Pro školní rok: *2022/2023*
Třída: *4.A.*
Obor: *Informační technologie 18-20-M/01*

Téma práce: Inteligentní Květináč
Vedoucí práce: Vlaďka Janů

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Inteligentní květináč, který bude fungovat na bázi malého počítače a senzorů, které se bezdrátově připojí k domácí síti, následně k účtu a budou v jednoduchém a funkčním UI zobrazovat uživateli užitečná data, vlhkost, teplotu v závislosti k fázi dne apod. Aplikace bude sestavena v Next.js a bude viset na webu, uživatel si vytvoří účet a květináče si naváže na něj.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

Konec září: hotové UI, navržená databáze
Konec října: hotový květináč a hotové přihlašování k daným účtům
Konec listopadu: doladění celé aplikace a hardwaru květináče
Konec prosince: dokončení aplikace jako takové, nasazení na web, testování funkčnosti

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, výhradně s použitím uvedené literatury.

Upřímně děkuji vedoucímu projektu Paní Bc. Vlaďce Janů za vedení projektu a za všechny nervy které se mnou musela vydržet, dále bych chtěl poděkovat Zuzaně Dvořákové za psychyckou pomoc a za vytištění květináče a poskytnutí většiny hardwaru, dále Matějovi Neumannovi za pomoc se zapojením všech senzorů se kterými jsem si neporadil sám, dále Pavlovi Špryňarovi za veškerou podporu proti prokrastinaci a obecnou pomoc při programování, Kryštofu Kulhánkovi za pomoc s problémy s frameworkem Next.js se kterými jsem se setkal a nakonec všem ostatním již nezmýněných.

Anotace

Cílem projektu bylo vytvořit uživatelsky přívětivý dashboard pro zobrazování dat z inteligentního květináče nebo více květináčů, tento produkt je pro kohokoliv, kdo má problémy s udržením květin při životě.

Obsah

Úvod

Project_pot by měl sloužit všem lidem, kteří pokaždé když dostanou nebo si koupí jakoukoliv květinu, po nějakém čase květina uhyne. Můj projekt by měl uživateli pomoct v několika základních ale důležitých aspektech, které květina potřebuje k přežití. Když je nějaká konstanta ohledně květiny v nepořádku, například je málo světla v místě kde se květina nachází, v aplikaci uživatel uvidí hodnotu a následně bude moci reagovat dříve než květina pojde.

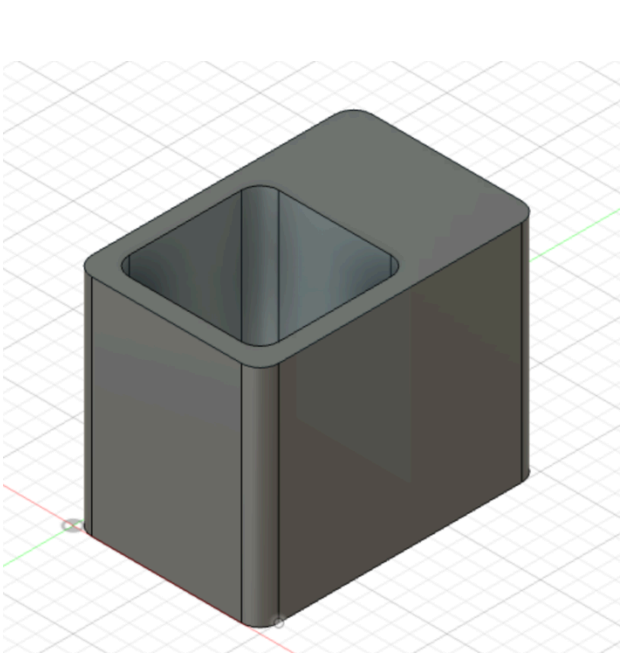
Osobní zkušenost

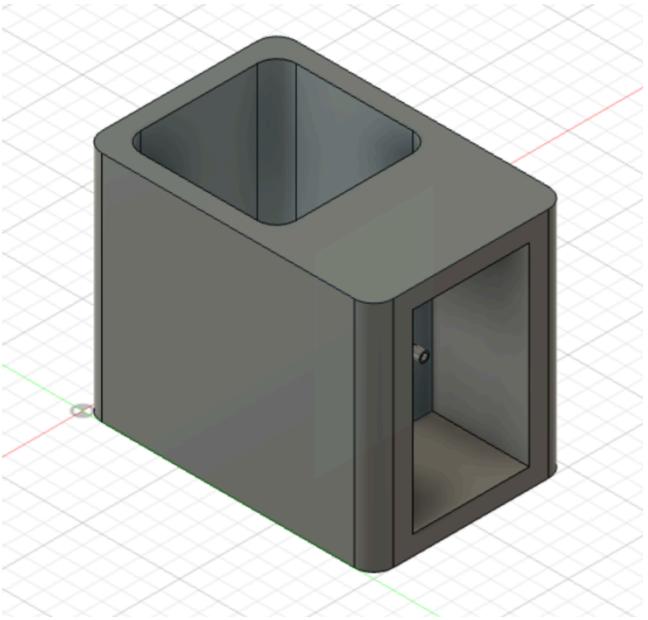
S přítelkyní jsme měli hned několik květin na společném stole a pokaždé, dříve nebo později, květina pošla. Když jsem pracoval na projektu, začal jsem si všímat že tenhle problém mají i spoustu dalších lidí kolem mě. Začal jsem uvažovat jak tento problém vyřešit, nikdo nechce aby květina kterou dostal k narozeninám nebo jako jiný dárek vidět uschlou nebo pošlou z jiného důvodu.

Hardware

Tělo mého květináče je vytištěno na 3d tiskárně z materiálu PLA. Samotný hardware je první verze mého projektu a chtěl jsem samotný květináč vytvořit levně, rychle a případně ho moct upravit. Proto jsem si květináč vymodeloval v programu Fusion 360 aby až bude květináč vytisknutý, mohl jsem dělat úpravy. Květináč jsem navrhnul tak, aby se vešel na běžný stůl člověka, který sedí u počítače a na stole nebude mít moc místa na velký květináč. Proto jsou jeho rozměry zdánlivě malé. Tím že jsem si ale květináč navrhl sám, nebyl by problém navrhnout květináč větší v kterémkoliv rozměru, pro kompatibilitu s větším množstvím květin.

Na to aby se z “obyčejného” květináče stal tzv. inteligentní květináč, do květináče jsem použil mikropočítač Raspberry Pi Zero 2w. Tento mikropočítač jsem si vybral z mnoha důvodů. Jedním z nich je široká kompatibilita s senzory určenými pro jiné mikropočítače. Všechny senzory, které používám v mém projektu, jsou od výrobce určené pro Arduino, ale díky 40-pin GPIO na tomto Raspberry Pi. Nebyl velký problém připojit tyto senzory k tomuto mikropočítači. Pro květiny jsou důležité hned několik hodnot. Jednou z nich je teplota a vlhkost vzduchu. Pro to jsem použil senzor DHT11 vyroben společností Adafruit. Další senzor od stejné společnosti je BH1750, senzor na měření intenzity světla. Poslední senzor který jsem použil je kombinace senzoru vlhkosti půdy, komparátoru LM393 a i2c konvertoru z analogového signálu na digitální. Důvod použití konvertoru byl, když jsem se pokusil poprvé připojit komparátor k GPIO a zjistil, že Raspberry, narozdíl od Arduina, neumí pracovat s analogovým signálem. Mezi poslední hardwarové komponenty patří bateriový systém který má sloužit jako powerbanka. Pro moje potřeby byl perfektní, protože měl už od výroby BMS(Battery Management System) a k tomu sloty na dvě baterie typu 18650. Tento typ baterií jsem si vybral pro jejich dobrý poměr kapacity ku fyzické velikosti baterie. V květináči jsou tyto baterie dvě a v případě nových baterií by tento prototyp měl být schopen existovat a fungovat ±1 měsíc mimo napájení.

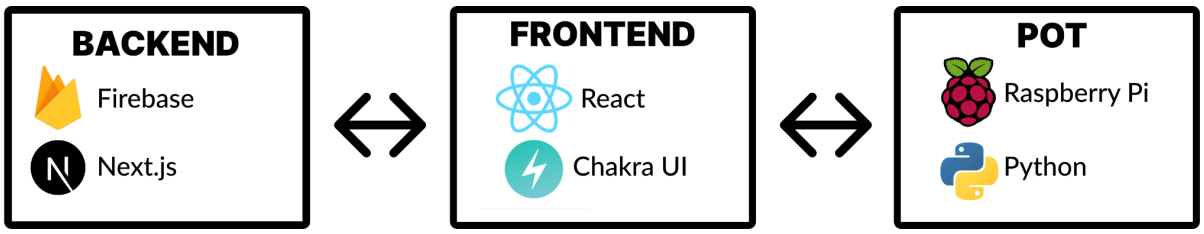




Webová stránka

Stránka funguje jako jednoduchý dashboard, kde uživatel uvidí všechny květináče které má navázané jako tabule s realtime údaji.

Architektura



Backend

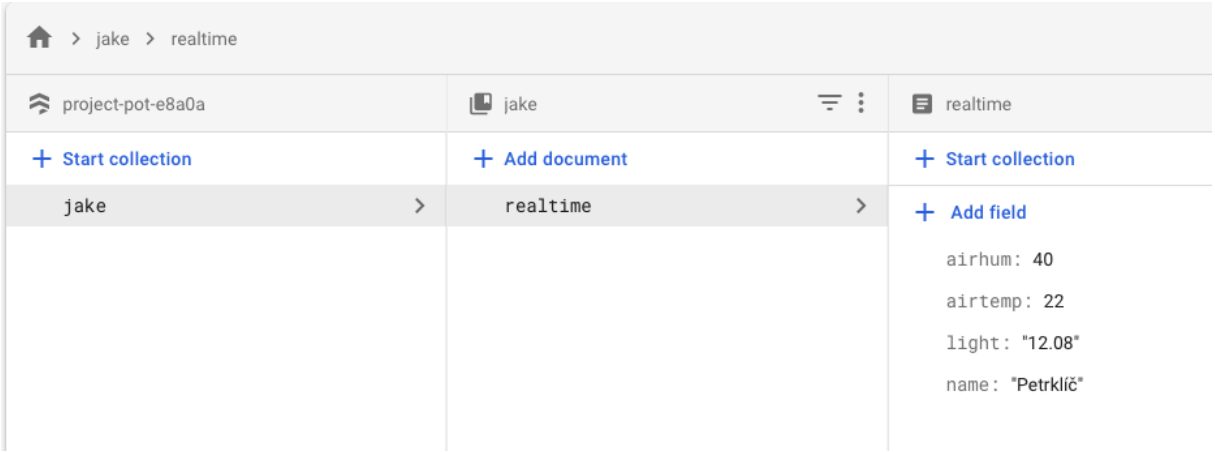
Backend aplikace je postaven na takzvané serverless architektuře. To znamená že jediné, na co potřebuji server, je hosting samotný a jinak se vše odehrává na cloudu nebo v mém případě v květináči.

Databáze

Pro databázi jsem využil Cloud Firestore od Firebase kterou vyvíjí společnost Google. Využil jsem ji protože mi byla doporučena pro její jednoduchou integraci, široké využití a vysokou spolehlivost.

Databázi jsem využíval na odesílání hodnot z květináče a následně pro jednoduché vyobrazení již zmíněných dat v UI stránky.

Později v průběhu práce na projektu jsem přidal i možnost loginu pomocí služby NextAuth.js, která mi pomohla jednoduše zprovoznit autentifikaci a následné ukládání uživatelů do databáze, funkce je plně implementována ale dokumenty s květináči se prozatím nespojí.

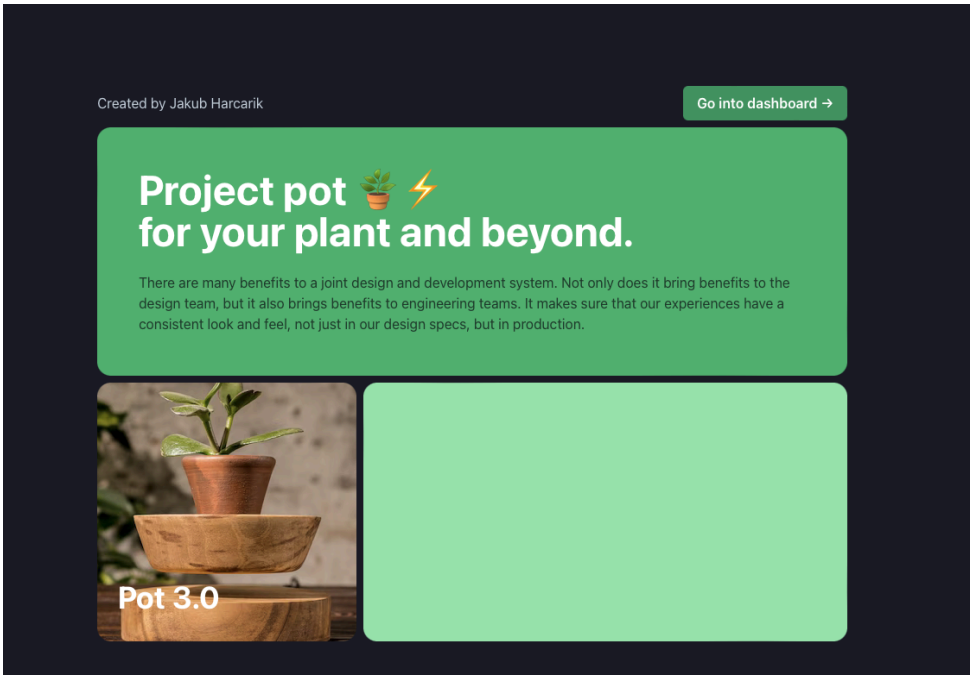


Květináč

Backend na květináči jsem psal v jazyku Python. Python jsem si vybral pro jeho jednoduchost a schopnost pracovat s knihovnami pro moje senzory. Kód na květináči se spustí automaticky potom, co uživatel květináč zapne. Kód začne automaticky odesílat data ze senzorů přečte každých ±5 sekund a odešle je do databáze.

Frontend

Celý frontend mého projektu se skládá z několika součástí. Tím že byl projekt již několikrát předělán, původně jsem veškerý frontend řešil pomocí MUI, knihovnou s mnoha ready-to-use komponenty pro React, postupně jsem ale přešel k jiné knihovně, ChakraUI, která má spoustu stejných vlastností jako MUI, ale dává uživateli víc možností jak již hotovou komponentu upravit přesně tak jak je potřeba. Ve finále jsem chtěl, aby kdokoliv mohl přijít na stránku a tušil co má dělat hned poprvé, když se na stránce ukáže. Design je jednoduchý, ale i přesto pohledný a funkční. Uživatel se nejprve připojí na landing page, kde jsou základní informace o projektu a o mě jakožto tvůrci, na landing page se nachází jediné tlačítko které přesměruje uživatele do dashboardu, kde až se přihlásí, uvidí květináče a jejich atributy.

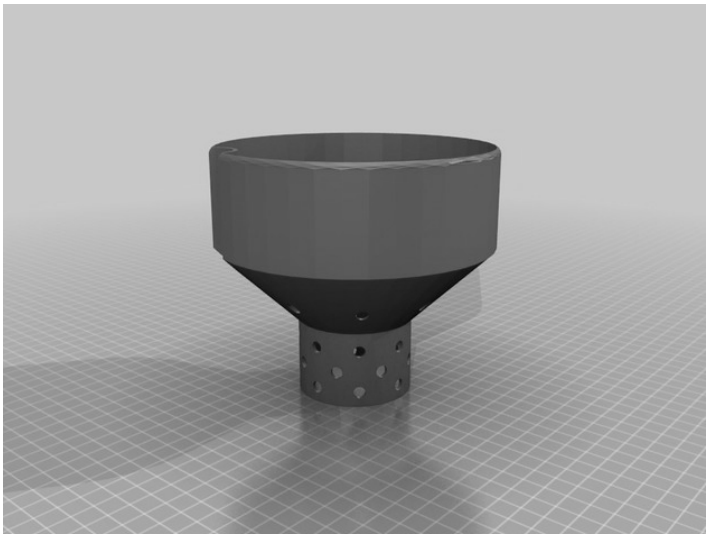


Další funkce

Mezi další funkce, již nezmíněné výše, patří například nachystání aplikace, aby uživatel když se přihlásí, mohl si přes klienta, míněno počítač, telefon nebo zařízení které je vybaveno technologií Bluetooth, mohl vyhledat okolní zařízení, když klient najde květináč, zadal by do formuláře SSID sítě a heslo k té samé síti a květináč by si tyto údaje přebral a připojil se k síti, kterou uživatel poskytl. Momentálně je SSID a

heslo nastavené pevně na květináči takže touto vlastností zatím nedisponuje, hardware i software toho ale schopný je.

Dále jsem chtěl, co se hardwarové části týče, přidat do květináče malý vklad. Momentálně je hlína a voda v jedné velké nádržce v květináči, ale v případě že uživatel květinu zalyje moc, květináč sice detekuje přebytek vlhkosti půdy, ale odstranit vodu ze zalité hlíny moc nejde. Proto do budoucna chci vytvořit další vklad ve tvaru trychtýře, který ke dnu květináče bude mít dírky na odtok přebytečné vody a v případě že uživatel květinu doopravdy přelije, bude moct jen vyndat celou květinu společně s hlínou a přebytek vody jednoduše odstranit.



Použité Technologie

React

React je knihovna pro vytváření uživatelského rozhraní (UI) v JavaScriptu. Je zaměřen na efektivní zobrazování změn v UI a umožňuje vytvářet komponenty, které jsou snadno znovupoužitelné a dobře oddělují logiku aplikace od prezentace. React také podporuje deklarativní přístup k psaní UI, což znamená, že programátoři specifikují, jak by měl UI vypadat v různých stavech, a React se stará o aktualizaci UI, když se tyto stavy mění. React je často používán v kombinaci s dalšími nástroji a knihovnami pro tvorbu moderních webových aplikací, jako jsou například React Router nebo Redux.

NextJS

Next.js je open-source framework pro React, který poskytuje nástroje pro vývoj webových aplikací s rychlým načítáním a optimalizací pro vyhledávače. Next.js umožňuje tvůrcům aplikací snadno vytvářet statické a server-side renderované stránky, automatické před-načítání dat a mnoho dalších funkcí, které usnadňují vývoj moderních webových aplikací. Next.js také nabízí podporu pro TypeScript, CSS-in-JS a mnoho dalších technologií. Díky své flexibilitě a širokému spektru funkcí je Next.js často používán v průmyslu pro vývoj webových aplikací různých druhů, od jednoduchých statických stránek až po velké a složité aplikace.

NodeJS

Node.js je open-source, multiplatformní prostředí pro běh JavaScriptu mimo webový prohlížeč. Node.js umožňuje vývojářům psát JavaScriptový kód na straně serveru a využívat v něm řádu modulů, knihoven a nástrojů pro různé účely jako například tvorbu webových aplikací, souborové operace, komunikaci s databázemi nebo IoT. Node.js je založen na architektuře událostí a asynchronním I/O modelu, což znamená, že dokáže zpracovávat velké množství požadavků současně bez nutnosti vytváření nových vláken nebo procesů. Díky své vysoké efektivitě a širokému spektru použití se Node.js stal velmi populární technologií, používanou v mnoha odvětvích jako je například vývoj webových aplikací, IoT nebo serverless technologie.

Chakra UI

Chakra UI je moderní, responzivní a přizpůsobitelná knihovna uživatelského rozhraní (UI) pro React aplikace. Je navržena tak, aby vývojářům usnadnila vytváření atraktivních a dobře přístupných uživatelských rozhraní s minimálním úsilím.

Jednou z hlavních vlastností Chakra UI je její jednoduchost a snadná použitelnost. Poskytuje sadu předdefinovaných komponent, které lze snadno kombinovat a upravovat, což umožňuje rychlé vytváření a úpravu UI prvků. Knihovna obsahuje širokou škálu komponent, včetně tlačítek, formulářových prvků, záložek, karet, modálních oken a mnoha dalších, které pokrývají běžné potřeby vývojářů.

Dalším klíčovým aspektem Chakra UI je jeho přizpůsobitelnost. Knihovna umožňuje snadno upravovat vzhled komponent pomocí vlastností jako barva, velikost, styl rámečků a mnoho dalšího. To znamená, že můžete snadno přizpůsobit vzhled a styl komponent tak, aby odpovídaly vašemu designu a značce.

Chakra UI se také zaměřuje na přístupnost. Každá komponenta je vyvíjena s ohledem na správné používání z hlediska přístupnosti, jako jsou dostupná klávesová zkratka, atributy ARIA a vhodné barevné schéma. To umožňuje vytvářet uživatelská rozhraní, která jsou snadno použitelná a přístupná pro všechny uživatele, včetně těch se znevýhodněním.

Výhodou Chakra UI je také její rozsáhlá dokumentace a aktivní komunita. Dokumentace obsahuje podrobný popis všech dostupných komponent, příklady kódu a návody, což usnadňuje začlenění a používání knihovny. Komunita Chakra UI je také velmi aktivní a poskytuje podporu, odpovědi na otázky a přispívá k rozvoji knihovny.

Firestore

Firebase Firestore je plně spravovaná cloudová NoSQL databáze od společnosti Google, která je navržena speciálně pro vývoj mobilních a webových aplikací. Firestore nabízí flexibilní schéma dat a umožňuje ukládat, synchronizovat a dotazovat se na data ve vaší aplikaci.

Jednou z hlavních výhod Firestore je jeho jednoduchost a přizpůsobivost. Vývojáři mohou snadno vytvářet kolekce a dokumenty, ukládat data ve formátu klíč-hodnota a provádět dotazy pomocí mocného jazyka pro dotazování. Firestore automaticky synchronizuje data mezi klientem a serverem, což umožňuje okamžitou propagaci změn napříč aplikací.

Firestore se vyznačuje také škálovatelností. Jako NoSQL databáze se Firestore snadno škáluje podle potřeb vaší aplikace. To znamená, že zvládne zpracovat nárůst uživatelů a objemu dat bez nutnosti starostí s infrastrukturou a škálováním. Firestore poskytuje vysokou dostupnost a spolehlivost, což zajišťuje neustálý přístup a bezpečné uložení vašich dat.

Další výhodou Firestore je jeho integrace s dalšími službami Firebase. Snadno můžete propojit Firestore s autentizací uživatelů, cloudovým úložištěm, notifikacemi a dalšími nástroji Firebase, což vám umožní efektivně vyvíjet plnohodnotné aplikace s funkcemi jako přihlášení uživatele, ukládání souborů nebo odesílání upozornění.

Javascript

JavaScript (často zkracován na JS) je programovací jazyk používaný pro vývoj webových aplikací. Je to skriptovací jazyk, který se provádí přímo ve webovém prohlížeči klienta, což umožňuje interaktivitu a dynamické chování webových stránek.

Jednou z klíčových vlastností JavaScriptu je jeho schopnost manipulovat s HTML a CSS, což umožňuje změnu a aktualizaci obsahu stránky v reálném čase. JavaScript také poskytuje pokročilé funkce pro práci s událostmi, validací formulářů, animacemi, asynchronním načítáním dat a komunikací s webovými API.

Raspberry Pi

Raspberry Pi je miniaturní jednodeskový počítač, který byl vyvinut s cílem podporovat výuku programování a elektroniky. Je to cenově dostupné zařízení, které se velikostí a vzhledem podobá kreditní kartě, ale přesto nabízí výkon a funkce počítače.

Hlavním cílem Raspberry Pi je poskytnout uživatelům nástroj, který umožňuje experimentování, vývoj a vytváření různých projektů. Raspberry Pi je vybaveno procesorem, pamětí, vstupy a výstupy, a také operačním systémem, který je obvykle založený na Linuxu.

Jednou z klíčových výhod Raspberry Pi je jeho rozšiřitelnost. Na desce jsou k dispozici různé konektory a porty, které umožňují připojení dalších periferních zařízení, jako jsou senzory, displeje, kamery, reproduktory a další. To umožňuje uživatelům přizpůsobit Raspberry Pi pro konkrétní účely a rozšířit jeho funkce podle svých potřeb.

Závěr

Vytvořil jsem aplikaci, která pouze nastiňuje vizi kterou jsem měl při zadávání tohoto projektu, nicméně v aplikaci je naimplementované to nejdůležitější. Pro nasazení na hosting jsem použil Vercel.

Web není hotový pro veřejnou publikaci, chybí mu spoustu věcí aby toho byl schopen ale je dostatečně nachystaná pro zatím malou skupinu kutilů, kteří tento problém mají.

Při práci na webu jsem narazil na spoustu bariér. Jednou to byla že jsem z Raspberry Pi odesílal data do Firebase Realtime Database, ze které jsem nebyl schopný data přečíst, poté to byl Firebase Authentication, kterou jsem nahradil již zmíněnou knihovnou NextAuth.js. Obecně projekt není hotový a chci na něm pracovat jako na svém osobním projektu i po dokončení školy.

Až bude někdy web téměř dokonalý, chtěl bych zkusit projekt předprezentovat širšímu publiku, míněno hardware i software, určitě bych nebyl sám kdo měl stejný problém. Dále bych chtěl projekt převést i na mobilní aplikaci, důvod je očividný, takový to dashboard si asi moc lidí z desktopu zobrazovat nebude.

Seznam použité literatury

[1] React - A JavaScript library for building user interfaces. *React* [online]. [cit. 2022-01-22]. Dostupné z: https://reactjs.org/

[2]*Next.js: Docs* [online]. San Francisco , U.S.: web, 2016 [cit. 2023-03-31]. Dostupné z: https://nextjs.org/docs/

[3]*NextAuth.js* [online]. San Francisco, CA: Vercel, 2023 [cit. 2023-06-15]. Dostupné z: https://next-auth.js.org

[4]*Firebase* [online]. worldwide: web, 2012 [cit. 2023-03-31]. Dostupné z: https://firebase.google.com/docs/guides

[5]*Pip* [online]. worldwide: web, 2011 [cit. 2023-03-31]. Dostupné z: https://pypi.org/project/pip/

[6]*StackOverflow* [online]. worldwide: web, 2008 [cit. 2023-03-31]. Dostupné z: https://stackoverflow.com

[7]*ChakraUI* [online]. Nigeria: Segun Adebayo, 2023 [cit. 2023-06-15]. Dostupné z: https://next-auth.js.org