**JELGAVAS TEHNIKUMS**

**Niks Barščausks**

**Laikapstākļu datu ievākšana un vizualizācija lietotājsaskarsmē**

Kvalifikācijas darbs

kvalifikācijas ieguvei

programmēšanas tehniķis

#### Darba izpildītājs: Niks Barščausks

Jelgava

2022

#### ANOTĀCIJA

# Niks Barčausks – kvalifikācijas darbs.

Šīs kvalifikācijas darba mērķis ir izveidot mājas lapu kas uzkrās un vizualizēs laikapstākļu datus kas tiks ievākti sadarbībā ar Alekša Baloža darbu, kas ievāc laikapstākļu datus izmantojot mikrokontrolierus un nosūtīs datus uz serveri izmantojot *RestAPI JSON* formātā. Tiks pielietots *ReactJS* priekš lietotāj interfeisa un *ExpressJS* priekš servera interfeisa. Kā arī tiks izmantots *MongoDB Cloud* priekš datu uzkrāšanas *JSON* formātā. *ReactJS* sazināsies ar *ExpressJS* caur *RestAPI*.

#### ANNOTATION

Niks Barčausks – qualification work.

This qualification 's goal is to create a website which will visualize and graph data collected from mini weather station created by Aleks Balodis. By using NextJS framework for Web-app installation, the app will receive data from associated co-project and store it in MongoDB Cloud database in JSON format. Front-end and Back-end will be handled by NextJS and other Javascript frameworks listed in the documentation.

#### SATURS

[Niks Barčausks – kvalifikācijas darbs. 2](#_Toc105670632)

[1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA 6](#_Toc105670633)

[1.1. Ievads 6](#_Toc105670634)

[1.1.1. Dokumentācijas mērķis un darba mērķis 6](#_Toc105670635)

[1.1.2. Produkta Sfēra 6](#_Toc105670636)

[1.1.3. Dokumentācijas pārskats 6](#_Toc105670637)

[1.2. Vispārīgs apraksts 7](#_Toc105670638)

[1.2.1. Produkta funkcijas 7](#_Toc105670639)

[1.2.2. Produkta pielietojums 7](#_Toc105670640)

[1.2.3. Mērķauditorija 7](#_Toc105670641)

[1.3. Nefunkcionalās prasības 8](#_Toc105670642)

[1.3.1. Kopējās nefunkcionālās prasības 8](#_Toc105670643)

[1.3.2. Mājas logs (*Homepage*). 10](#_Toc105670644)

[1.3.3. Tabulas logs (*Data*). 11](#_Toc105670645)

[1.3.4. Grafika logs (*Graphs)* 12](#_Toc105670646)

[1.4. Funkcionālās prasības 14](#_Toc105670647)

[1.4.1. Galvenā lapa 14](#_Toc105670648)

[1.4.2. Datu tabulas lapa 15](#_Toc105670649)

[1.4.3. Grafika lapa 15](#_Toc105670650)

[1.4.4. Filtra forma 15](#_Toc105670651)

[1.4.5. Navigācijas josla 15](#_Toc105670652)

[2. IZSTRĀDEI IZMANTOTĀS TEHNOLOĢIJAS UN PROGRAMMATŪRA 16](#_Toc105670653)

[2.1. Izmantotās programmas koda veidošanai 16](#_Toc105670654)

[2.2. Izmantotās programmēšanas vides 17](#_Toc105670655)

[2.3. Komponenšu saraksts 17](#_Toc105670656)

[3. Testēšana 18](#_Toc105670657)

[4. Programmas *API* 18](#_Toc105670658)

[4.1. Aplikācijas lapu funkcijas 18](#_Toc105670659)

[4.2. RestAPI ceļi 19](#_Toc105670660)

[4.3. Datu bāzes funkcijas 19](#_Toc105670661)

[5. Programmas darbības shēma 19](#_Toc105670662)

[5.1. Lietotājsaskarsme 19](#_Toc105670663)

[5.2. Servera saskarsme 20](#_Toc105670664)

# PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

## Ievads

### Dokumentācijas mērķis un darba mērķis

Izskaidrot un iepazīstināt ar kvalifikācijas darba funkcionalitāti un tehniskām prasībām. Darba mērķis ir izveidot mājaslapu, kas ar *RestAPI* ievāc laikapstākļu datus un projektē tabulas ar šiem datiem.

### Produkta Sfēra

Produkts ietilpst Interneta mājaslapas produkcijā. Produkta priekšrocība ir lokāli pieejamie laikapstākļu dati, bez nepieciešamības izmantot datus, kas nav lokalizēti.

### Dokumentācijas pārskats

## Vispārīgs apraksts

### Produkta funkcijas

Šī produkta funkcija ir nodrošināt laikapstākļu uzkrāto datu noformēšanu lietotājiem vienkārši saskatāmā grafikā, ar kuru var novērot laikapstākļu izmaiņas, balstoties uz konkrētiem datiem.

Mājaslapa nodrošina:

* Laikapstākļu datu saglabāšanu.
* Detalizētu datu tabulu.
* Informācijas grafiku.
* Datu filtrēšanu gan tabulā gan grafikos.

### Produkta pielietojums

Šis produkts ir pielietots laikapstākļu novērošanā un prognozēšanai pēc iepriekšējiem datiem.

### Mērķauditorija

Šis produkts ir paredzēts lietotājiem, kam ir ļoti svarīgi sekot laikapstākļu izmaiņām un sekot tendencēm, ar kurām ir iespējams prognozēt laikapstākļus. Kā arī lietotājiem, kas ir vēlas sekot laikapstākļiem un apskatīt lokālās tendences.

## Nefunkcionalās prasības

### Kopējās nefunkcionālās prasības

* **Loga izskats**

Loga fons : attēls (1. attēls).

Loga elementi centrēti horizontāli un vertikāli.

##### 

##### 1. att. Loga fons

* **Navigācijas josla** (skat. 2. att)

Navigācijas josla satur 3 navigācijas pogām :

* Weather Control – noved uz galveno lapu
* Data – noved uz tabulas logu (skat. 7. att.)
* Graphs – noved uz grafika logu (skat. 8. att.)

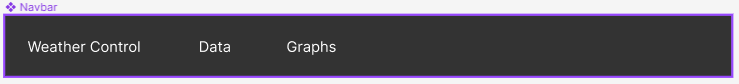
Fona krāsa: melna (#333333);

Teksta krāsa: balta (white; #000000);

Teksta izmērs: vidējs (medium)

Pogu fona krāsa: melna (#333333);

Pogu krāsa zem kursora: melna (#111111);



##### 2. att **Navigācijas josla**

* **Lapas kartiņas** (skat. 3. att)

Kartiņas fons: pelēka (#dad7d4);

Teksta krāsa: melna (#0f0f0f);

Malas noapaļotas.

Satur galveno lapas mēdiju.

Centrēts horizontāli un vertikāli lapai.

Tiek izmantota kā lapas konteiners(*container*).



3. att **Lapas kartiņā**

* **Filtra elements** (skat 5. att)

Fona krāsa: balta (#f0f0f0);

Teksta krāsa: #0d0f0f;

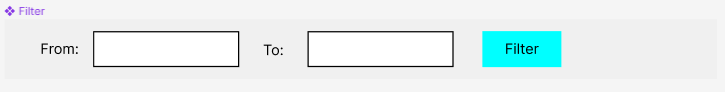
Filtra pogas krāsa: gaiši zils (*aqua*);

Pogas nospiešanas brīdī tiek izsaukta funkcija *filter();*

2 datuma ievadlauki: filtra diapzons *(From, To*);

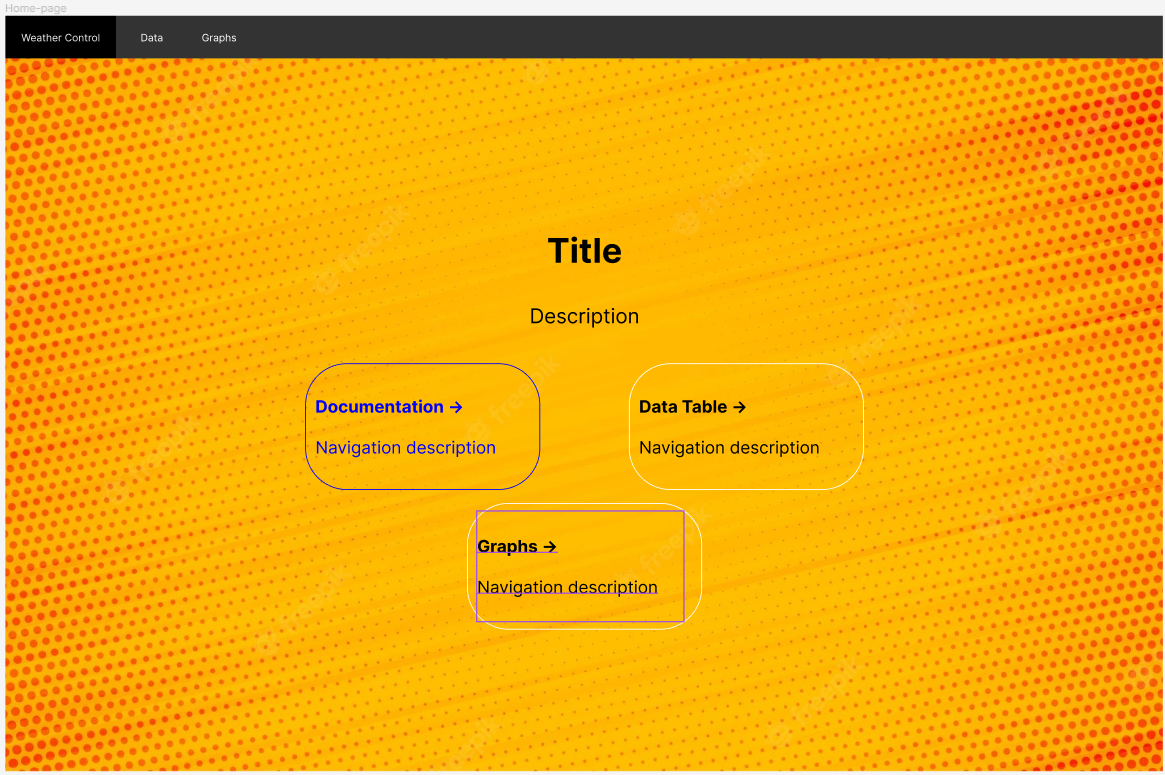
Kārtots no lapas labās puses;

Tiek izmantots Datu tabulas un Grafikas lapā



4. att **Filtra elements**

### Mājas logs (*Homepage*).

****

**5. att Mājas Loga izskats *(wireframe)***

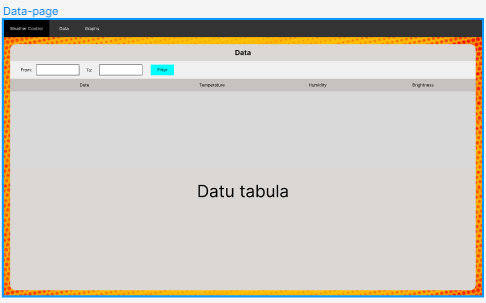
Galvenā lapa sastāv no Virsraksta, paskaidrojuma un 3 navigācijas pogām :

* Documentation – noved uz aplikācijas dokumentāciju.
* Data table – noved uz tabulas logu (skat. 6. att.).
* Graphs – noved uz grafika logu (skat. 7. att.).

Teksta krāsa: melna (#000000);

Visi elementi ir centrēti horizontāli un vertikāli;

### Tabulas logs (*Data*).



6.att **Datu tabulas loga izskats *(wireframe)***

Datu tabulas logs satur tabulas elementu (skat. 7. att.).

Datu tabulas logs ietver filtra elementu, kas atrodas starp virsrakstu un tabulu.

Lietotājam ir atļauts filtrēt datus izmantojot filtra elementu.

Ielādējot Graphs logu tiek izsaukts fetch() pieprasījums un grafiks tiek aizpildīts ar datiem.

* **Tabulas elements** (skat 7. att)

Fona krāsa: pelēka (#dad7d4);

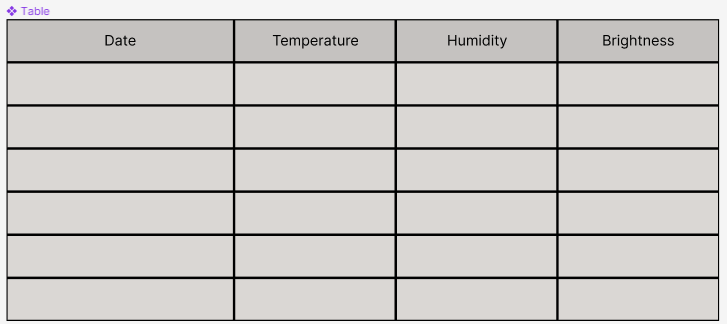
Tabulas 1. rindas fona krāsa: pelēka (#c5c2c0);

Tabulā ir 4 kollonas un neierobežotas rindas.

Temperatūras mērvienība ir apzīmēta ar C (celsijiem).

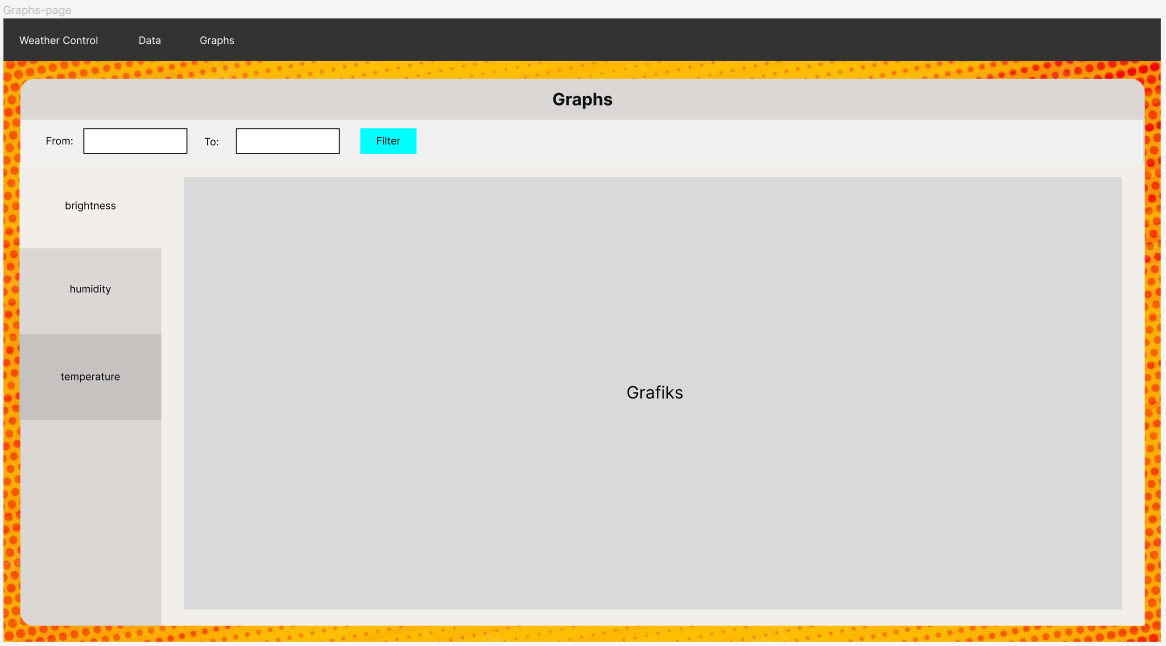
Mitruma mērvienība ir apzīmēta procentos.

Spilgtuma mērvienība ir apzīmēta ar lux(lukss)



7.att **Datu Tabula**

### Grafika logs (*Graphs)*



8. att **Grafika loga izskats *(wireframe)***

Grafika logs sastāv no grafikas elementa (skat. 9. att.) un filtra elementa.

Noklikšķinot vienu no trīm izvēlnēm, kas atrodas kreisajā pusē, mainās grafiks

Ielādējot Graphs logu tiek izsaukts fetch() pieprasījums un grafiks tiek aizpildīts ar datiem.

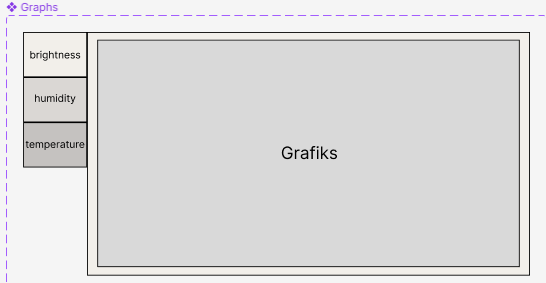
Ievadot datuma diamzonu un nospiežot filter tiek izpildīta filter() funkcija.

* **Grafikas elements** (skat. 9. att.)

Grafikas elementā ietilpst 3 pogas, kas atrodas grafikam pa kreisi, pogu krāsas sakrīt ar grafika fona krāsu.

Pogas:

* + Brightness – nomaina grafiku uz spilgtuma mērijumiem. Krāsa: (#f2efea)
  + Humidity – nomaina grafiku uz mitruma mērijumiem. Krāsa: (#dad7d4)
  + Temperature – nomaina grafiku uz temperatūras mērijumiem. Krāsa: (#c5c2c0)



9.att **Grafika elements**

## Funkcionālās prasības

Programmatūras galvenā prasība ir savienojums ar datu bāzi un nepārtraukta informācija par laikapstākļu izmaiņām. Kā arī ir jāpielieto *Javascript* bibliotēkas, kas atvieglo programmatūras optimizāciju, atļaujot lietotājiem, atvērt mājaslapu ļoti ātri un bez ilgas gaidīšanas. Tiek arī pielieti dažādi funkcionāli triki, kas palīdz samazināt lādēšanas ilgumu un atvieglo lietotāja sadarbību ar programmatūru.

### Galvenā lapa

Atverot mājaslapu jāparādās galvenajai lapai. Galvenajā lapā jāparādās Virsrakstam ar aprakstu par mājaslapu un 3 navigācijas pogām:

* *Documentation* – Aizved uz dokumentāciju.
* *Data* – Aizved uz datu tabulas lapu.
* *Graphs* – Aizved uz grafikas lapu

Mājaslapas augšā atrodas navigācijas josla.

### Datu tabulas lapa

Lapa atverama izmantojot hipersaites, vai caur saiti pārlūkā. Lapa parāda:

* Datu tabulu. Jābūt redzamam Temperatūrai, Mitrumam, Spilgtumam
* Filtra formu
* Navigācijas joslu

### Grafika lapa

Lapa atverama izmantojot hipersaites, vai caur saiti pārlūkā. Lapa parāda:

* Laikapstākļu grafiskos zīmējumus. Jābūt redzamam Temperatūrai, Mitrumam, Spilgtumam
* Filtra formu
* Navigācijas joslu

### Filtra forma

Parādot datus jābūt filtra formai, kas atlasa datus atkarībā no formas datiem.

Formā ir 2 ievad lauki un viena poga Filter. Ievadlaukos jaievada datums: pirmajā laukā datums no, otrajā laukā datums līdz. Nospiežot filter pogu tiek atlasīts diapzons datiem. Lapa parāda šo diapzona datus.

### Navigācijas josla

Atrodas katras lapas augšpusē un nospiežot hipersaiti tiek pārlādēta lapa uz izvēlēto aplikācijas lapu. Navigācijas poga sastāv no 3 pogām:

* Weather Control – Noved uz Galveno lapu
* Data – Noved uz Datu tabulas lapu
* Graphs – Noved uz grafikas lapu

# IZSTRĀDEI IZMANTOTĀS TEHNOLOĢIJAS UN PROGRAMMATŪRA

Mājaslapas jeb web-aplikācijas izstrāde notiek *Javascript* programmēšanasvalodā. Pati mājaslapa tiek balstīta uz *NodeJS* dzinēju, kas darbina *Javascript* Pirmkodu ar ES6, vai ES7 tehnoloģiju, atkarībā, no kuras versijas tiek, izmantota. *NodeJS* dzinējam ir ļoti plašs bibliotēku saraksts, kur atvērta pirmkoda bibliotēkas tiek globāli izdalītas caur *Node Package Manager (NPM),* kas ļauj izveidot dažādus koda šablonus, ļaujot atvieglot programmatūras izveidi un optimizāciju.

Izstrādei tiek izmantots *Visual studio code* pirmkoda redaktors, jo tajā ir pieejami ļoti daudz paplašinājumu, kuri palīdz ar koda paredzamību un labošanu, atvieglojot kodēšanas procesu, un tai pašā brīdī tiek palaists terminālis, uz kura *NodeJS* dzinējs tiek kontrolēts. Tiek arī izmantos *Github* serviss pirmkoda uzglabāšanai un pārvietošanai uz citiem datoriem, saglabājot to pašu vidi starp datoriem.

## Izmantotās programmas koda veidošanai

* Visual studio code – Ir galvenā programma, kura tika pielietota pirmkoda veidošanai. Viusal studio code ir integrētā izstrādes vide ar attīstītu paplašinājumu klāstu.

Versija: v1.63.2

* Git – Ir pirmkoda kontroles vide, kur notiek versiju kontrole un koda glabāšanu, kā arī sadarbības iespēja ar citiem koda veidotājiem.

Versija: v2.32.1.

* Opera GX – Pārlūks ar ko tika testēta aplikācija.

Versija: v87.0.4390.56

## Izmantotās programmēšanas vides

* NodeJS – Ir Javascript dzinēja vide, kas nodrošina Javascipt koda darbību. Izmantots kā galvenais aplikācijas dzinējs. Atbild par aplikācijas darbību gan servera gan klienta pusē.

Versija: v17.3.0

* Node package manager (NPM) – Ir NodeJS sub-programma, kas ir atbildīga par bibliotēku atjaunināšanu un palaišanu. Izmantojot NPM var pievienot programmai dažādas bibliotēkas no tiešsaistes bibliotēkas.

Versija: v8.3.0

* Kods sastāv no Javascript, CSS un HTML programmēšanas valodās. Viss servera kods sastāv no Javascript.

Versijas:

* + Javascript: vES7
  + CSS: v3
  + HTML: v5

## Izmantotās NPM paketes

* NextJS - Ir Galvenā komponente kas balstās uz *ReactJS* priekš lietotāja interfeisa un *back-end* nodrošināšana. *NextJS* ir izveidots, lai varētu atvieglot *ReactJS* integrēšanu ar *back-end.* Šī *framework* priekšrocības ir atvieglota lapu vadīšana un atvieglota lietotnes strukturizācija.

Versija: 12.1.0.

* *Mongoose* – Ir Datubāzes komponente, kas atbild par datu saglabāšanu un lasīšanu. Šī bibliotēka izmanto *MongoDB* kā galveno datu bāzi un papildina to ar shēmām kas nodrošina mainīgo tipu saderību datubāzē, jo oriģinālu MongoDB strādā kā *JSON*.

*Versija*: v6.2.6.

* *Chart.js –* Komponente kas nodrošina vizuālo izskatu grafikiem

*Versija:* v3.7.1.

# Programmas *API*

## Aplikācijas lapu funkcijas

* Home() – Galvenās lapas funkcija. Tiek izsaukta atverot mājaslapas hipersaiti : <https://www.graw.io>. Funkcija atgriež hipersaites uz aplikāciju lapām un virsrakstu ar paskaidrojumu.
* Data() – Datu tabulas funkcija. Tiek izsaukta atverot linku <https://www.graw.io/data/>. Funkcija atgriež datu tabulu, kura tiek aizpildīta ar “/api/data/” pieprasījumu no servera.
* Graphs() – Grafiku zīmējumu funkcija. Tiek izsaukta atverot linku [https://www.graw.io/graphs /](https://www.graw.io/graphs%20/). Funkcija atgriež grafiku ar aizpildītiem datiem no “/api/data/” pieprasījumu no servera.

## Universālās funkcijas

* Filter() – Filtra funkcija kas tiek izmantota filtrējot lapas *data* mainīgo ar filtrētiem datiem. Tiek izsaukta nospiežot filter pogu

## RestAPI ceļi

“/api/data” – GET metode atgriež datus. Izsaucot šo linku kopā ar host atrašanās vietu tiek atgriezti nefiltrēta datu objekts, kas tiek atgūti no datubāzes. GET funkcijas formāts:

[{“\_id” : ObjectID, “date”: Date, “temperature” : Number, “humidity” : Number, “brightness” : Number}…]

“api/data/{datums}/{datums}” – GET metode atgriež datus kas ir filtrēti no šī pieprasījuma *query.*

“/api/data/upload” – POST metode, kas pieprasījuma laikā pārbauda ienākošo pieejas atslēgu, kas ir nodota mikrokontrolierim. Atslegas sakrišanas gadījumā tiek saglabāti dati kas ir nodoti data objektā. POST funkcijas formāts:

{ headers: { 'Content-Type': 'application/json' },

body: { “access”: TOKEN, “data”: DATA}

method: “POST”

}

## Datu bāzes funkcijas

add({date, temperature, humidity, brightness}) – Funkcija tiek izsaukta no servera, kad tiek izmantots “/api/data/upload” POST pieprasījumu. POST pieprasījuma *body* aizpilda funkcijas parametrus. Izsaucot funkciju tiek pārbaudīti parametri un ievietoti datu bāzē.

list({from, to}) – Funkcija tiek izsaukta, kad tiek izmantos “/api/data/” GET pieprasījums. Funkcias parametri ir paredzēti datuma filtrēšanai. Funkcijas parametri var arī būt nenorādīti, tad tiks atgriezti visi dati bez filtra.

# Testēšana

* **Filter() testēšana.**

Ievadīšanas parametrs ir objekts ar 2 datuma vienumiem, izsaukuma gadījuma atgriež datu kopu datumu diapzonā.

Funkcija pārbauda parametra formātu. Kļūdas gadījumā dati netiek filtrēti un tiek atgriezti nefiltrēti dati.

# Programmas darbības shēma

## Lietotājsaskarsme

## 

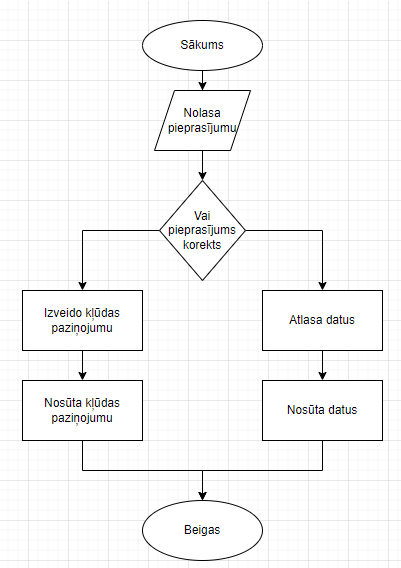
?. att Lietotāja saskarsmes darbības shēma

Lietotājsaskarsme tiek nodoršināta ar NextJS bibliotēku kura kontrolē html un css priekš vizuālā izskata un saskarsmi ar lietotāju izmantojot pogas un teksta lodziņus.

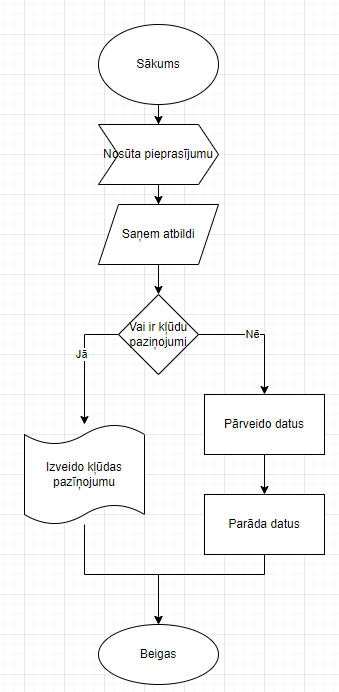
Caur lietotājsaskarsmi mājaslapa sazinās ar serveri caur *NextJS hooks* un *RestAPI.*

Dati tiek ielādēti līdz ar lapas ielādi un tiek izkārtoti attiecīgi lapas noformējumam ar JSON tipa mainīgo. Dati tiek atjaunoti katru reizi, kad tiek filtrēti vai lapa tiek pārlādēta.

## Servera saskarsme



?.att *back-end­* servera darbības shēma



?.att front-end servera darbības shēma

Serveris arī tiek uzturēts ar NextJS bibliotēku un kontrolē visus ienākošos pieprasījumus no *RestAPI* un *NextJS hooks.*

Serveris arī apstrādā pieprasījumus no mikrokontroliera, kas uzskaita datus un ieraksta datubāzē. Datubāze atrodas mākonī – *MongoDB Cloud.* Lai Izveidotu jaunu ierakstu caur serveri ir nepieciešama digitālā atslēga kuru zin tikai kontrolieris un serveris.