačítka uvnitř textového pole se symboly + a -), změně a současnému odeslání (tlačítka se šipkami vně textové pole), nebo pouze odeslání (modrá tlačítka nacházející se vně textového pole). Výjimku zde tvoří sekce hlavních akcí, která obsahuje kromě tlačítek také expandující panely (komponenta `v-expand-panel`) obsahující ovládací prvky pro řízení automatického posunování (zaškrtávací políčko tvořené komponentou `v-simple-checkbox`) a legendu k časovým osám umožňující ovládání zobrazených událostí.

Výsledky experimentu (`RunResult`)

Poslední obrazovkou v této aplikaci je zobrazení výsledků běhu experimentu. Výsledná obrazovka je tvořena trojicí karet v horní části, které reprezentují klíčové údaje zděděné ze simulace a experimentu, na jejichž základě byl běh proveden. Třetí karta pak obsahuje údaje konkrétního běhu, jako je například autor, čas provedení a dosažená přesnost vykonání. Dále následují jednotlivé grafy zobrazující časové průběhy veličin během experimentu. K získání těchto dat využívá endpoint `result` (B.26), ke stahování jednotlivých souborů pak endpointy `downloadCSV` (B.27), `downloadPDF` (B.28) a `downloadZip` (B.29).

Oproti původnímu návrhu byla tato obrazovka rozšířena o dvě tabulky obsahující seznam všech akcí provedených během průběhu experimentu nad rámec původního plánu. Těmito akcemi jsou například vynucená čerpání a korekce pH pomocí ovládacích prvků předchozí obrazovky. Rovněž jsou zde zahrnuty akce provedené systémem automatického hlídání průběhu experimentu pomocí vizuálního programování. Druhou změnou je pak další přidaná tabulka sloužící jako seznam potvrzení odběrů a vstříknutí enzymů s časovými značkami. Tyto tabulky byly doplněny na základě požadavků na rozšíření vycházející z uživatelských testů.

Poslední částí obrazovky jsou pak tlačítka, která umožňují stažení souborů s výstupy experimentu. Dostupnými formáty jsou zde CSV (čárkami oddělené seznamy hodnot), dokument typu PDF a ZIP archív obsahující dokument i data. V zápatí je pak také tlačítko umožňující návrat do hlavní nabídky aplikace.

Přidání nového pohledu

Pro přidání nového pohledu (stránky aplikace) do tohoto systému je potřeba vytvořit nový soubor komponenty v adresáři `views`. Následně je potřeba přidat v adresáři `router` do souboru `index.js` řádek importující nově vzniklý pohled a zaregistrovat novou cestu k pohledu v poli `router` nacházející se v tomto souboru.

6.1.6 Překlady (translations)

Překlady celého uživatelského rozhraní jsou realizovaný pomocí rozšíření s názvem `i18n`. Tento systém umožňuje překlad systému s použitím zástupných řetězců, které jsou po volbě jazyka nahrazeny konkrétním textem.

Za tímto účelem byla vytvořena v adresářové struktuře projektu nová složka s názvem `translations`, která obsahuje jak soubory s překlady, tak soubor `languages.js`, který obsahuje jejich seznam. Tento seznam je následně použit komponentou Language, umístěnou v horním panelu aplikace, která slouží k volbě jazyka.

Příklad překladových souborů a jejich použití

Překladový systém je možné použít v rámci aplikace buď s pomocí speciálního zápisu pro umístění textu používaného Vue.js, tedy `{{výraz}}`, kde výraz bude nahrazen voláním metody pro překlad se specifikací požadovaného řetězce. Alternativně může být metoda pro překlad zavolána i z těl jiných metod, nebo použitím direktivy.

Příkladem obsahu souboru `languages.js`, který slouží jako seznam dostupných překladů může být následující kód. Jeho struktura je založena na formátu JSON a klíč každé položky se musí shodovat s názvem souboru pro překlad.

```
export default {
    en: "English",
    cs: "Czech",
}
```

Výpis 6.1: Soubor dostupných jazyků

Jako příklad překladového souboru mějme následující kód, který musí být pro správnou funkci systému umístěn v adresáři `translations` a pojmenován ve shodě s klíčem v předchozím souboru `en.js`.

```
config: {
    users: "Users",
    parameters: "Parameters",
    debug: "Debug",
    calibration: "Calibration",
},
configUsers: {
    title: "User management",
    new: "Add new user",
    edit: "Edit user",
    deleteWarning: "Do you really want to remove user?",
},
```

Výpis 6.2: Soubor obsahující překlady

Pro získání překladu požadované položky do aktuálně nastaveného jazyka (určeno proměnnou `$vuetify.lang.current`) je potřeba použít volání `{{$vuetify.lang.t('$vuetify.?')}}`, kde `?` musí být nahrazen konkrétní cestou k položce překladu. Příkladem může být `{{$vuetify.lang.t('$vuetify.config.calibration')}}`, jehož výstupem by byl v případě použití ukázkového kódu a proměnné `$vuetify.lang.current` nastavené na hodnotu „en“ řetězec „Calibration“.

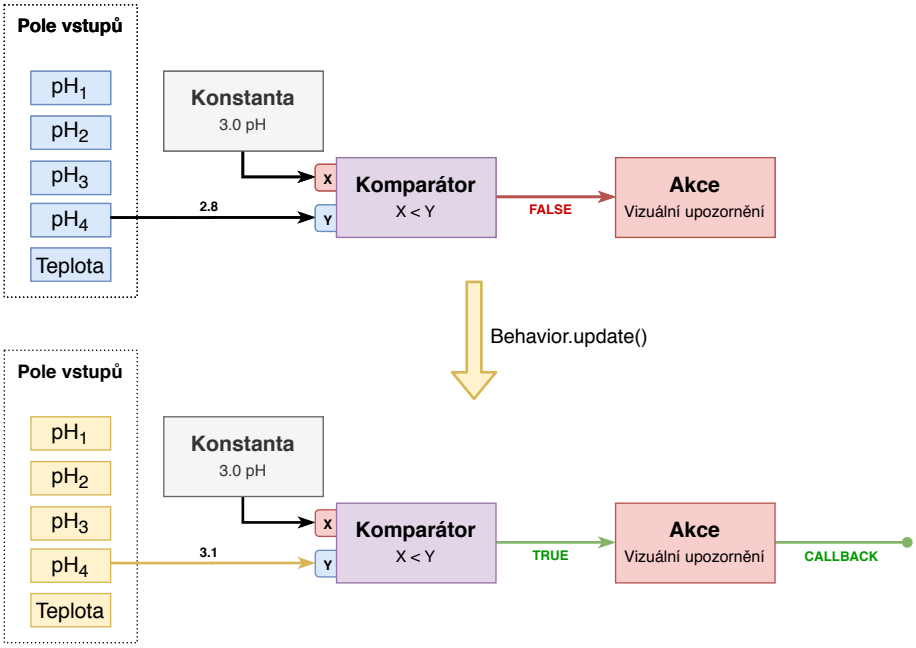
Přidání nového překladu

Pro přidání nového jazyka je potřeba vytvořit kopii již existujícího souboru s překladem a nahradit všechny obsažené texty novými v daném jazyce. Nově vytvořený překlad je poté nezbytné doplnit i do souboru `languages.js`.

6.1.7 Řízení průběhu laboratorních experimentů

Systém pro automatické řízení průběhu laboratorních experimentů pomocí vizuálního programování je založen jak na již popsaném pohledu `SimulationBehavior`, tak na nově vytvořené třídě `Behavior`, která má za účel zpracování změn klíčových parametrů a reakce na tyto změny.

Třída je, stejně jako zbytek systému, založena na jazyce JavaScript a ke svému fungování používá výstupy pohledu `SimulationBehavior`, ve kterém se nachází editor pro vizuální specifikaci řízení zvolené simulace. Na základě specifikace je při spuštění experimentu vytvořena stromová struktura objektů realizující jednotlivé bloky, které jsou vzájemně propojeny. S každou aktualizací dat obdrženou od zařízení (voláním metody `update()` dojde k propagaci těchto dat do všech vstupních bloků v rámci této struktury a opětovnému vyhodnocení jejich výstupních hodnot. Pokud se nově vypočítané hodnoty liší od původních, jsou předány



Obrázek 6.28: **Diagram principu funkce knihovny Behavior** – diagram znázorňuje původní stav systému a následné vyvolání aktualizace pomocí volání metody `update()`. Směr šipky naznačuje přechod systému do nového stavu a žlutě jsou vyznačeny změněné hodnoty. Dále je zde znázorněno šíření změn mezi funkčními bloky, až po vyvolání reakce pomocí callback metody

všem cílovým blokům. Tento proces aktualizace a propagace dat se provádí až do chvíle, kdy data dorazí do koncových bloků (akce). Výstupní bloky následně v případě změny výstupní hodnoty provedou zpětné volání (callback) přiřazené akce. Tímto dojde k vykonání předem definované reakce na změnu stavu experimentu. Toto chování je graficky znázorněno na obrázku 6.28.

6.2 Propojení se serverovou částí

Propojení se serverovou částí bylo realizováno jako REST API (Representational State Transfer Application Interface) založené na jednoduchých HTTP voláních, která byla postavena na základě frameworku Express. Jako komunikační formát byl vzhledem k použití JavaScriptových frameworků zvolen nativně podporovaný JSON a indikace stavů pomocí standardních HTTP kódů. Struktura API a datové typy byly definovány pomocí typové nadstavby TypeScript. Detailní popis API a datových typů je obsažen v příloze B.

6.2.1 Stavové kódy

Pro indikaci různých stavů API bylo zvoleno standardní použití stavových kódu protokolu HTTP. V tomto projektu však byla použita pouze jejich omezená množina, která je shrnuta v následujícím seznamu:

- HTTP 200 OK – Standardní odpověď všech endopointů v případě úspěšného vykonání dotazu.

- HTTP 301 Moved Permanently – Tento stavový kód spolu s novou cílovou adresou je použit v případě, že je potřeba uživatele přesměrovat na jinou adresu.
- HTTP 401 Unauthorized – Neautorizovaný přístup k endpointu je oznámen tímto stavovým kódem.
- HTTP 404 Not Found – Tímto kódem je oznámeno buď nenalezení konkrétního záznamu, nebo neexistující endpoint.
- HTTP 500 Internal Server Error – Tento stavový kód API odešle v případě, že došlo k jejímu selhání.

6.2.2 Autentizace uživatele

Pro účely autentizace uživatele a jeho spojení s konkrétní uživatelskou relací byl použit unikátní automaticky generovaný UUID token, který je vytvořen po úspěšném přihlášení uživatele do systému a uložen jak na straně serveru, tak na straně uživatele v podobě cookie s názvem sessionId.

Tento token je následně odesílán při každém volání jako položka hlavičky požadavku s názvem sessionId. V případě, že token není vyplněn, nebo není platný, reaguje API stavovým kódem 401 Unauthorized a uživatelské rozhraní uživatele automaticky přesměruje na přihlašovací obrazovku.

6.2.3 Struktura API

Struktura API odpovídá struktuře cest pro navigaci v uživatelském rozhraní, které je uvedeno v podkapitole 5.2, struktura je však doplněna o nové cílové adresy (endpointy) sloužící pro různé typy volání. Výchozím formátem těl dotazů i odpovědí je JSON a výjimku tvoří pouze přenosy souborů.

Detailní popis struktury API a vytvořených datových typů naleznete v příloze.

6.2.4 Testovací API

Jelikož implementace nového uživatelského rozhraní probíhala před dokončením řídící jednotky a vytvořením reálného aplikačního rozhraní, bylo nezbytné nad rámec zadání vytvořit testovací API, která by dokázala dostatečně realisticky simulovat chod zařízení Golem. API byla vytvořena pomocí frameworku Express a byla naplněna statickými daty pro účely testování.

Postupným rozvojem uživatelského rozhraní byla API dále rozšířena o možnost ukládání dat a vytváření uživatelských relací. Poslední fází bylo doplnění generátoru dat, které umožnily simulovat běh experimentu. Generátor vytvářel jak testovací výstupy senzorů teploty a pH, tak i simuloval běh experimentu a vznik událostí.

6.2.5 Reálná API

Reálná API byla následně vytvořena na základě testovací API, která byla Bc. Zbyškem Vodou upravena tak, aby pracovala s interní databází zařízení, senzory a firmware.[24] Možnost generování testovacích dat byla zachována i v této verzi aplikačního rozhraní, aby bylo možné vyvíjet a testovat uživatelské rozhraní i bez nutnosti připojení k fyzickému zařízení.

Kapitola 7

Testování

Uživatelské testování systému bylo prováděno ve formě navazujících iterací během celého procesu vývoje a výsledky testování byly následně použity k úpravě a vylepšení stávajícího rozhraní. V okamžiku přechodu z rozhraní se statickými daty na verzi pracující s dynamickými daty získanými z testovací API do procesu testování vstoupilo také integrační testování pro ověření správného napojení na jednotlivé endpointy. Po dokončení poslední iterace bylo přistoupeno k závěrečnému akceptačnímu testování a vytvoření akceptačního protokolu.

7.1 Uživatelské testy

Jelikož se jedná o software vyvíjený na míru konkrétní skupině uživatelů, zvolil jsem jako primární způsob testování, stejně jako při návrhu, uživatelské testy prováděné zejména se členy výzkumné skupiny. Při těchto testech byly uživatelé požádání, aby v uživatelském rozhraní provedli různé akce uvedené ve funkčních požadavcích.

Při provádění těchto úkolů byli uživatelé sledováni a studovány jejich reakce. Uživatelé byli rovněž požádání o to, aby nahlas sdělovali své myšlenky, pocity a nápady, které byly zaznamenávány ve formě poznámek.

Tímto způsobem byla v prototypu systému odhalena řada špatně navržených ovládacích prvků a stránek, kde měli uživatelé problém s pochopením jejich významu a použití. Postupným řešením těchto problémů a prováděním dalších iterací byl systém optimalizován tak, aby uživatelům poskytoval srozumitelné uživatelské rozhraní obsahující všechny ovládací prvky potřebné pro efektivní provádění experimentů.

7.1.1 Identifikované problémy

Během iterací bylo odhaleno poměrně velké množství menších nedostatků. Typickým příkladem bylo špatné rozmístění prvků na stránce a nepřesně specifikované požadavky na funkce dané obrazovky.

Dále bylo rozhraní při iteracích rozšířeno o velké množství nových vstupů a ovládacích prvků, které v původním návrhu rozhraní nebyly zahrnuty. Tato rozšíření byla vytvořena pro umožnění přesnější specifikace a řízení experimentu, nebo pro zjednodušení ovládání systému.

Komponenta Knob

Původně byla tato komponenta vytvořena tak, aby uživateli umožňovala měnit hodnotu pomocí kliknutí, nebo umístěním kurzoru nad komponentu a otáčením kolečka myši. Tento způsob ovládání se však ukázal jako nevhodný, jelikož nepočítal s použitím dotykových polohovacích zařízení, na kterých je kolečko myši nahrazeno gesty, která mohou být pro uživatele komplikované. Druhým problémem pak bylo vertikální posunování (scrollování) v rámci stránky, které mohlo vést k neúmyslné změně hodnoty.

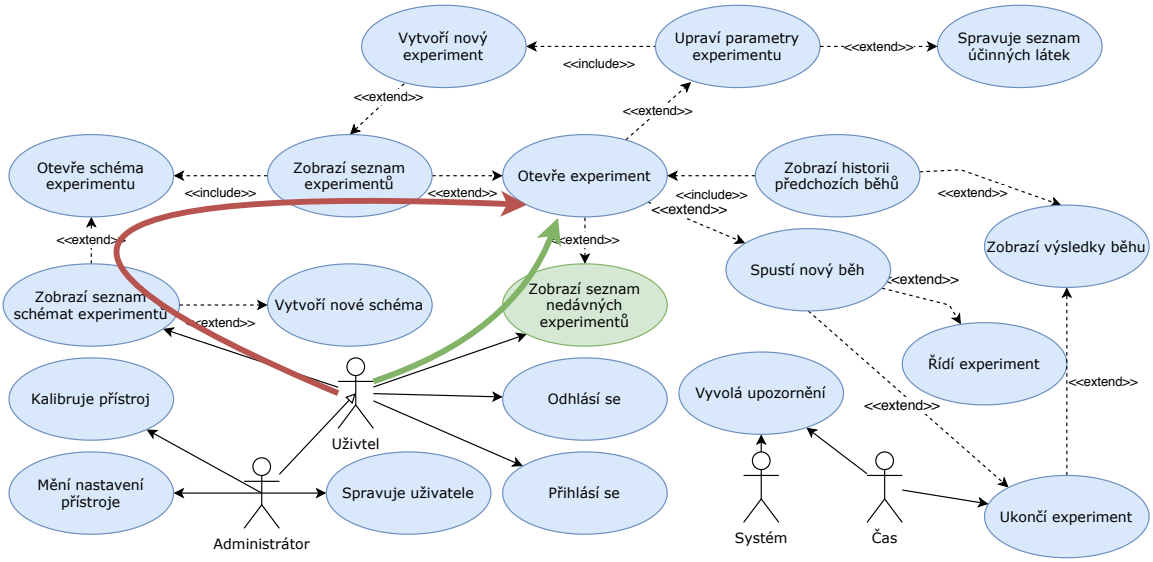
Pro vyřešení tohoto problému byla možnost použití kolečka myši pro změnu hodnoty odstraněna a bylo zachováno pouze zadávání hodnoty skrz textové pole, které se zobrazí po kliknutí na komponentu.

Léky a lékové formy

Při uživatelských testech bylo zjištěno, že stránka pro vytváření experimentů by měla být rozšířena o ovládací prvky pro výběr a správu léků a lékových forem. K tomuto rozšíření bylo přistoupeno pro zjednodušení zadávání a lepší organizaci experimentů, kde by zadávání skrze textové pole vedlo na méně ergonomické rozhraní a možnost vzniku překlepů či jiných chyb.

Zdlouhavý výběr experimentu

Dalším odhaleným nedostatkem byla pro uživatele příliš dlouhá trasa pro výběr konkrétního experimentu. Uživatelé museli pro výběr experimentu přejít z obrazovky hlavního menu do seznamu simulací, z tohoto seznamu pak na seznam experimentů a až na této stránce bylo možné spustit nový běh experimentu. Tento problém je vyznačen v následujícím diagramu případů užití.



Obrázek 7.1: **Diagram případů užití s vyznačenými trasami** – červeně je vyznačena původní trasa, zeleně pak doplněná zkratka.

Pro řešení tohoto nedostatku bylo v hlavní nabídce nahrazeno tlačítka pro vytvoření nové simulace tlačítkem pro přechod na stránku se seznamem nedávných experimentů, které

odkazuje na nově přidanou obrazovku obsahující seznam nedávno použitých experimentů. Z této obrazovky je pak možné výběrem z tabulky přejít přímo ke konkrétnímu experimentu.

Výpočetní náročnost grafů

Výpočetní náročnost grafů se ukázala jako problém v závěrečné fázi testování, kdy byly provedeny experimenty trvající v řádu hodin. Při těchto experimentech bylo zjištěno, že implementace knihovny Chart.js není optimalizována na vykreslování desítek tisíc bodů, což vede na pomalé vykreslování a vysokou spotřebu paměti.

Jako řešení se zde nabízí shlukování bodů v historii dané řady dat do skupin o určité velikosti (například 10 bodů) a jejich nahrazení bodem s hodnotou odpovídající průměru hodnot bodů ve skupině. Nevýhodou tohoto řešení by však bylo snížení přesnosti grafů zobrazených během průběhu experimentu. Alternativou může být úprava knihovny pro účely použití v tomto projektu, což však může být komplikované a znemožnit aktualizace knihovny.

Vzhledem k odhalení této chyby během závěrečných testování a její nižší závažnosti, bude její odstranění provedeno během instalace nového uživatelského rozhraní a řídící jednotky do přístroje GOLEM, které je plánováno na přelom září a října tohoto roku (2020).

7.2 Integrační testování

Po dokončení implementace rozhraní se statickými daty bylo přikročeno k postupné implementaci endpointů testovacího aplikačního rozhraní a jejich integraci do systému. Během tohoto procesu bylo manuálně ověřováno dodržení formátu dat a správnost vytvářených požadavků a odpovědí. Tímto způsobem bylo celé rozhraní po jednotlivých částech rozšiřováno a testováno, až do jeho úplného dokončení.

V následující fázi integračních testů byly k testování přizváni uživatelé, kteří prováděli typické činnosti, jako je otevírání experimentů, změny ve formulářích a jejich následné ukládání. Díky tomuto postupu a zaznamenávání chyb na straně serveru i prohlížeče byly získány informace o chybách v integraci, které byly následně odstraněny. Tímto postupem bylo odhaleno několik chyb v integraci, které zabraňovaly uložení některých dat ve formulářích, nebo jejich načtení. Většina těchto chyb byla způsobena buď špatným pojmenováním záznamů v rámci přijímaných nebo odesílaných datových struktur, nebo jejich chybným zpracováním.

Poslední fáze integračních testů následovala po zapracování aplikačního rozhraní do firmware vyvíjeného Bc. Zbyškem Vodou a stejně jako v předchozích případech, byla i tato fáze založena na sledování chybových hlášení na straně serveru i klienta a manuálním vyhodnocování správnosti obdržených dat. Během této fáze bylo opět nalezeno určité množství chyb, které vycházely ze špatné integrace obou systémů. Tyto chyby byly rovněž následně odstraněny.

7.3 Akceptační testování

Akceptační testování představuje závěrečnou sadu testů systému prováděných se zadavatelem práce, kterého zde zastupoval PharmDr. Martin Čulen, Ph.D. Během tohoto testování bylo ověřeno, že výsledná aplikace splňuje všechny funkční požadavky, které byly stanoveny během návrhu tohoto projektu. Akceptační testování této práce proběhlo dne 23.7.2020 a výsledný akceptační protokol je obsažen v příloze této práce.

Kapitola 8

Závěr

V této práci byl proveden návrh, realizace a testování nového uživatelského rozhraní pro řízení laboratorního přístroje GOLEM, vyvíjeného Veterinární a farmaceutickou univerzitou v Brně, který slouží k výzkum vstřebávání léčiv v lidské trávicí soustavě.

Pro usnadnění pochopení přístroje GOLEM, jeho účelu a způsobu použití je v této práci uvedena kapitola věnující se základům farmakokinetiky a lidské trávicí soustavě, jejíž části tento přístroj simuluje. Dále je pak provedena detailní analýza přístroje GOLEM, jeho principu použití, součástí, komunikačního protokolu a aplikace pro jeho řízení. Na základě předchozích částí práce byl, v úzké spolupráci se zadavatelem, proveden návrh inovovaného uživatelského rozhraní a také výběr technologií pro realizaci. V této části je také uveden popis zvolených technologií a základy jejich použití. Následující část práce je věnována implementaci a dokumentaci nového modulárního uživatelského rozhraní. Poslední část práce je věnována procesu testování, vyhodnocování a oprav nalezených nedostatků, které bylo v konečné fázi následováno akceptačním testováním díla se zadavatelem, jehož výsledkem bylo přijetí bez výhrad. Akceptační testování je doloženo protokolem v příloze této práce.

Nad rámec zadání byl do systému, na přání zadavatele, přidán internacionalizační systém a také vytvořena specifikace aplikačního rozhraní pro komunikaci s řídící jednotkou. Na základě této specifikace bylo vytvořeno testovací rozhraní, které simuluje běh zařízení a umožňuje testování rozhraní bez přístroje GOLEM. Tento systém byl následně zakomponován do firmware řídící jednotky, kterou ve své diplomové práci vyvinul Bc. Zbyšek Voda.

Rovněž byla se zadavatelem naplánována následující fáze spolupráce na vývoji tohoto zařízení, během které bude provedena ve spolupráci s Bc. Zbyškem Vodou instalace nové řídící jednotky a uživatelského rozhraní. Dále bude také opraven problém s výpočetní náročností grafů při dlouhých experimentech, který je způsoben velkým počtem bodů. Během této spolupráce bude rozhraní dále optimalizováno a také odstraněny případné nedostatky uživatelského rozhraní a řídící jednotky, které by mohly být odhaleny při testování v reálných podmínkách. Předpokládaný termín je stanoven na přelom září a října tohoto roku (2020).

Celkově považuji práci za úspěšnou a všechny body zadání za splněné.

Literatura

[1] *FlaTrode Specification Sheet*. 4970 Energy Way Reno, NV 89502 U.S.A: Hamilton Company. Dostupné z: https://www.hamiltoncompany.com/process-analytics/product-specs/49676.

[2] Operating Manual Cavro ® Smart Valve Plus. Tecan. 2006, October.

[3] Operator's Manual Model XE 1000 Pump. Tecan. 2009, October.

[4] *Vita Universitatis: časopis Veterinární a farmaceutické univerzity Brno*. Brno: Veterinární a farmaceutická univerzita Brno, 2011. Dostupné z: https://www.vfu.cz/files/vu_2011_5.pdf.

[5] Vytvorenie metódy pre dynamickú biorelevantnú disolúciu na prístroji Golem v2. In: *Konference Interní grantové agentury VFU Brno*. 1. vyd. Brno: Veterinární a farmaceutická univerzita Brno, 2018, s. 208–211. ISBN 978-80-7305-811-1.

[6] *PT-1000*. Atlas Scientific 43-15 11th Street Long Island City, NY 11101: Atlas Scientific, 2019. Dostupné z: https://www.atlas-scientific.com/_files/_datasheets/_probe/PT-1000-probe.pdf.

[7] *JsPlumb Toolkit*. JSPLUMB UK, 2020. Dostupné z: https://docs.jsplumbtoolkit.com/toolkit/current/index.html.

[8] *NATIONAL INSTRUMENTS: LabVIEW*. NATIONAL INSTRUMENTS, 2020. Dostupné z: https://www.ni.com/cs-cz/shop/labview/labview-details.html.

[9] *Node-RED: Documentation*. JS Foundation, 2020. Dostupné z: https://nodered.org/docs/.

[10] *Rete.js*. Vitaliy Stoliarov, 2020. Dostupné z: https://rete.js.org/#/.

[11] *Total.js: Flow*. Total Avengers, 2020. Dostupné z: https://www.totaljs.com/flow/.

[12] *Encyklopedie SÚKL: Co je to léková forma a jaké druhy jsou?* Praha: SÚKL, c2001-2008. Dostupné z: http://www.olecich.cz/encyklopedie/co-je-to-lekova-forma-a-jake-druhy-jsou.

[13] *Arduino Uno*. Arduino, c2020. Dostupné z: https://store.arduino.cc/arduino-uno-rev3.

[14] *TriContinent Catalog: C-Series Syringe Pumps*. Kipseli: Tecan, c2020. Dostupné z: https://hoseforce.gr/upload/TriContinent-catalog-pages.pdf.

[15] *Vue.js: The Progressive JavaScript Framework*. © 2014-2020. Dostupné z: https://vuejs.org/.

[16] Bendová, M. *Základy farmakokinetiky, Základy farmakodynamiky, Terapeutické monitorování hladin léčiv a metody stanovení*. Brno: Úsek klinické farmacie NL FN Brno, 2017. Dostupné z: https://is.muni.cz/el/med/jaro2017/KBSM/um/Bendova_Zaklady_farmakokinetiky.pdf.

[17] Brno, V. a farmaceutická univerzita. *Dokumentace GOLEM II*. 2009.

[18] Google. *Material Design*. Mountain View, California: Google, 2020. Dostupné z: https://material.io.

[19] Heimgärtner, R. *Intercultural User Interface Design*. 1. Springer, Cham: Springer Nature Switzerland AG 2019, 2019. ISBN 978-3-030-17426-2.

[20] Hemza, J. a Hanzlová, J. *Základy anatomie soustavy trávicí, žláz s vnitřní sekrecí a soustavy močopohlavní*. Brno: Fakulta sportovních studií, Masarykova univerzita, 2013. Dostupné z: https://is.muni.cz/do/fsps/e-learning/zaklady_anatomie/zakl_anatomie_II/pages/brisni_panevni_cast.html.

[21] Krug, S. *Don't make me think, revisited: a common sense approach to web usability*. [Berkeley]: New Riders, [2014]. ISBN 978-0321965516.

[22] Macrae, C. *Vue.js : up and running : building accessible and performant web apps*. Sebastopol: O'Reilly, 2018. ISBN 978-1-491-99724-6.

[23] Stupák, I., Pavloková, S., Vysloužil, J., Dohnal, J. a Čulen, M. Optimization of Dissolution Compartments in a Biorelevant Dissolution Apparatus Golem v2, Supported by Multivariate Analysis. *Molecules*. 2017, roč. 22, č. 12. Dostupné z: http://www.mdpi.com/1420-3049/22/12/2042. ISSN 1420-3049.

[24] Voda, Z. *Vestavěná řídicí jednotka pro ovládání laboratorního zařízení*. Brno, CZ, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22339/.

[25] Vuetify. *Vuetify: Material Design Component Framework*. Fort Worth, Texas: Vuetify, 2020. Dostupné z: https://vuetifyjs.com/.

[26] Čulen, M., Tuszyński, P. K., Polak, S., Jachowicz, R., Mendyk, A. et al. Development of In Vitro - In Vivo Correlation/Relationship Modeling Approaches for Immediate Release Formulations Using Compartmental Dynamic Dissolution Data from "Golem": A Novel Apparatus. *BioMed Research International*. 2015, roč. 2015, č. 328628, s. 1–13. Dostupné z: http://www.hindawi.com/journals/bmri/2015/328628/. ISSN 2314-6133.

Příloha A

Obsah elektronické přílohy a postup spuštění

A.1 Obsah elektronické přílohy

- latex – zdrojové soubory technické zprávy
- dokumentace – články a dokumentace GOLEM
- aplikace – zdrojové kódy aplikace
  - frontend – zdrojové kódy uživatelského rozhraní
    - vue-simple-flowchart - zdrojové kódy upravené[1] knihovny
  - backend – zdrojové kódy firmware vytvořené Bc. Zbyškem Vodou
- pdf – výsledné PDF soubory

A.2 Postup spuštění aplikace

1. Ověřit, že porty TCP 8888 (web) a 3000 (API) nejsou využity žádnou jinou aplikací.
2. Ověřit, že je nainstalováno npm.
3. Vstoupit do složky aplikace.
4. Spustit skript `run.sh` a vyčkat na dokončeni instalace a spuštění.
5. Přistoupit skrz webový prohlížeč na `http://localhost:8888/`

[1] https://github.com/xtruhl05/vue-simple-flowchart

Příloha B

Aplikační rozhraní a jeho datové typy

Tato příloha slouží jako technická dokumentace popisující jednotlivé endpointy REST API, která byla v rámci tohoto projektu vytvořena jako testovací API a následně do finální podoby upravena Bc. Zbyškem Vodou, který její strukturu využil jako základ pro realizaci API sloužícího ke komunikaci mezi řídící jednotkou a uživatelským rozhraním vytvořeným v této práci.

Všechny endpointy mohou kromě uvedených návratových kódů v následujících tabulkách ohlásit také HTTP kód 500 v případě interní chyby, nebo kód 401 v případě neplatného autentizačního tokenu v položce hlavičky dotazu s názvem sessionId.

B.1 Koncové body API

/login

Jedná se o endpoint pro získání autentizačního klíče pro komunikaci s API. Tento endpoint slouží jako vstupní brána do systému a musí být úspěšně použit jako první. V opačném případě všechny ostatní endpointy reagují na dotazy stavovým kódem 401 - Unauthorized. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.1.

Cíl:	/login		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST	`username: string,` `password: string`	200	`state: string,` `sessionId: string`
		405	`state: string`

Tabulka B.1: **Deklarace API endpointu login**

/getUserData

Jedná se o endpoint pro získání základních informací o přihlášeném uživateli na základě autentizace pomocí tokenu obsaženého v hlavičce sessionId. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.2.

Cíl:	/getUserData		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`userData: User (T.14)`

Tabulka B.2: **Deklarace API endpointu getUserData**

/logout

Volání tohoto endpointu způsobí ukončení uživatelské relace (vymazání sessionId) a odhlášení uživatele ze systému. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.2.

Cíl:	/logout		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST		200	`state: string`

Tabulka B.3: **Deklarace API endpointu logout**

/recent

Endpoint recent slouží k získání seznamu nedávno použitých experimentů. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.4.

Cíl:	/recent		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`experiments: Experiments (T.29),` `drugs: Drugs (T.45)`

Tabulka B.4: **Deklarace API endpointu recent**

/config/calibration

Tento endpoint slouží k získání veškerých dat potřebných pro fungování obrazovky pro kalibraci (`ConfigCalibration`). Součástí odpovědi jsou datové struktury obsahující informace o kalibraci sond. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.5.

Cíl:	/config/calibration		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`calibration: Calibration` (T.2)
POST	`calibration: Calibration` (T.2)	200	

Tabulka B.5: **Deklarace API endpointu calibration**

/config/calibration/getRaw

Tento endpoint slouží k získání aktuální hodnoty napětí z pH sondy. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.5.

Cíl:	/config/calibration		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`raw: number, voltage: number`

Tabulka B.6: **Deklarace API endpointu calibration**

/config/debug

Tento endpoint slouží k získání veškerých dat potřebných pro fungování obrazovky pro ladění (`ConfigDebug`). Součástí odpovědi jsou datové struktury obsahující informace o stavech zařízení, historii prováděných experimentů a seznam záloh systému. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.7.

Cíl:	/config/debug		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`debug: Debug` (T.5)

Tabulka B.7: **Deklarace API endpointu debug**

/config/backupCreate

Tento endpoint slouží k vytvoření nové zálohy systému. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.8.

Cíl:	/config/debug/backupCreate		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST		200	

Tabulka B.8: **Deklarace API endpointu debug**

/config/debug/backupRestore

Tento endpoint slouží k navrácení systému do stavu z vybrané zálohy. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.9.

/config/users

Tento endpoint slouží k získání seznamu všech uživatelů. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.10.

Cíl:	/config/debug/backupRestore		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST		200	`backupId: Number`

Tabulka B.9: **Deklarace API endpointu debug**

Cíl:	/config/users		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`users: User[]` (T.14)

Tabulka B.10: **Deklarace API endpointu users**

/config/users/save

Účelem tohoto endpointu je přijmout upravený objekt uživatele a uložit jej. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.11.

Cíl:	/config/users/save		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST	`user: User`	200	

Tabulka B.11: **Deklarace API endpointu save**

/config/users/remove

Účelem tohoto endpointu je odstranit záznam uživatele z interní databáze. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.12.

Cíl:	/config/users/remove		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST	`user: User`	200	

Tabulka B.12: **Deklarace API endpointu remove**

/config/params

Tento endpoint slouží k získání objektu obsahujícího nastavení systému. Tento objekt je využit pohledem `ConfigParams`. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.13.

/config/params/sync

Tento endpoint slouží k provedení synchronizace zařízení se zvoleným NTP serverem. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.14.

Cíl:	/config/params		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`params: Params[]` (T.10)

Tabulka B.13: **Deklarace API endpointu /config/params**

Cíl:	/config/params/sync		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST	`NTPserver: String`	200	GET

Tabulka B.14: **Deklarace API endpointu /config/params/sync**

/config/params/wifi

Tento endpoint slouží k uložení nastavení WiFi. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.14.

Cíl:	/config/params/wifi		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST	`wifi: WiFi (T.12)`	200	GET

Tabulka B.15: **Deklarace API endpointu /config/params/wifi**

/simulation/open

Tento endpoint slouží k získání sezmanu simulací z interní databáze. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.16.

Cíl:	/simulation/open		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST		200	`simulations: Simulation[]` (T.16)

Tabulka B.16: **Deklarace API endpointu pro získání seznamu simulací**

/simulation/:simulationId/edit

Tento endpoint slouží k získání dat aktuální simulace v případě dotazu typu GET a pro jejich uložení v případě POST. Pokud uložená simulace má `:simulationId` nastaveno na hodnotu `new` (jedná se o novou simulaci), je vytvořena nová simulace a její nově přiřazený identifikátor je odeslán v těle odpovědi. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.17.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`settings: SimulationSettings` (T.17)
POST	`settings: SimulationSettings` (T.17)	200	`simulationId: Number`

Tabulka B.17: **Deklarace API endpointu /simulation/:simulationId/edit** /simulation/:simulationId/edit

/simulation/:simulationId/parameters

Tento endpoint slouží k získání a uložení parametrů simulace. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.18.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`parameters: SimulationParameters` (T.22)
POST	`parameters: SimulationParameters` (T.22)	200	

Tabulka B.18: **Deklarace API endpointu /simulation/:simulationId/edit** /simulation/:simulationId/edit

/simulation/:simulationId/behavior

Tento endpoint slouží k získání a uložení dat specifikace chování experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.17.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`behavior: SimulationBehavior` (T.18)
POST	`behavior: SimulationBehavior` (T.18)	200	

Tabulka B.19: **Deklarace API endpointu /simulation/:simulationId/edit** /simulation/:simulationId/behavior

/simulation/:simulationId/experiment/open

Tento endpoint slouží k získání seznamu všech experimentů dané simulace. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.20.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`experiments: SimulationExperiment[]` (T.29) `drugs: Drugs[]` (T.45)

Tabulka B.20: **Deklarace API endpointu pro získání seznamu experimentů** /simulation/:simulationId/experiment/open

/simulation/:simulationId/experiment/:experimentId/edit

Tento endpoint slouží k získání dat, editaci a vytváření nového experimentu.K vytvoření nového experimentu dojde při odeslání požadavku s `:experimentId` nastaveného na hodnotu `new`. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.21.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`settings: ExperimentSettings[]` (T.30) `drugs: Drugs[]` (T.45)
POST	`settings: ExperimentSettings[]` (T.30)	200	

Tabulka B.21: **Deklarace API endpointu pro editaci experimentu** /simulation/:simulationId/experiment/:experimentId/edit

/simulation/:simulationId/experiment/:experimentId/run/open

Tento endpoint slouží k získání seznamu běhů experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.22.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`runs: ExperimentnRun[]` (T.31)

Tabulka B.22: **Deklarace API endpointu pro získání seznamu běhů** /simulation/:simulationId/experiment/:experimentId/run/open

/simulation/:simulationId/experiment/:experimentId/run/:runId/create

Tento endpoint slouží k vytvoření nového běhu experimentu a získání jeho id. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.23.

/simulation/:simulationId/experiment/:experimentId/run/:runId/execute

Tento endpoint slouží k získání dat nově vytvořeného běhu experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.24.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
POST		200	`runId: Number`

Tabulka B.23: **Deklarace API endpointu pro vytvoření běhu experimentu** /simulation/:simulationId/experiment/:experimentId/run/:runId/create

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**

GET		200	`run: ExperimentRun (T.31)`
			`timelines:Timeline[4] (T.26)`
			`chart: RunCharts (T.35)`
			`behavior: SimulationBehavior (T.18)`

Tabulka B.24: **Deklarace API endpointu pro vykonání zásahu do průběhu experimentu** /simulation/:simulationId/experiment/:experimentId/run/:runId/execute

/simulation/:simulationId/experiment/:experimentId/run/:runId/update

Tento endpoint slouží k získání aktuálních dat běhu experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.25.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**

GET		200	`progress: Number,`
			`charts: RunCharts[] (T.35)`
			`mixing: Number[4],`
			`events: TimelineEvent[] (T.27)`

Tabulka B.25: **Deklarace API endpointu pro získání aktuálnách dat běhu** /simulation/:simulationId/experiment/:experimentId/run/:runId/update

/simulation/:simulationId/experiment/:experimentId/run/:runId/result

Tento endpoint slouží k získání dat výsledků běhu experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.26.

/simulation/:simulationId/experiment/:experimentId/run/:runId/downloadCSV

Tento endpoint slouží ke stažení výstupního CSV souboru běhu experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.27.

/simulation/:simulationId/experiment/:experimentId/run/:runId/downloadPDF

Tento endpoint slouží ke stažení výstupního CSV souboru běhu experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.28.

Cíl:			
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**

GET		200	`run: ExperimentRun (T.31),`
			`timelines:Timeline[4] (T.26),`
			`experiment: SimulationExperiment (T.29),`
			`drug: Drugs (T.45),`
			`simulation: Simulation (T.16)`

Tabulka B.26: **Deklarace API endpointu /config/params/sync** /config/params/sync

Cíl:	/config/params/sync		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`text/plain`

Tabulka B.27: **Deklarace API endpointu /config/params/sync**

Cíl:	/config/params/sync		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`application/pdf`

Tabulka B.28: **Deklarace API endpointu /config/params/sync**

/simulation/:simulationId/experiment/:experimentId/run/:runId/downloadZip

Tento endpoint slouží ke stažení výstupního ZIP archivu s výstupy běhu experimentu. Stavové kódy a návratové hodnoty jsou uvedeny v tabulce B.29.

Cíl:	/config/params/sync		
Dotaz		**Odpovědi**	
Typ	**Tělo dotazu**	**Kód**	**Tělo odpovědi**
GET		200	`application/zip`

Tabulka B.29: **Deklarace API endpointu /config/params/sync**

B.2 Datové typy

V této podkapitole jsou uvedeny rozhraní a datové typy, které jsou v tomto systému využity jak pro komunikaci, tak pro perzistentní uložení dat. Jednotlivé typy jsou rozděleny do jednotlivých skupin podle nadřazeného typu.

B.2.1 Config

V této části je zdokumentována část datových typů, které jsou určeny pro konfiguraci systému.

```
interface Config {
    calibration: Calibration (T.2),
    debug: Debug (T.5),
    params: Params (T.10),
    users: User[] (T.14),
}
```

Výpis T.1: TypeScript deklarace rozhraní Config

Calibration

Tato část obsahuje deklarace datových typů pro kalibraci zařízení.

```
interface Calibration {
    calibrationPoints : CalibrationPoint[][] (T.3),
    selectedCalibrationSchemas: CalibrationSchema[] (T.4),
}
```

Výpis T.2: TypeScript deklarace rozhraní Calibration

```
interface CalibrationPoint {
    ph: Number,
    voltage: Number,
}
```

Výpis T.3: TypeScript deklarace rozhraní CalibrationPoint

```
interface CalibrationSchema{
    value: Number
}
```

Výpis T.4: TypeScript deklarace rozhraní CalibrationSchema

Debug

V této části jsou uvedeny deklarace datových typů a rozhraní pro ladění systému.

```
interface Debug {
    backups: Backup[] (T.6),
    deviceStates: DeviceStatesRow[] (T.8),
    runHistory: RunHistoryRow[] (T.9),

}
```

Výpis T.5: TypeScript deklarace rozhraní Debug

```
export interface Backup {
    id: number,
    date: string,
    type: BackupType (T.7),,
    size: string,
}
```

Výpis T.6: TypeScript deklarace rozhraní Backup

```
export enum BackupType {
    AUTO = 0,
    MANUAL = 1
}
```

Výpis T.7: TypeScript deklarace rozhraní BackupType

```
interface DeviceStatesRow {
    id: Number,
    ph: Boolean,
    pump: Boolean,
    temperature: Boolean,
}
```

Výpis T.8: TypeScript deklarace rozhraní DeviceStatesRow

```
interface RunHistoryRow {
    id: Number,
    date: String,
    login: String,
    result: Boolean,
}
```

Výpis T.9: TypeScript deklarace rozhraní RunHistoryRow

Params

V této části jsou uvedeny deklarace datových typů a rozhraní pro nastavení systému.

```
export interface Params {
    ntp: Ntp (T.11),
    wifi: WiFi (T.12),
}
```

Výpis T.10: TypeScript deklarace rozhraní Params

```
export interface Ntp {
    date: string,
    NTPserver: string
```

```
}
```

Výpis T.11: TypeScript deklarace rozhraní Ntp

```
export interface WiFi {
    name: string,
    password?: string,
    mode: WifiMode (T.13),
}
```

Výpis T.12: TypeScript deklarace rozhraní WiFi

```
export enum WifiMode {
    CLIENT = 0,
    AP = 1,
}
```

Výpis T.13: TypeScript deklarace výčtového typu WifiMode

Users

V této části jsou uvedeny deklarace datových typů a rozhraní pro ukládání a správu uživatelů.

```
interface User {
    id: number,
    login: string,
    first_name: string,
    last_name: string,
    type: UserType (T.15),
}
```

Výpis T.14: TypeScript deklarace rozhraní User

```
export enum UserType {
    ADMIN = 'admin',
    USER = 'user'
}
```

Výpis T.15: TypeScript deklarace rozhraní UserType

B.2.2 Simulation

V této části jsou uvedeny deklarace datových typů a rozhraní pro simulace.

```
export interface Simulation {
    id: number,
    settings: SimulationSettings (T.17),
```

```
    behavior : SimulationBehavior (T.18),
    parameters : SimulationParameters (T.24),
    experiments : Experiments (T.29),
    timelines?: Timelines (T.26),
}
```

Výpis T.16: TypeScript deklarace rozhraní Simulation

SimulationSettings

V této části jsou uvedeny deklarace datových typů a rozhraní nastavení simulace.

```
export interface SimulationSettings {
    description: string,
    bagCount: number,
    temperature: number,
    volume: number,
    enzymeTargetBag: number,
    bags: number[],
}
```

Výpis T.17: TypeScript deklarace rozhraní SimulationSettings

SimulationBehavior

V této části jsou uvedeny deklarace datových typů a rozhraní pro automatické řízení průběhu experimentu.

```
export interface SimulationBehavior {
    centerX: number,
    centerY: number,
    scale: number,
    nodes: SimulationBehaviorNode[] (T.19),
    links: SimulationBehaviorLink[] (T.20),
}
```

Výpis T.18: TypeScript deklarace rozhraní SimulationBehavior

```
interface SimulationBehaviorNode {
    id: number,
    x: number,
    y: number,
    type: string,
    label: string,
    content: string,
    data: {
        source?: string,
```

```
            index?: number,
            value?: string,
            action?: string,
            operator?: string,
        },
        unit?: string,
        color: string,
        inputs: SimulationBehaviorPort[] (T.21),
        outputs: SimulationBehaviorPort[] (T.21),
    }
```

Výpis T.19: TypeScript deklarace rozhraní SimulationBehaviorNode

```
interface SimulationBehaviorLink {
    id: number,
    from: {node: number, output: number},
    to: {node: number, input: number},
}
```

Výpis T.20: TypeScript deklarace rozhraní SimulationBehaviorLink

```
interface SimulationBehaviorPort {
    name: string,
    color: string
}
```

Výpis T.21: TypeScript deklarace rozhraní SimulationBehaviorPort

SimulationParameters

V této části jsou uvedeny deklarace datových typů a rozhraní pro parametry simulace.

```
export type SimulationParameters = SimulationParametersRow[];
```

Výpis T.22: TypeScript deklarace typu SimulationParametersRow

```
interface SimulationParametersCol {
    value: number,
    unit: string
}
```

Výpis T.23: TypeScript deklarace rozhraní SimulationParametersCol

```
interface SimulationParametersRow {
    time: [{ h: number, m: number, s:number }],
    pump: SimulationParametersCol[] (T.23),
    sample: SimulationParametersCol[] (T.23),
```

```
    enzyme: SimulationParametersCol[] (T.23),
    mixing: SimulationParametersCol[] (T.23),
    notification: string,
}
```

Výpis T.24: TypeScript deklarace rozhraní SimulationParametersRow

Timelines

V této části jsou uvedeny deklarace datových typů a rozhraní časových os simulací.

```
export type Timelines = Timeline[];
```

Výpis T.25: TypeScript deklarace typu Timeline

```
interface Timeline {
    events: TimelineEvent[] (T.27),
}
```

Výpis T.26: TypeScript deklarace rozhraní Timeline

```
interface TimelineEvent {
    time: number,
    color: string,
    iconColor: string,
    type: string,
    icon?: string,
    speed?: string,
    volume?: string,
    notification?: string,
    size?: string,
    done: Boolean,
}
```

Výpis T.27: TypeScript deklarace rozhraní TimelineEvent

```
interface TimelineEventType {
    label: string,
    icon: string,
    color: string
},
```

Výpis T.28: TypeScript deklarace rozhraní TimelineEventType

B.2.3 Experiment

V této části jsou uvedeny deklarace datových typů a rozhraní experimentu.

```
export type Experiments = Experiment[];
export interface Experiment {
    runs?: ExperimentRuns (T.31),
    settings?: ExperimentSetting (T.30),
}
```

Výpis T.29: TypeScript deklarace typu Experiments a rozhraní Experiment

ExperimentSetting

V této části jsou uvedeny deklarace datových typů a rozhraní nastavení experimentu.

```
interface ExperimentSetting {
    id: number,
    name: string,
    author: string,
    drug: number,
    dose: number,
    description: string,
    temperatureMaxDeviation: number,
    phMaxDeviation: number,
    access: Boolean
}
```

Výpis T.30: TypeScript deklarace rozhraní ExperimentSetting

B.2.4 Run

V této části jsou uvedeny deklarace datových typů a rozhraní běhu.

```
type ExperimentRuns = ExperimentRun[];

interface ExperimentRun {
    data?: RunData (T.32),
    charts?: RunCharts (T.35),
    settings?: RunSettings (T.34),
}
```

Výpis T.31: TypeScript deklarace typu ExperimentRuns a rozhraní ExperimentRun

RunData

V této části jsou uvedeny deklarace datových typů a rozhraní dat experimentu.

```
interface RunData {
    isRunning: Boolean,
    isPaused: Boolean,
    isDone: Boolean,
    duration: number,
    progress: string
    targetConditions: TargetConditions (T.33),
```

```
    result: Boolean,
    sampleMaxDelay: number,
    enzymeMaxDelay: number,
    phDeviation: number,
    temperatureDeviation: number,
}
```

Výpis T.32: TypeScript deklarace rozhraní RunData

```
interface TargetConditions {
    ph: Number[4],
    mixing: Number[4],
    pumps: Number[4],

    temperature: number,
    phStabilization: number,
    temperatureStabilization: number,
    ignorePhEvents: Boolean,
    pHMode: number,
    manualPump: Boolean,
    manualMixing: Boolean,
    manualTemperature: Boolean,
    manualEnzyme: Boolean,
    delay: number,
    volume: number,
    temperatureMaxDeviation: number
    phMaxDeviation: number,
}
```

Výpis T.33: TypeScript deklarace rozhraní TargetConditions

RunSettings

V této části jsou uvedeny deklarace datových typů a rozhraní nastavení běhu.

```
interface RunSettings {
    id: number,
    name: string,
    author: string,
    date: string,
}
```

Výpis T.34: TypeScript deklarace rozhraní RunSettings

RunCharts

V této části jsou uvedeny deklarace datových typů a rozhraní grafů běhu.

```
interface RunCharts {
    ph: ChartDataset[] (T.36),
```

```
    temperature: ChartDataset[] (T.36),
    volumes: ChartDataset[] (T.36),
}
```

Výpis T.35: TypeScript deklarace rozhraní RunCharts

```
type ChartDataset = chartPoint[];

    interface ChartPoint {
        x: number,
        y: number,
    }
```

Výpis T.36: TypeScript deklarace rozhraní ChartPoint a typu chartDataset

B.2.5 RunHistory

V této části jsou uvedeny deklarace datových typů a rozhraní historie běhu.

```
export interface RunHistory {
    id: number,
    date: string,
    login: string,
    result: boolean,
    forcedChanges?: ForcedChange[] (T.40),
    confirmations?: Confirmation[] (T.43),
}
```

Výpis T.37: TypeScript deklarace rozhraní RunHistory

```
    export enum ForcedChangeType {
        PUMP = 'pump',
        MIXING = 'mixing',
        PH = 'ph',
        TEMP = 'temperature',
        ENZYME = 'enzyme',
        STOP = 'stop'
    }
```

Výpis T.38: TypeScript deklarace výčtového typu ForcedChangeType

```
    export enum ForcedChangeAction {
        UP = 'up',
        DOWN = 'down'
    }
```

Výpis T.39: TypeScript deklarace rozhraní ForcedChangeAction

```
interface ForcedChange {
    name: ForcedChangeType,
    index: number,
    value: number | boolean,
    time: number,
    action?: ForcedChangeAction (T.39),
}
```

Výpis T.40: TypeScript deklarace rozhraní ForcedChange

```
export enum ConfirmationType {
    ENZYME = 'enzyme',
    SAMPLE = 'sample',
}
```

Výpis T.41: TypeScript deklarace výčtového typu ConfirmationType

```
export enum ConfirmationAction {
    CONFIRM = 'confirm'
}
```

Výpis T.42: TypeScript deklarace výčtového typu ConfirmationAction

```
interface Confirmation {
    name: ConfirmationType,
    index: number,
    value: boolean,
    action: ConfirmationAction (T.42),
    time: number,
    delay: number,
}
```

Výpis T.43: TypeScript deklarace rozhraní Confirmation

B.2.6 Substances

V této části jsou uvedeny deklarace datových typů a rozhraní lékových substancí.

```
export interface Substances {
    variants: Variants (T.46),
    drugs: Drugs (T.45),
}
```

Výpis T.44: TypeScript deklarace rozhraní Substances

```
export type Drugs = Drug[];
interface Drug {
    id: number,
```

```
    name: string,
}
```

Výpis T.45: TypeScript deklarace rozhraní Drug a typu Drugs

```
export type Variants = Variant[];
interface Variant {
    id: number,
    name: string,
    drug: number,
}
```

Výpis T.46: TypeScript deklarace rozhraní Variant a typu Variants

Příloha C

Snímky obrazovek aplikace

V této kapitole přílohy jsou uvedeny uvedeny snímky obrazovek jednotlivých částí systému. Tato kapitola slouží zejména jako prezentace snímků výsledného uživatelského rozhraní, které byly příliš rozměrné pro umístění do textu této diplomové práce.

C.1 Podpůrné obrazovky

Tato podkapitola obsahuje snímky obrazovek, které mají podpůrný charakter. Patří zde přihlašovací obrazovka, která je uvedena na obrázku C.1 a obrazovka hlavní nabídky aplikace na obrázku C.2.

C.2 Konfigurace

V této podkapitole jsou obsaženy snímky uživatelského rozhraní pro konfiguraci systému (C.7), kalibraci (C.5), ladění (C.4) a správu uživatelů (C.8). Je zde rovněž uvedena i obrazovka sloužící jako hlavní nabídka pro konfiguraci (C.3).

C.3 Simulace

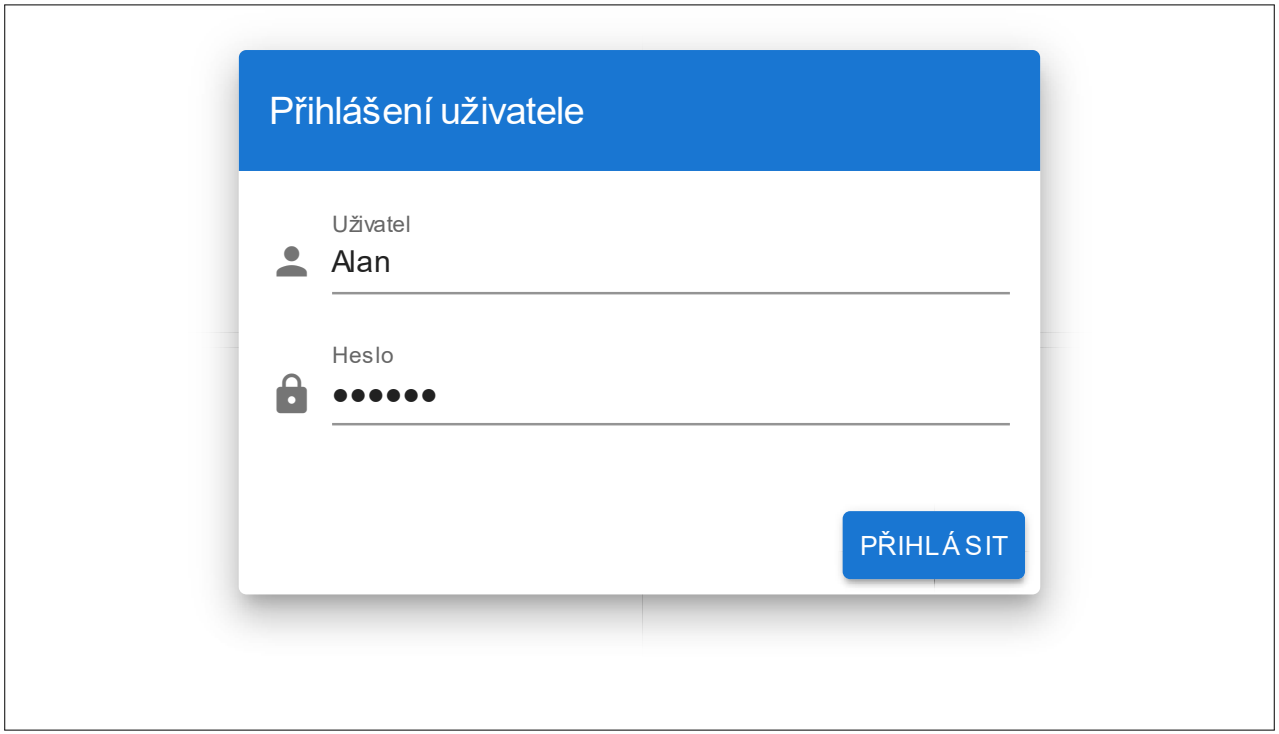
Tato část obsahuje snímky pro správu simulací. Hlavní je zde obrazovka pro výběr experimentu (C.9), skrze kterou je po volbě konkrétní simulace možné přistoupit ke stránce pro editaci simulace (C.10). Z této stránky je následně přístupná stránka pro editaci parametrů simulace (C.11) a stránka pro definování řízení průběhu experimentu pomocí vizuálního programování, kterou můžete vidět na obrázku C.12.

C.4 Experiment

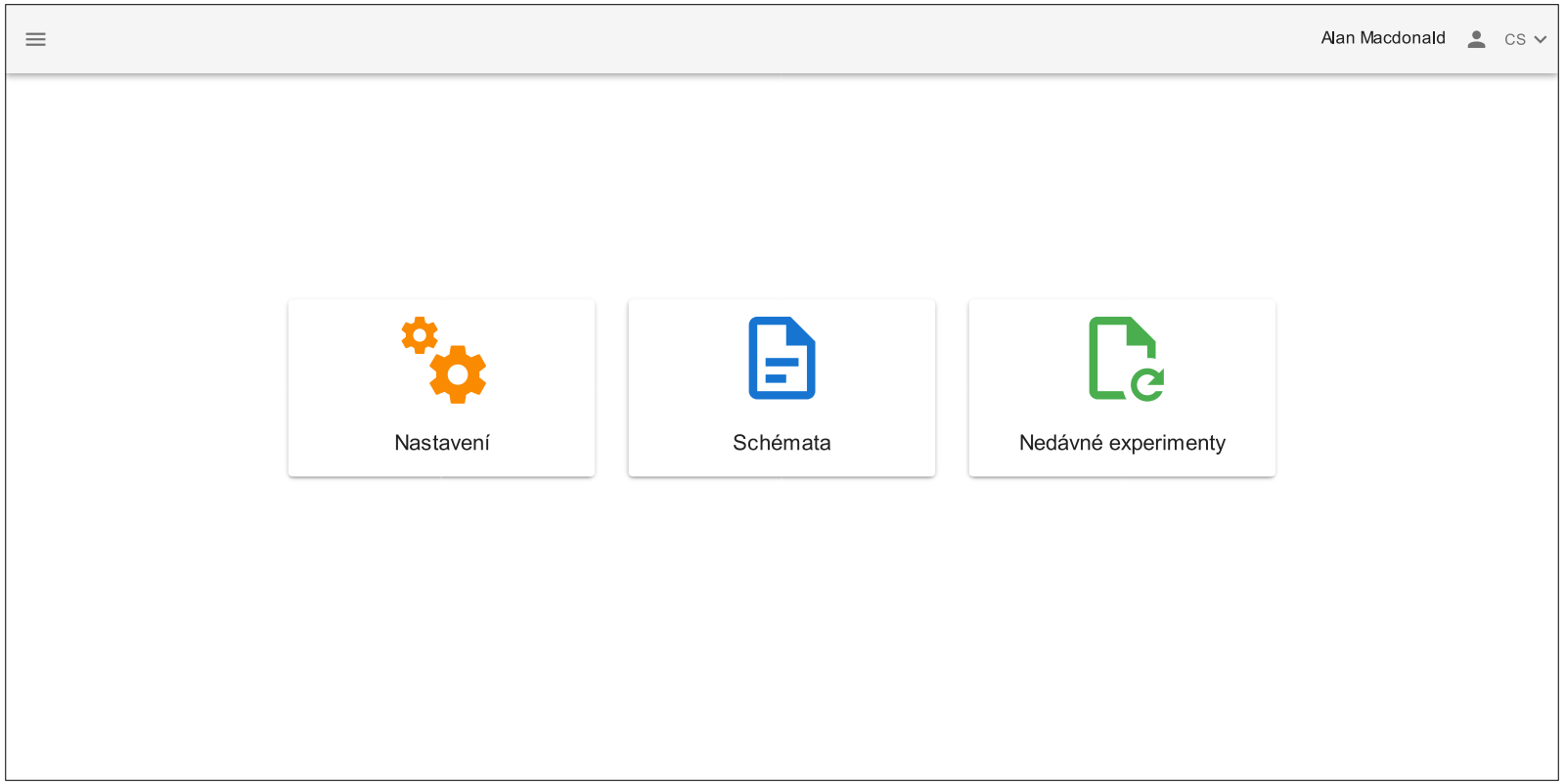
Tato sekce obsahuje snímky pro otevření experimentu (C.14), editaci experimentu (C.15) a správu databáze léků (C.16).

C.5 Běh

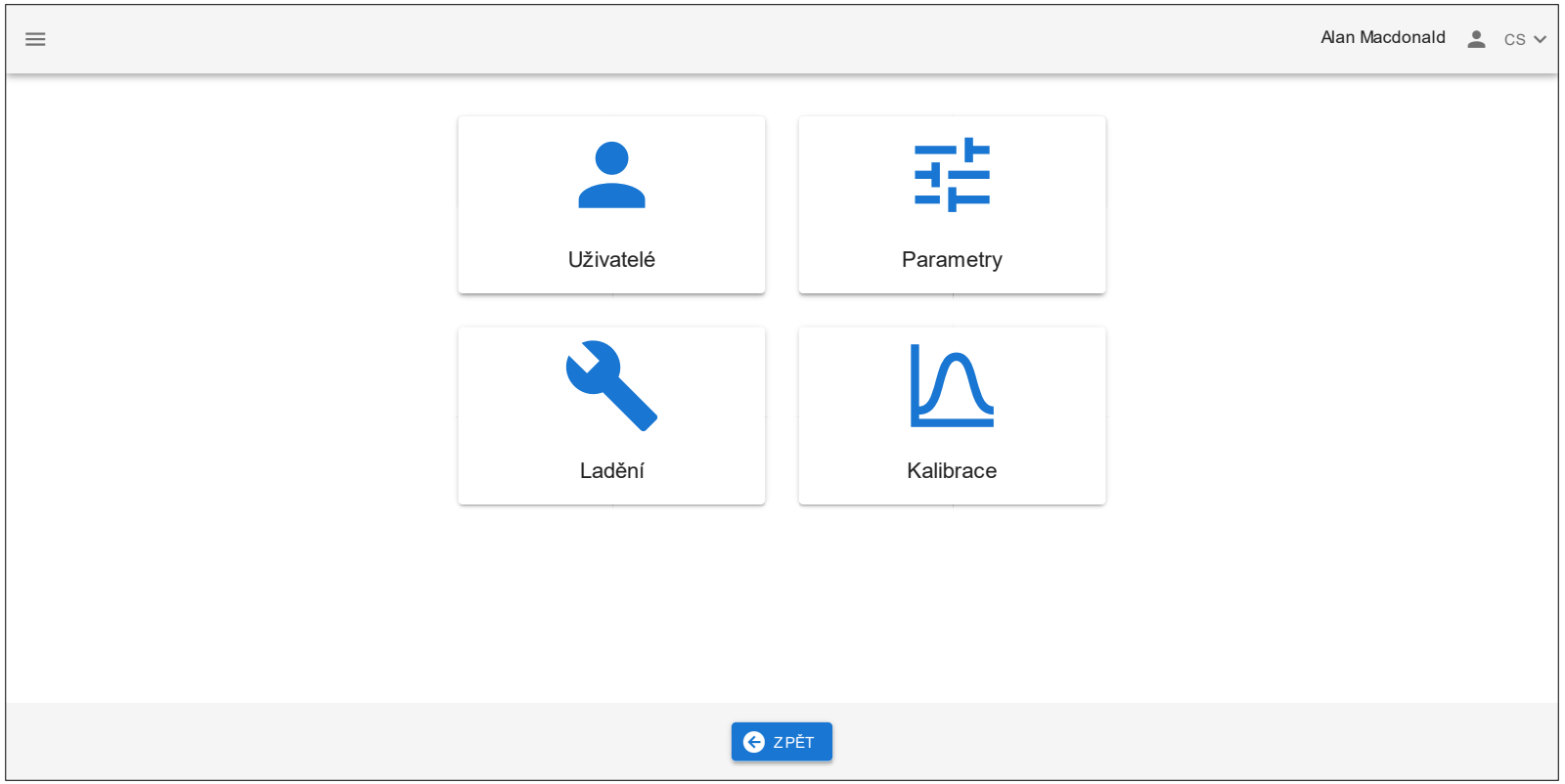
Tato sekce sdružuje snímky obrazovek pro výběr a spuštění běhu (C.17), řízení a sledování běhu experimentu (C.18) a obrazovku pro zobrazení výsledků experimentu (C.19, C.20).



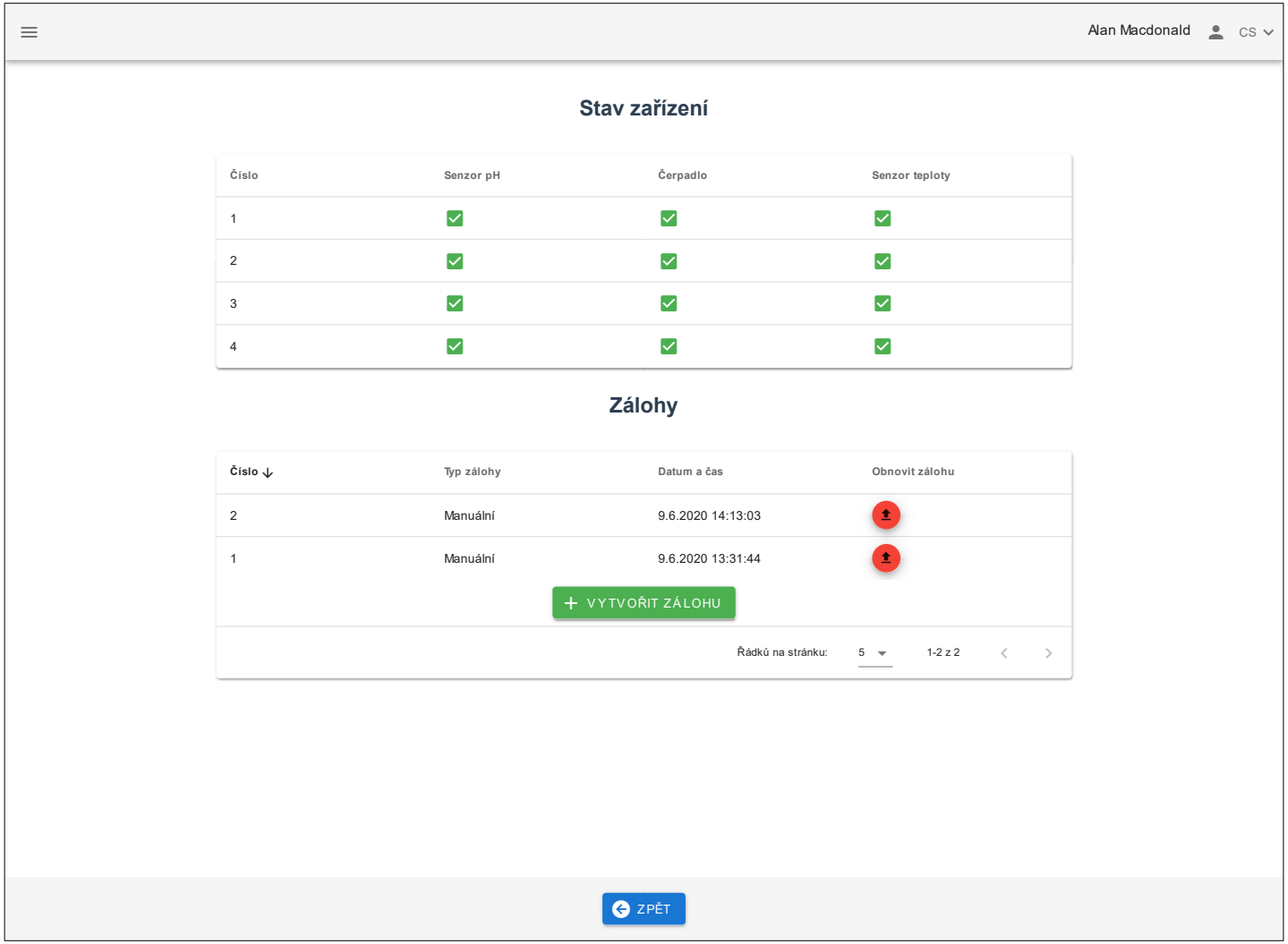
Obrázek C.1: **Snímek přihlašovací obrazovky**



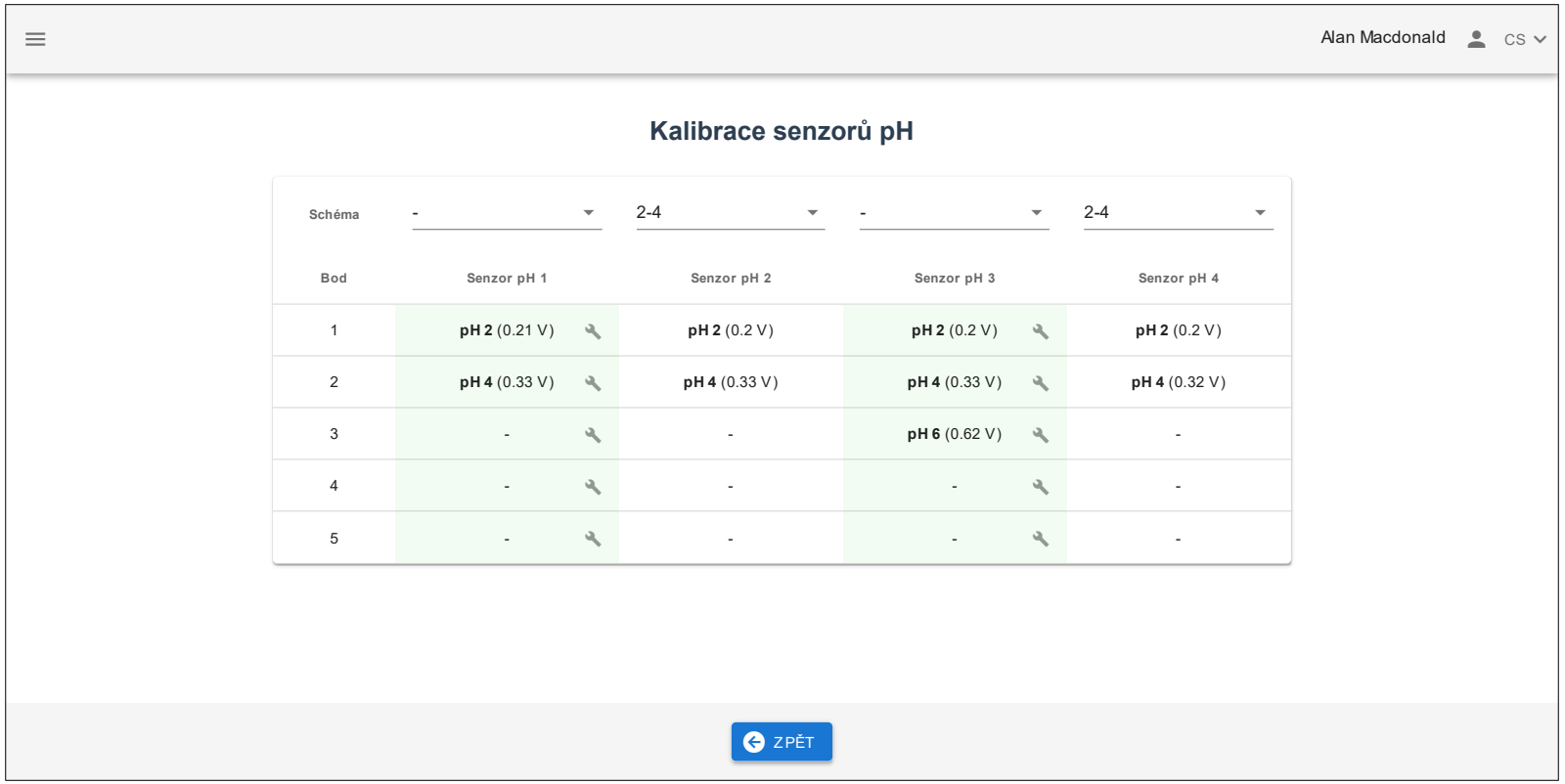
Obrázek C.2: **Snímek hlavní nabídky**



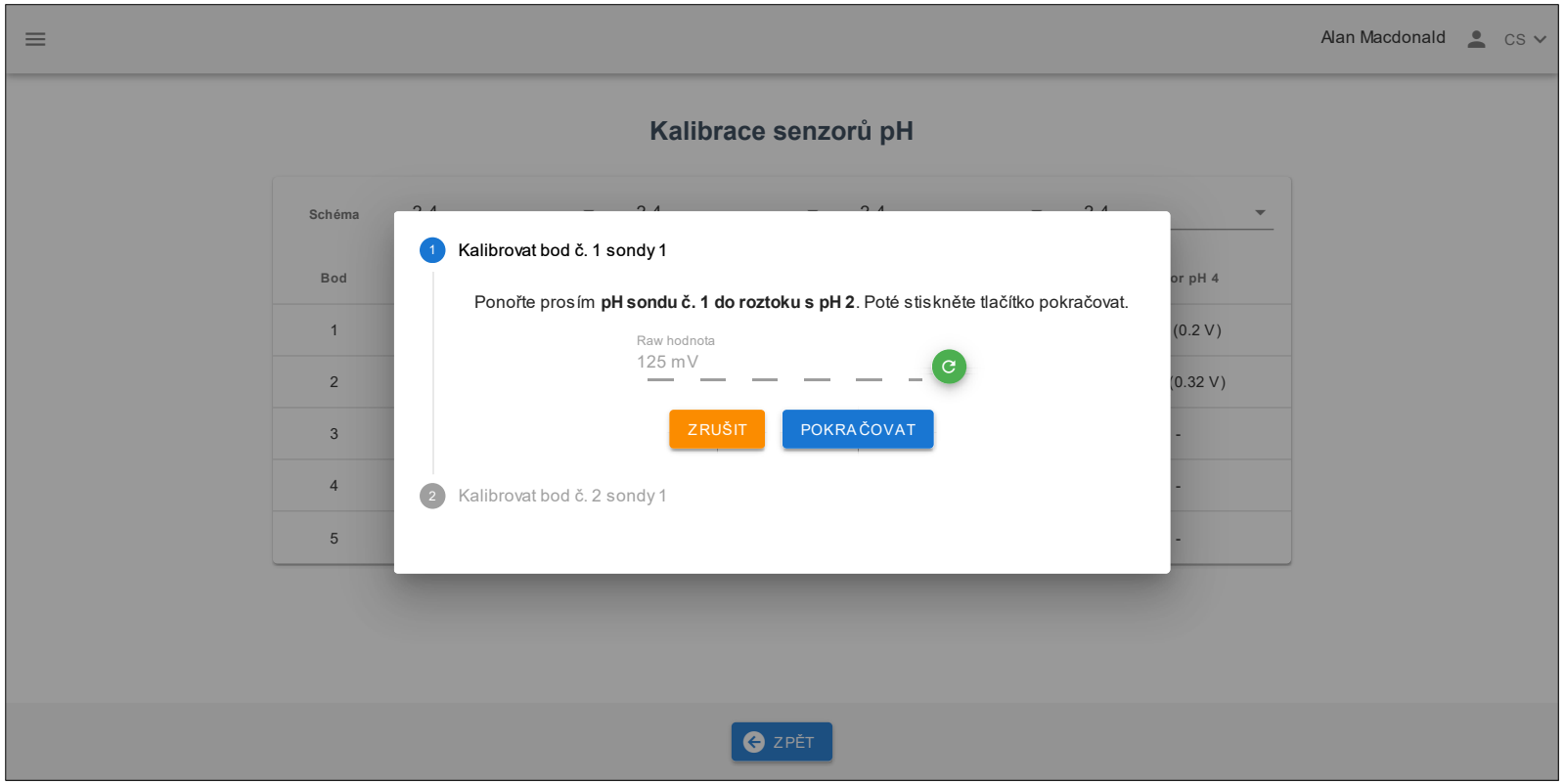
Obrázek C.3: **Snímek konfigurační nabídky**



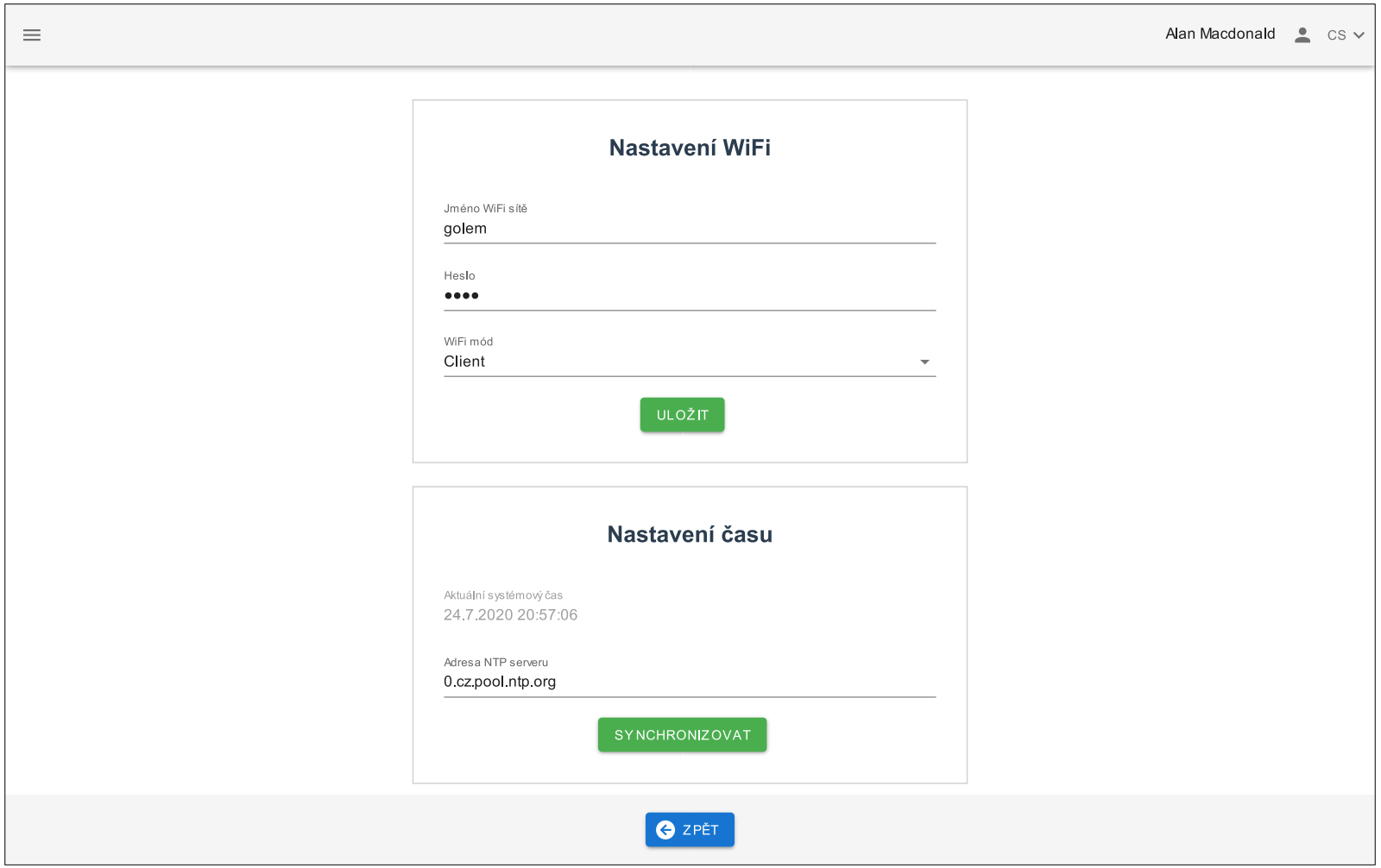
Obrázek C.4: **Snímek obrazovky pro ladění**



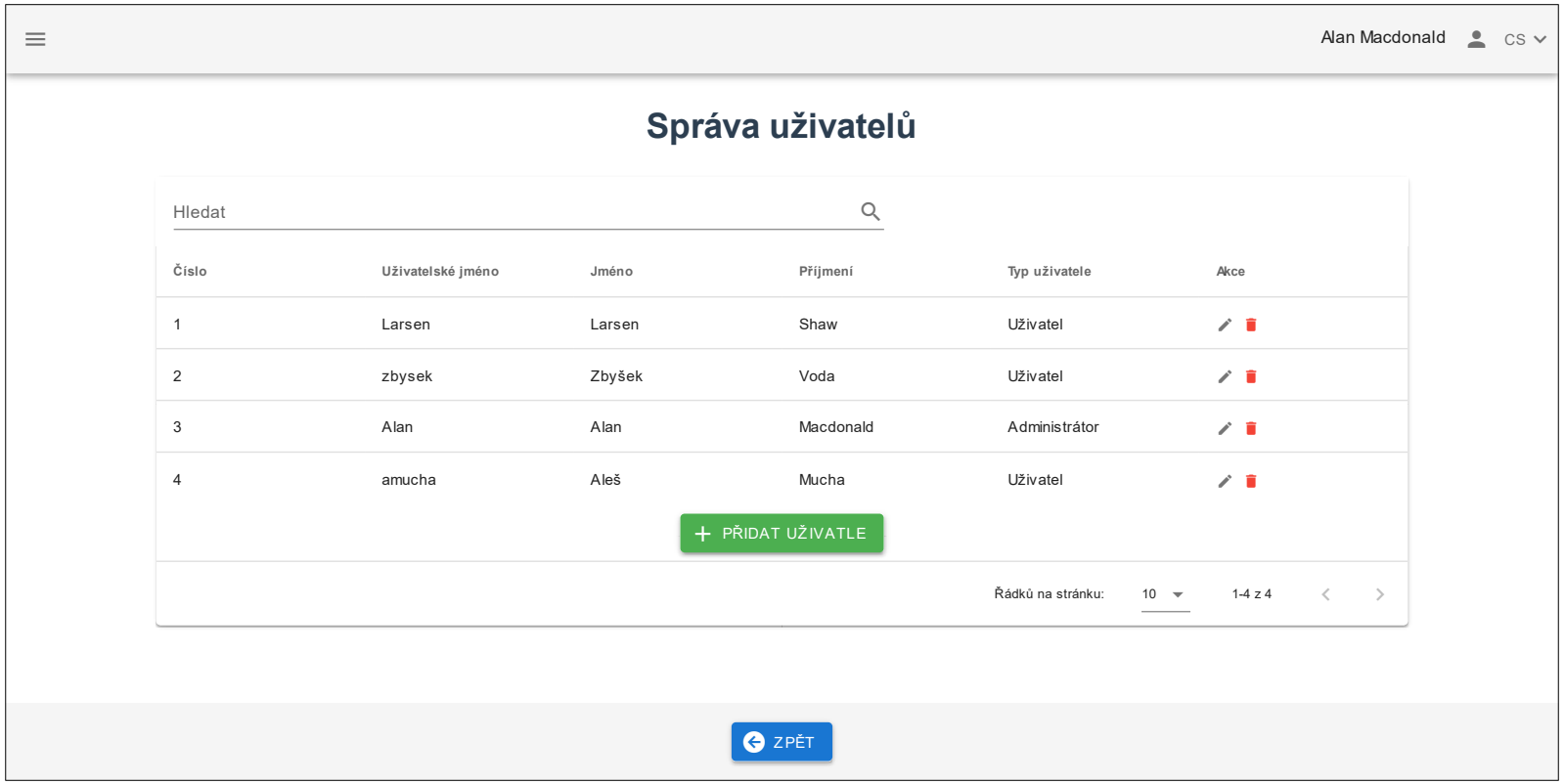
Obrázek C.5: **Snímek obrazovky pro kalibraci**



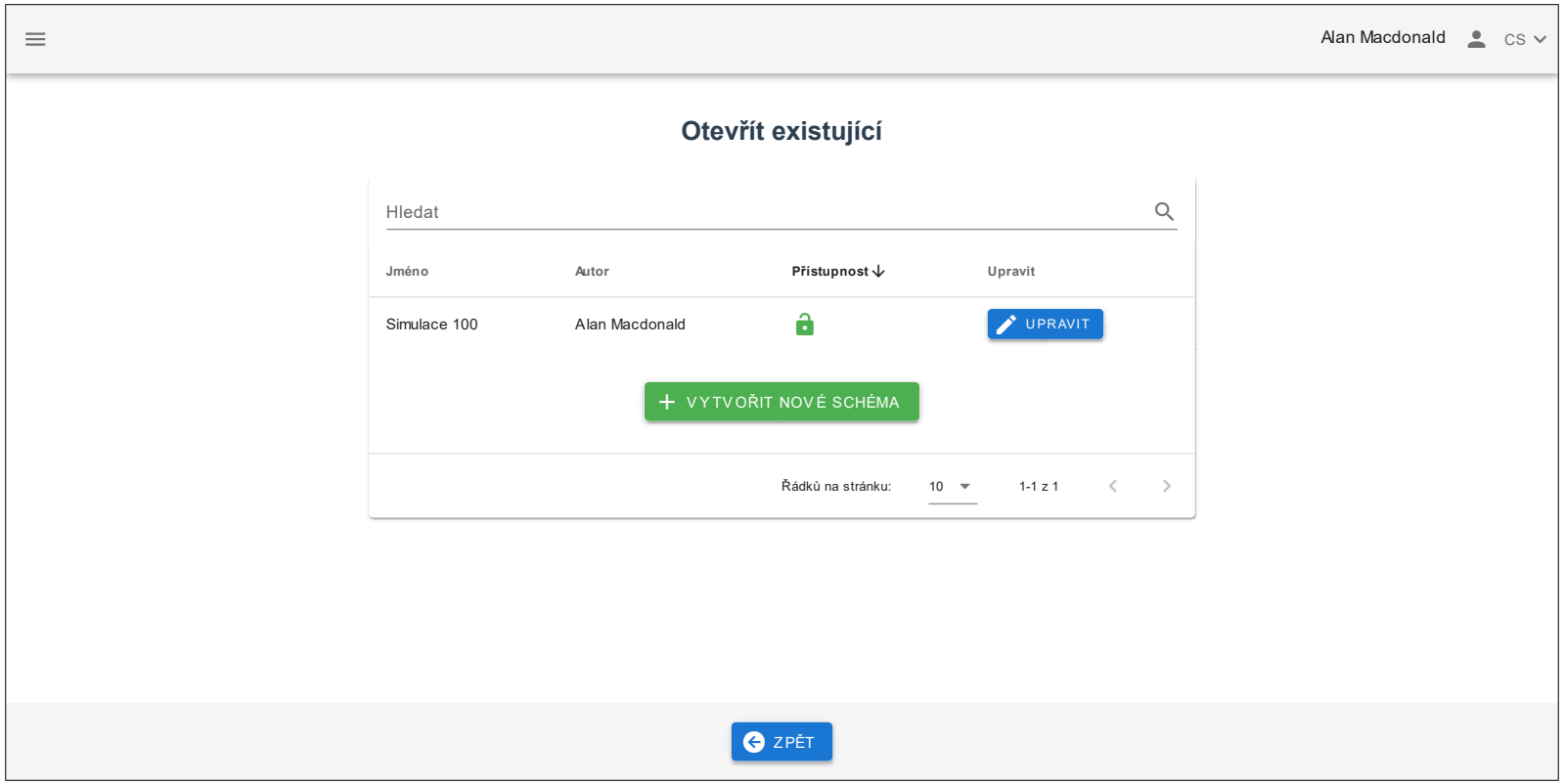
Obrázek C.6: **Snímek obrazovky s kalibračním dialogem**



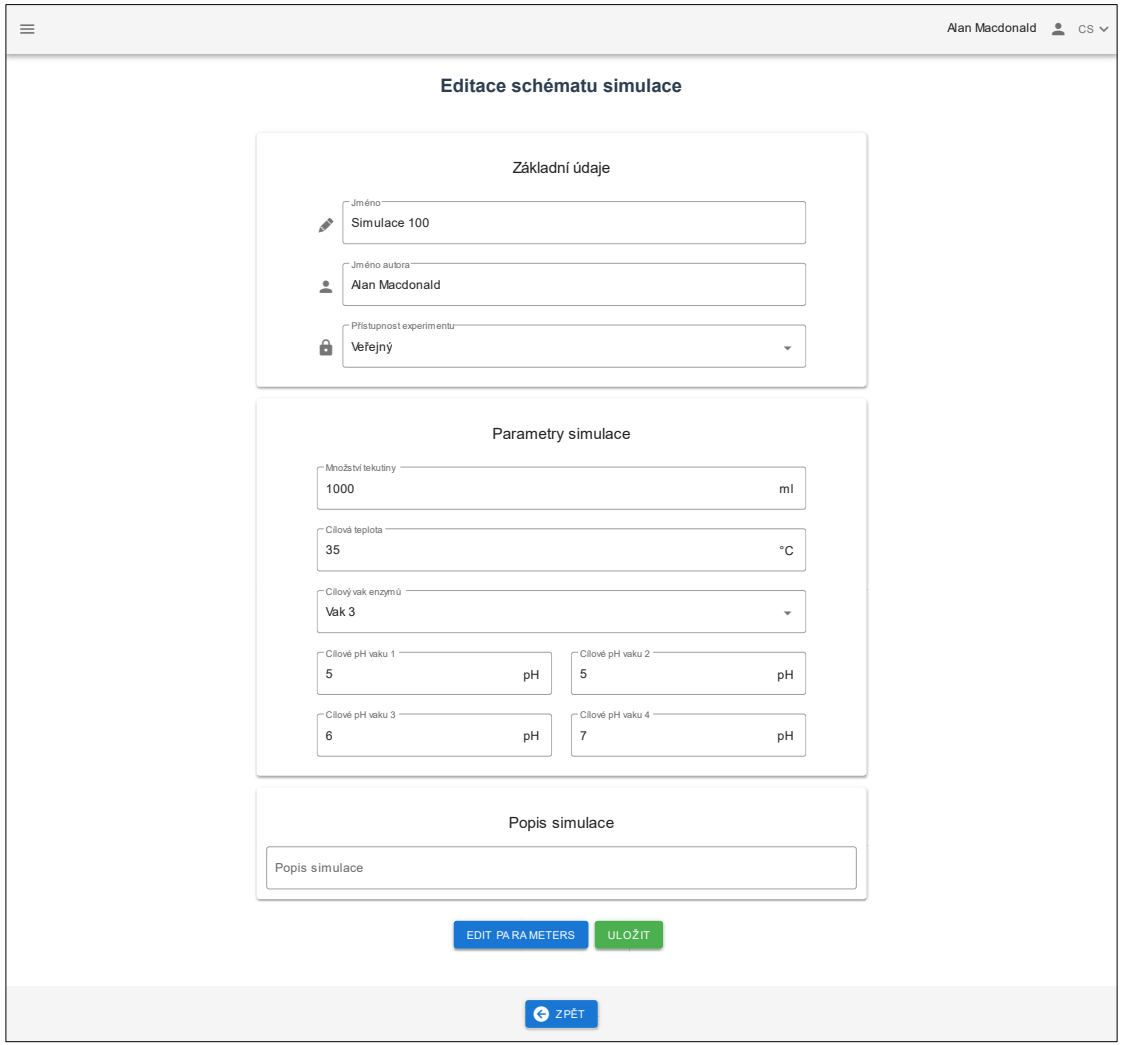
Obrázek C.7: **Snímek obrazovky nastavení zařízení**



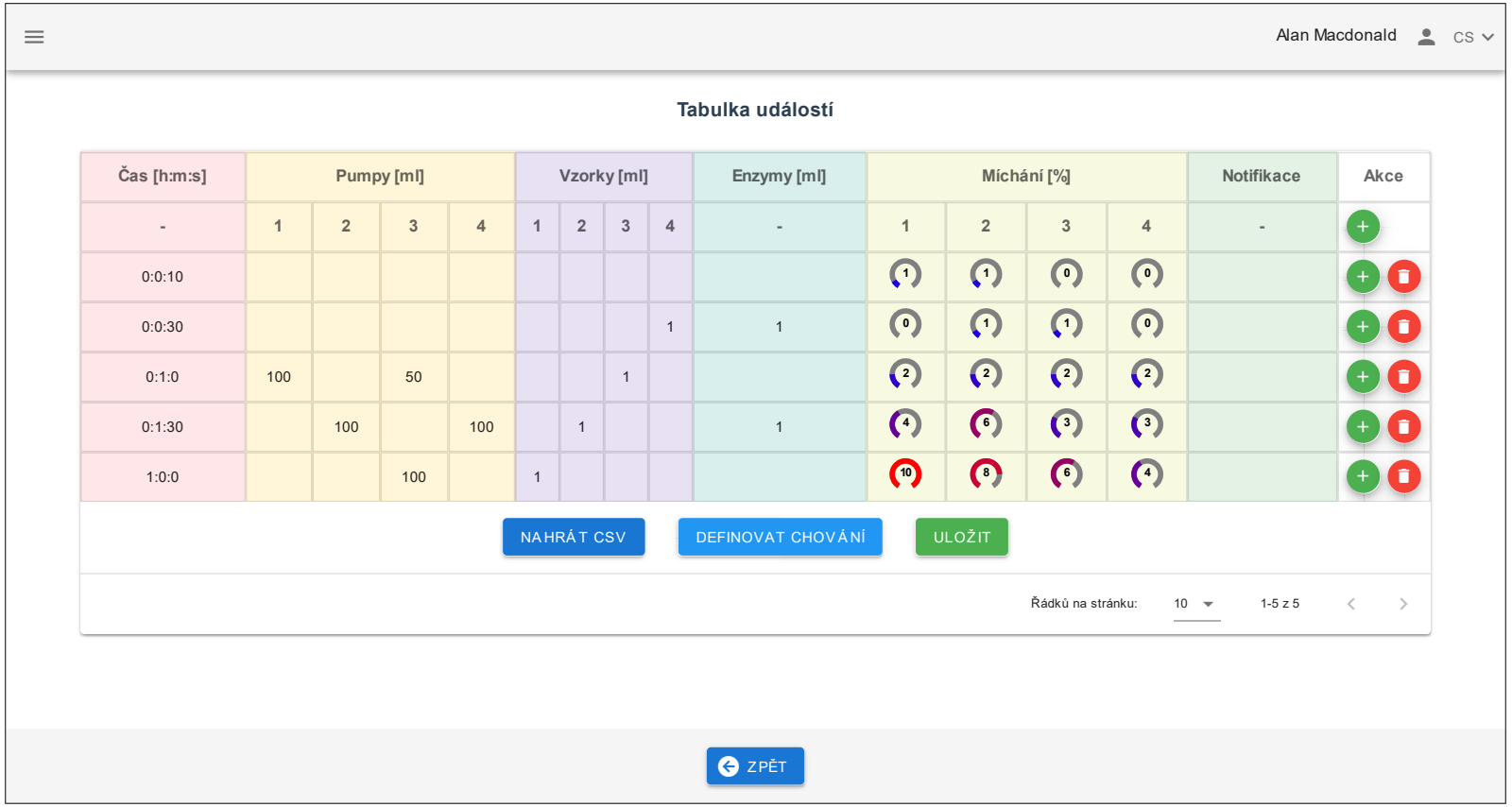
Obrázek C.8: **Snímek obrazovky pro správu uživatelů**



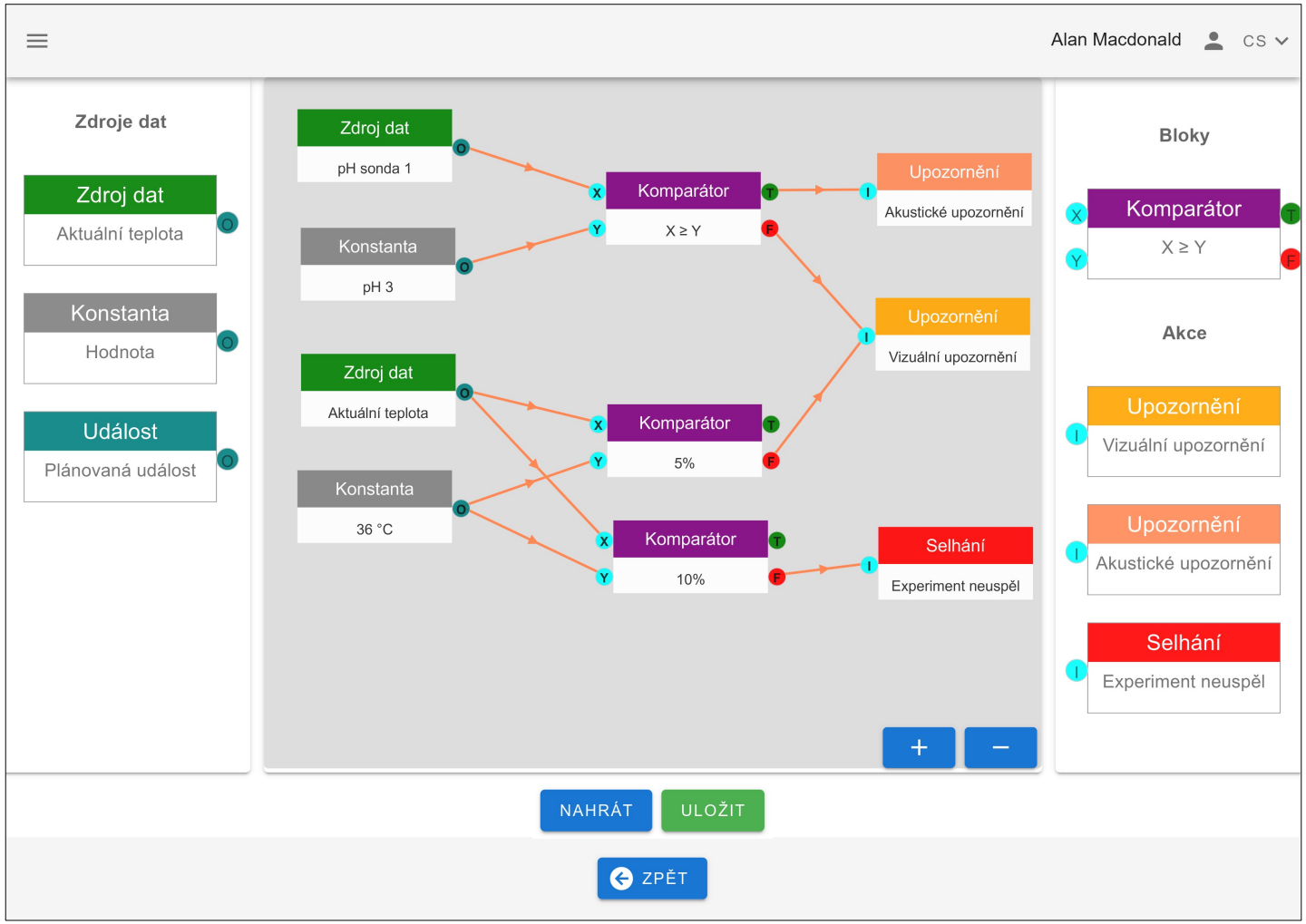
Obrázek C.9: **Snímek obrazovky pro výběr simulace**



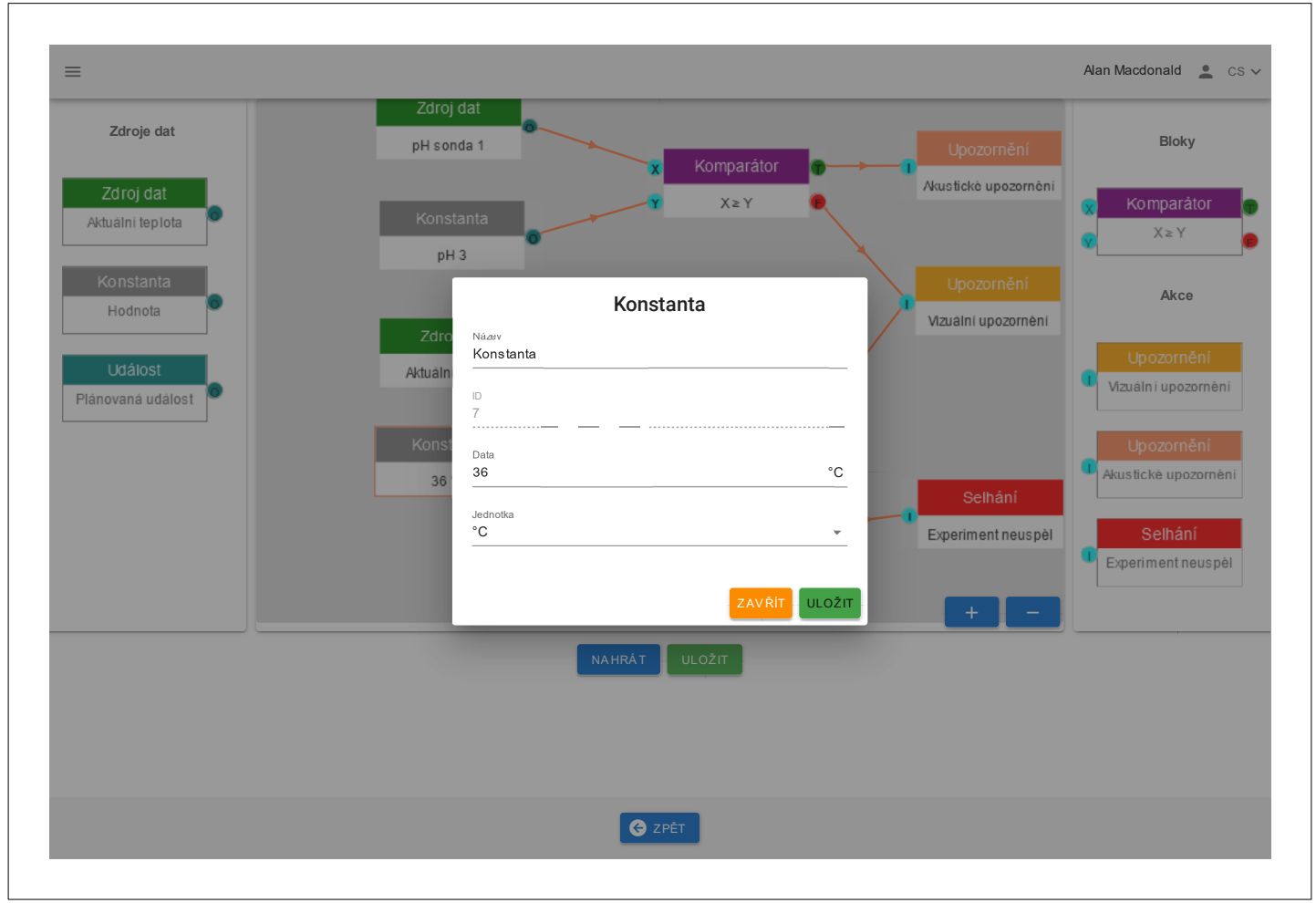
Obrázek C.10: **Snímek obrazovky pro editaci simulace**



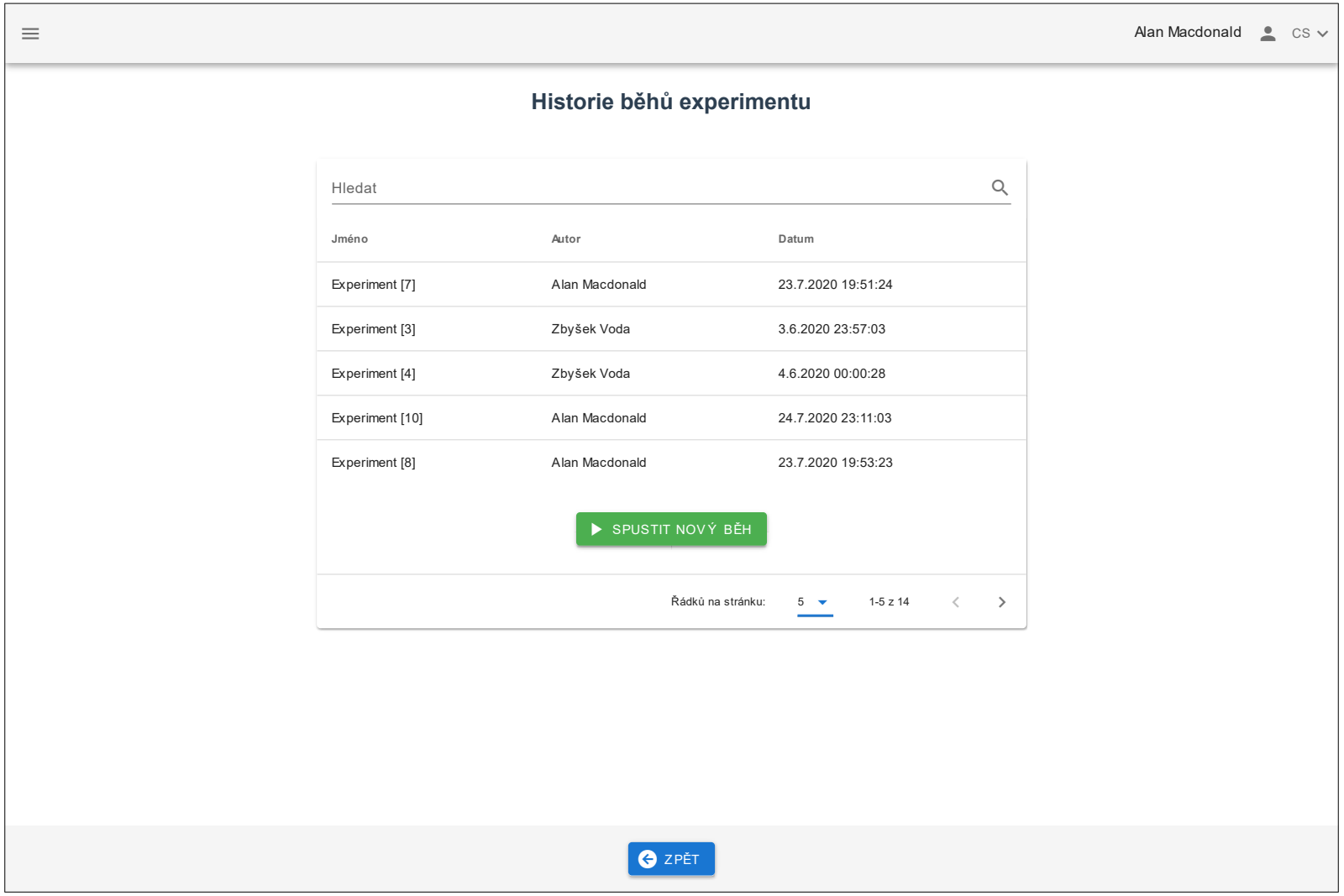
Obrázek C.11: **Snímek obrazovky pro editaci parametrů simulace**



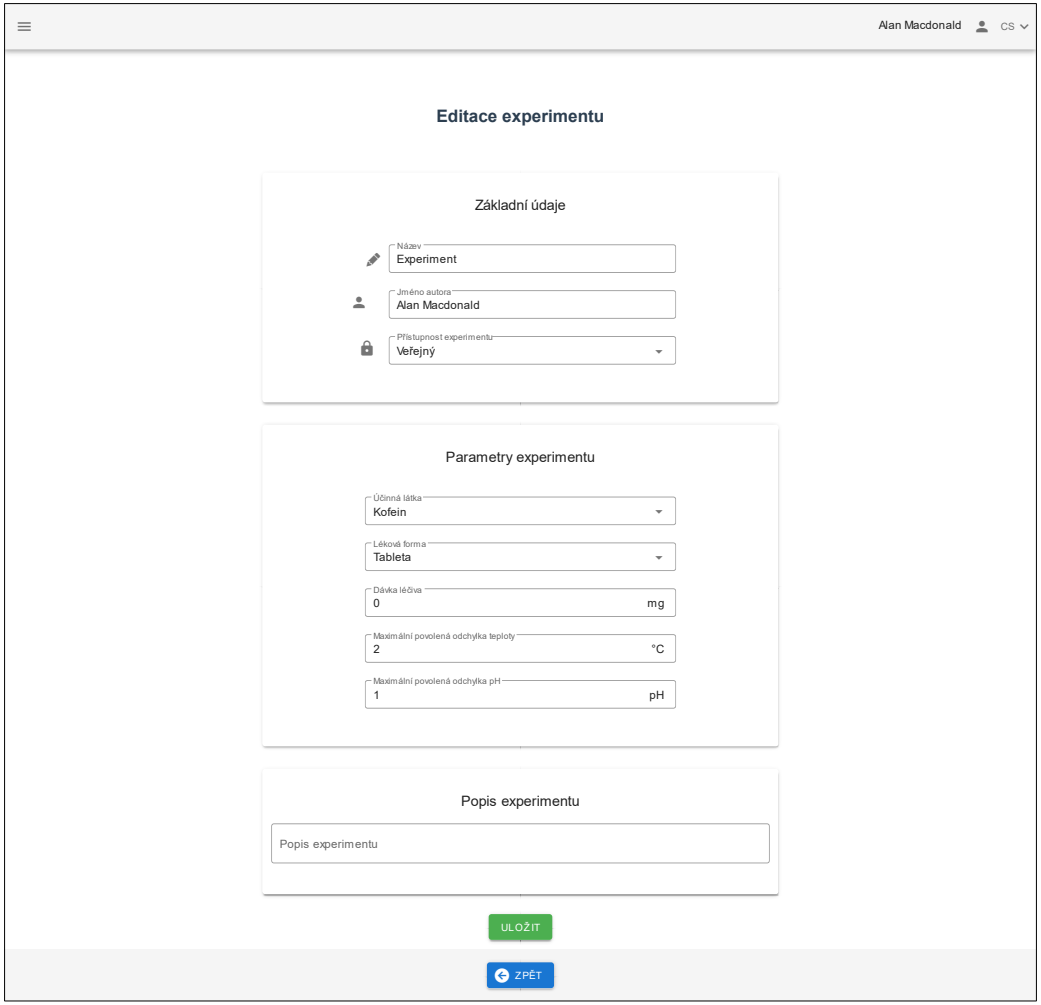
Obrázek C.12: **Snímek obrazovky pro vizuální specifikaci řízení průběhu experimentu**



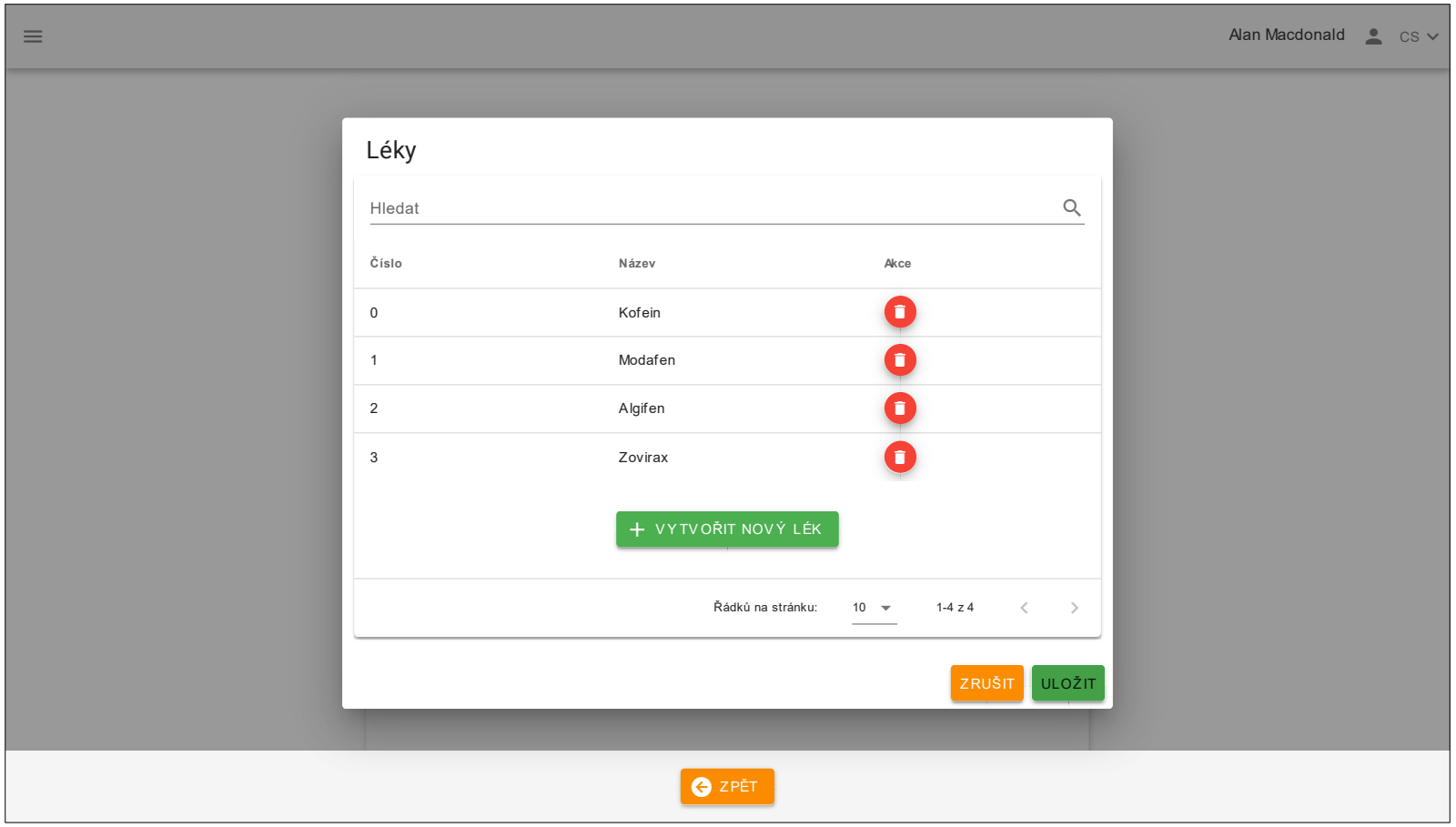
Obrázek C.13: **Snímek dialogového okna pro nastavení parametrů uzlu**



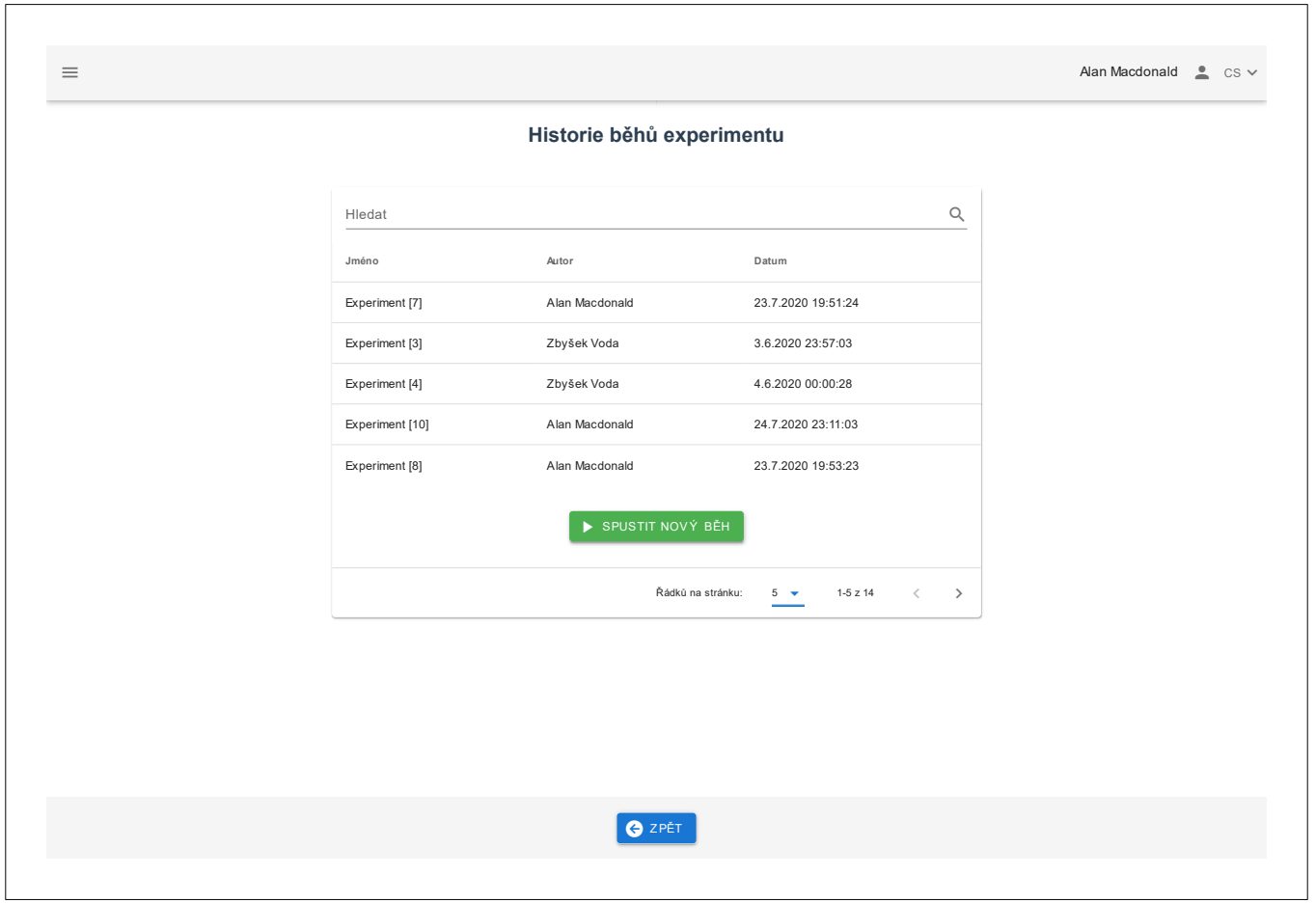
Obrázek C.14: **Snímek obrazovky pro výběr experimentu**



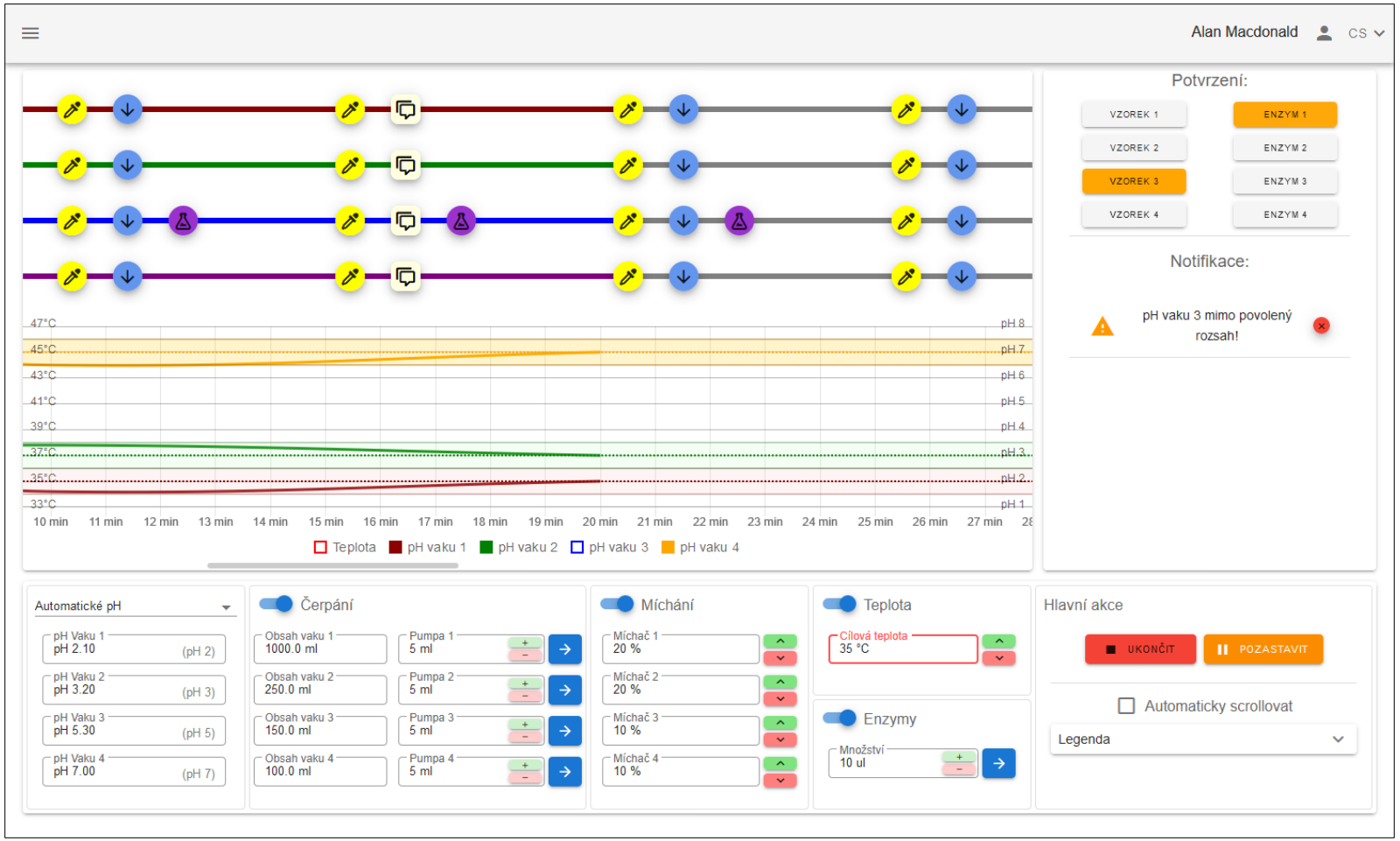
Obrázek C.15: **Snímek obrazovky pro editaci experimentu**



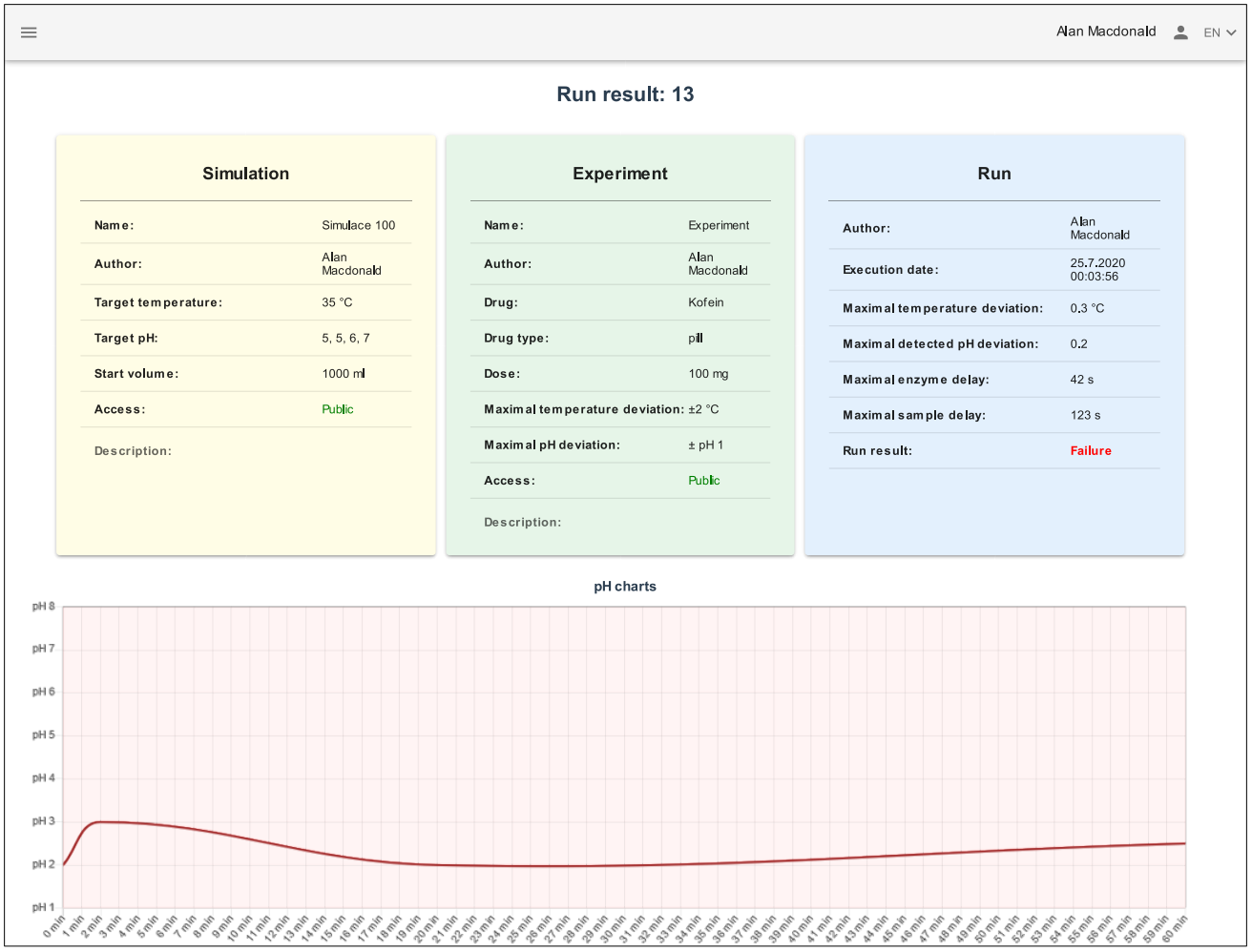
Obrázek C.16: **Snímek obrazovky s dialogovým oknem editace léků**



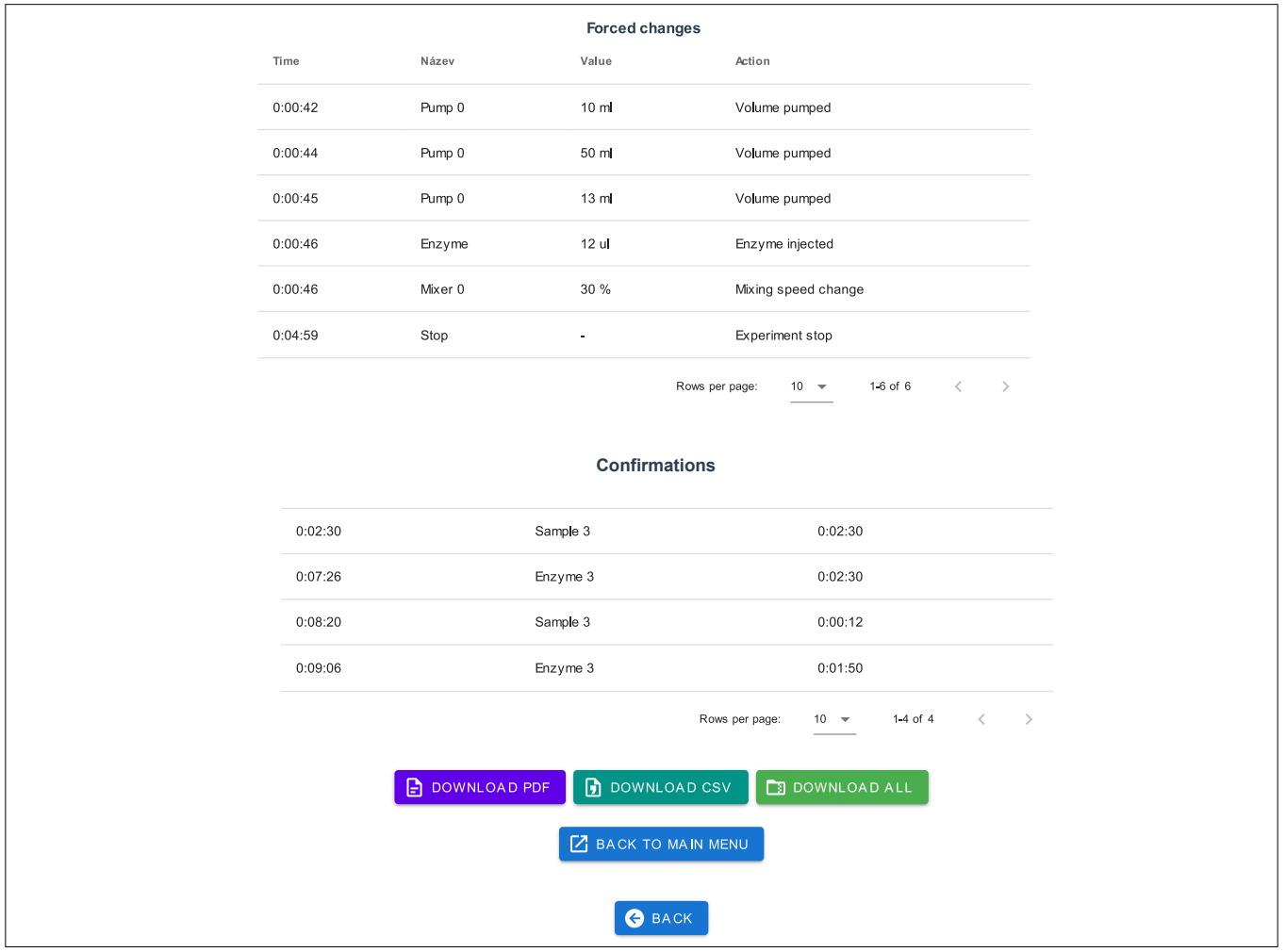
Obrázek C.17: **Snímek obrazovky pro výběr běhu**



Obrázek C.18: **Snímek obrazovky pro řízení běhu experimentu**



Obrázek C.19: **Snímek obrazovky výsledků experimentu (první část)** Snímek této obrazovky je oproti ostatním snímkům v režimu překladu do angličtiny, pro ukázku této funkce. Ze snímku byly také kvůli velkým rozměrům odstraněny zbývající grafy pro pH a teplotu.



Obrázek C.20: **Snímek obrazovky výsledků experimentu (druhá část)**

Příloha D

Akceptační protokol

Na následující stránce je umístěn akceptační protokol potvrzující přijetí práce zadavatelem. Zadavatel tímto potvrzuje, že výsledné dílo obsahuje všechny požadované funkce.

Protokol o akceptaci

Základní identifikace

Název projektu	Softwarový nástroj pro vizuální specifikaci řízení průběhu laboratorních experimentů <22340>
Zhotovitel	Bc. Jan Truhlář <xtruhl05@stud.fit.vutbr.cz>
Zadavatel	Veterinární a farmaceutická univerzita Brno

Předmět akceptace

Předmětem akceptace je nově vytvořené uživatelské rozhraní pro laboratorní přístroj GOLEM, které vzniklo v rámci diplomové práce.

Seznam výhrad

ID	VÝHRADA	ZÁVAŽNOST
1	bez výhrad	
2		
3		
4		
5		

Výsledek akceptace

- [x] Akceptováno
- [ ] Akceptováno s výhradami
- [ ] Neakceptováno

V Brně dne 23.7.2020

Podpisy

Za zadavatele	Zhotovitel
.....	.....
PharmDr. Martin Čulen, Ph.D.	Bc. Jan Truhlář