DOKUMENTASI PROYEK “WAREGARAGE”

OPREC NETLAB 2023

Logo

Description automatically generated

Muhammad Rizky Utomo, 2106731320

Universitas Indonesia

1. Topologi

Ini adalah repositori dari aplikasi WareGarage, sebuah aplikasi yang memudahkan Anda dalam melakukan manajemen terhadap gudang penyimpanan.

Sebuah perusahaan dapat membuat beberapa tempat penyimpanan. Tempat penyimpanan tersebut memiliki kapasitas tertentu dan dapat dibagi menjadi beberapa section yang tiap sectionnya menyimpan barang tertentu. Jika barang tidak terlalu banyak dan masih ada ruang di dalam gudang, alokasi untuk sebuah jenis barang bisa ditambah dan dikurang. Barang dapat disimpan dan ditarik di mana tiap barang memiliki ID yang menyesuaikan jenisnya dan tanggal masuk. Tiap gudang memiliki sebuah akun pengurus dengan sebuah akun utama yang bisa membuat perusahaan dan mengakses semua gudang (OWNER dan MANAGER). Perusahaan juga memiliki data keluar masuk barang yang ditandai dengan tanggal masuk, tanggal keluar, dan kapasitas yang keluar.

Untuk menjalankan database gudang dan konten-kontennya, pengembang menggunakan PostgreSQL yang berjalan di server cloud milik Neon. Database ini terhubung ke backend berupa Node.js yang terhubung ke server PostgreSQL menggunakan pg. Backend ini menjadi server untuk frontend berupa aplikasi Android yang dikembangkan dengan Android Studio.

ERD

A picture containing diagram, plan, pixel, design

Description automatically generated

UML

A picture containing text, screenshot, diagram, number

Description automatically generated

Flowchart

A picture containing diagram, text, sketch, drawing

Description automatically generated

1. Tabel SQL

Dengan kriteria seperti di bagian I, ini adalah tabel SQL-nya.

Employee

A screenshot of a computer

Description automatically generated with low confidence

Company

A picture containing text, screenshot, font, number

Description automatically generated

Warehouse

A picture containing text, screenshot, font, number

Description automatically generated

Inout\_records

A picture containing text, screenshot, font, number

Description automatically generated

Section

A screenshot of a computer

Description automatically generated with low confidence

Commodity

A picture containing text, screenshot, font, number

Description automatically generated

1. Konfigurasi SQL, Android, dan Node.js

SQL

create type employee\_type as enum ('MANAGER', 'OWNER');

create table employee (

id bigserial primary key,

username text unique not null,

password text not null,

name text not null,

company\_id bigint not null,

role employee\_type not null

);

create table company (

id bigserial primary key,

name text unique not null

);

create table warehouse (

id bigserial primary key,

address text not null,

name text not null,

company\_id bigint not null,

manager\_id bigint,

rem\_capacity bigint not null

);

create table inout\_records (

id bigserial primary key,

company\_id bigint not null,

warehouse text not null,

section text not null,

commodity\_id text not null,

name text not null,

type commodities not null,

volume bigint not null,

in\_time date,

out\_time date

);

create type commodities as enum ('LIVE', 'FOOD', 'EARTH', 'ELECTRONIC', 'PHARMA', 'FURNITURE', 'TRANSPORT');

create table section (

id bigserial primary key,

warehouse\_id bigint not null,

name text not null,

commodity\_type commodities not null,

rem\_capacity bigint not null

);

create table commodity (

id bigserial primary key,

commodity\_id text unique,

section\_id bigint not null,

volume bigint not null,

name text not null

);

Node.js (Index.js)

const express = require('express')

const app = express()

const { Client } = require('pg')

const bcrypt = require('bcrypt');

const bp = require('body-parser')

app.use(express.json())

const db = new Client({

connectionString: 'postgres://muhammad.rizky18:HEGdpPmn8S9B@ep-hidden-mode-314042.ap-southeast-1.aws.neon.tech/proyek\_oop',

sslmode: "require",

ssl: true

})

db.connect((err)=>{

if(err){

console.log(err)

return

}

console.log('Database berhasil terkoneksi')

})

app.post('/register-owner',(req,res)=>{

const username = req.body.username,

password = req.body.password,

name = req.body.name,

company\_name = req.body.company\_name

bcrypt.hash(password, 8, (err, hashedPassword) => {

if (err) {

console.log(err)

res.status(400).send

return

}

const query1 = `insert into company(name) values ('${company\_name}')`;

db.query(query1, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

const query = `insert into employee(username, password, name, role, company\_id) values ('${username}', '${hashedPassword}', '${name}', 'OWNER', (select id from company where name = '${company\_name}'))`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

});

});

res.status(200).send

})

app.post('/register-manager',(req,res)=>{

const username = req.body.username,

password = req.body.password,

name = req.body.name,

company\_id = req.body.company\_id

bcrypt.hash(password, 8, (err, hashedPassword) => {

if (err) {

console.log(err)

res.status(400).send

return

}

const query = `insert into employee(username, password, name, role, company\_id) values ('${username}', '${hashedPassword}', '${name}', '${role}', cast('${company\_id}' as bigint)`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

});

})

/\*

app.post('/change-employee-detail',(req,res)=>{

const {name, username, password, employee\_id} = req.body

bcrypt.hash(password, 8, (err, hashedPassword) => {

if (err) {

console.log(err)

res.status(400).send

return

}

const query = `update employee set name = '${name}', username = '${username}', password = '${hashedPassword}' where id = cast('${employee\_id}' as bigint)`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

});

res.status(200).send

})\*/

app.post('/login',(req,res)=>{

const username = req.body.username

const password = req.body.password

bcrypt.hash(password, 8, (err, hashedPassword) => {

const query = `SELECT \* from employee WHERE username='${username}' AND password='${hashedPassword}'`; //query ambil data user untuk login

bcrypt.compare(password, hashedPassword, (err, isMatch) => {

if( err ) {

res.status(404).send

return err;

}

// If password matches then display true

console.log(isMatch);

db.query(query, (err, results) => {

if (err) {

console.log(err)

res.status(404).send

return

}

res.status(200).send(JSON.stringify(results))

});

});

});

})

app.post('/get-company-name',(req,res)=>{

const company\_id = req.body.company\_id

db.query(`SELECT \* FROM company WHERE id = ${company\_id}` ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.post('/company',(req,res)=>{

const company\_id = req.body.company\_id

db.query(`SELECT \* FROM warehouse WHERE company\_id = ${company\_id}` ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.post('/create-warehouse',(req,res)=>{

const address = req.body.address,

name = req.body.name,

company\_id = req.body.company\_id,

rem\_capacity = req.body.rem\_capacity

const query = `insert into warehouse(address, name, company\_id, rem\_capacity) values ('${address}', '${name}', cast('${company\_id}' as bigint), cast('${rem\_capacity}' as bigint))`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

})

app.post('/delete-warehouse',(req,res)=>{

const warehouse\_id = req.body.warehouse\_id

const query01 = `select \* from sections where warehouse\_id = ${warehouse\_id}`;

db.query(query01, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

if(results != NULL){

res.status(500).send

return

}

const query02 = `delete from warehouse where id = ${warehouse\_id}`;

db.query(query02, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

});

})

app.post('/get-manager',(req,res)=>{

const company\_id = req.body.company\_id

db.query(`SELECT \* FROM employee WHERE company\_id = ${company\_id} and role = 'MANAGER` ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.post('/get-manager-name',(req,res)=>{

const manager\_id = req.body.manager\_id

db.query(`SELECT \* FROM employee WHERE id = ${manager\_id}` ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.post('/assign-manager',(req,res)=>{

const manager\_id = req.body.manager\_id,

warehouse\_id = req.body.warehouse\_id

const query = `update warehouse set manager\_id = ${manager\_id} where id = ${warehouse\_id}`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

})

app.post('/unassign-manager',(req,res)=>{

const warehouse\_id = req.body.warehouse\_id

const query = `update warehouse set manager\_id = NULL where id = ${warehouse\_id}`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

})

app.post('/warehouse',(req,res)=>{

const warehouse\_id = req.body.warehouse\_id

db.query(`SELECT \* FROM section WHERE warehouse\_id = ${warehouse\_id}` ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.post('/create-section',(req,res)=>{

const warehouse\_id = req.body.warehouse\_id,

name = req.body.name,

commodity\_type = req.body.commodity\_type,

rem\_capacity = req.body.rem\_capacity

const query1 = `insert into section(warehouse\_id, name, commodity\_type, rem\_capacity) values (cast('${warehouse\_id}' as bigint), '${name}', '${commodity\_type}', cast('${rem\_capacity}' as bigint)`;

db.query(query1, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

const query2 = `update warehouse set rem\_capacity = rem\_capacity - ${rem\_capacity} where id = ${warehouse\_id}`;

db.query(query2, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send

});

})

app.post('/delete-section',(req,res)=>{

const {section\_id, rem\_capacity, warehouse\_id} = req.body

const query01 = `select \* from commodity where section\_id = ${section\_id}`;

db.query(query01, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

if(results != NULL){

res.status(500).send

return

}

const query02 = `delete from section where id = ${section\_id}`;

db.query(query02, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

});

const query2 = `update warehouse set rem\_capacity = rem\_capacity + ${rem\_capacity} where id = ${warehouse\_id}`;

db.query(query2, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

res.status(200).send

})

app.post('/section',(req,res)=>{

const section\_id = req.body.warehouse\_id

db.query(`SELECT \* FROM commodity WHERE section\_id = ${section\_id}` ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.post('/commodity-in', (req, res) =>{

const {company\_id, warehouse\_id, section\_id, name, type, volume, in\_time} = req.body

var commodity\_id = `C` + `${company\_id}` + `W` + `${warehouse\_id}` `S` + `${section\_id}` + `TI` + `${in\_time}`

var query = `insert into commodity(section\_id, volume, name) values (cast('${section\_id}' as bigint), ${volume}, ${name})`

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

query = `update commodity set commodity\_id = concat('${commodity\_id}','IN', cast(max(id) as text)) where id = max(id)`

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

});

query = `update section set rem\_capacity = rem\_capacity - ${volume} where id = ${section\_id}`

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

query = `insert into inout\_records values ( company\_id, warehouse\_id, section\_id, commodity\_id, name, type, volume, in\_time) values

( cast('${company\_id}' as bigint),

(select name from warehouse where id = cast('${warehouse\_id}' as bigint)),

(select name from section where id = cast('${section\_id}' as bigint)),

'${commodity\_id}', '${name}', '${type}',

cast('${volume}' as bigint),

to\_date('${in\_time}', 'yyyy-mm-dd'))`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

res.status(200).send

})

app.post('/commodity-out', (req, res) =>{

const {commodity\_id, section\_id, volume, out\_date} = req.body

var query = `delete from commodity where commodity\_id = '${commodity\_id}'`

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

query = `update section set rem\_capacity = rem\_capacity + cast('${volume}' as bigint) where id = cast('${section\_id}' as bigint)`

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

query = `update inout\_records set out\_time = cast('${out\_date}' as date) where commodity\_id = '${commodity\_id}'`;

db.query(query, (err, results) => {

if(err){

console.log(err)

res.status(400).send

return

}

});

res.status(200).send

})

app.post('/inout\_records',(req,res)=>{

const {

company\_id,

in\_time\_start,

in\_time\_end,

out\_time\_start,

out\_time\_end} = req.body

var query = `SELECT \* FROM inout\_records WHERE company\_id = cast('${company\_id}' as bigint) AND in\_time BETWEEN to\_date('${in\_time\_start}', 'yyyy-mm-dd') and to\_date('${in\_time\_end}', 'yyyy-mm-dd') AND out\_time BETWEEN to\_date('${out\_time\_start}', 'yyyy-mm-dd') and to\_date('${out\_time\_end}', 'yyyy-mm-dd')`;

if(out\_time\_start == null || out\_time\_end == null) {

query = query.replace(` AND out\_time BETWEEN to\_date('${out\_time\_start}', 'yyyy-mm-dd') and to\_date('${out\_time\_end}', 'yyyy-mm-dd')`, ``);

}

if(in\_time\_start == null || in\_time\_end == null) {

query = query.replace(` AND in\_time BETWEEN to\_date('${in\_time\_start}', 'yyyy-mm-dd') and to\_date('${in\_time\_end}', 'yyyy-mm-dd')`, ``);

}

db.query(query ,(err,results)=>{

if(err){

console.log(err)

res.status(400).send

return

}

res.status(200).send(JSON.stringify(results))

})

})

app.listen(1325 /\*angka terakhir harusnya 0\*/, ()=>{

console.log('Port 1325 tersambung')

})

Android (Activity)

A picture containing text, screenshot, design

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a login form

Description automatically generated with medium confidence A screenshot of a login

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence A screenshot of a phone

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence A screenshot of a login screen

Description automatically generated

A white background with black text

Description automatically generated with low confidence A screenshot of a login screen

Description automatically generated with low confidence

A white background with black text

Description automatically generated with low confidence A screenshot of a cell phone

Description automatically generated with medium confidence