Московский государственный технический университет имени Н.Э.Баумана

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по лабораторной 6 по курсу**

**Базовые компоненты**

**«Отчёт по сайту-органайзеру»**

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| Студент группы РТ5-31Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Чинаев А.В. |
|  |  | «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
|  |  |  |
| Проверил |  |  |
| Доцент кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю.Е. |
|  |  | «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

г. Москва – 2022

Оглавление

1 Задание…………………………………….…………………………………… 3

2 Тексты программ………………………………………………..…...………… 3

3 Результаты работы программ...…..…………………………………………... 3

**Задание**

Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.

**Текст программы**

index.js

const express = require("express");

const cors = require("cors");

const bodyParser = require("body-parser");

const router = require('./routes/router');

const authRoutes = require("./routes/auth");

const app = express();

const port = 3000;

app.use(cors());

app.use(bodyParser.json());

const middleware = require('./middleware/index');

app.use (middleware.decodeToken);

app.use(router);

app.listen(port, () => {

console.log(`Server is running on port: ${port}`);

});

index.js

const { getFirestore } = require('firebase/firestore');

const { initializeApp } = require('firebase/app');

// module.exports = {

// secret: "bmstu-secret"

// }

// Import the functions you need from the SDKs you need

// TODO: Add SDKs for Firebase products that you want to use

// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration

// For Firebase JS SDK v7.20.0 and later, measurementId is optional

const firebaseConfig = {

apiKey: "AIzaSyBb\_rs6DBRhxHNw1K2NeYzPUHXOeVACKl0",

authDomain: "bmstuhackathon.firebaseapp.com",

projectId: "bmstuhackathon",

storageBucket: "bmstuhackathon.appspot.com",

messagingSenderId: "498638407243",

appId: "1:498638407243:web:5157c6ceebead260e7afce",

measurementId: "G-X0Q6WSN81F"

};

// Initialize Firebase

const app = initializeApp(firebaseConfig);

const db = getFirestore(app);

module.exports = {

db,

}

auth.js

const express = require("express");

const router = express.Router();

const {

signup,

signin,

} = require("../controllers/auth");

router.post("/signup", signup);

router.post("/signin", signin);

module.exports = router;

router.js

const router = require('express').Router();

const { registrate, login } = require('../controllers/verification');

const { pin, notificate } = require('../controllers/taskFunctions');

const { add, remove, edit, getTask, getGroup, getWeekTasks, getFavourite, getDone } = require('../controllers/task');

router.get('/', getWeekTasks);

router.get('/groups/:name', getGroup);

router.route('/favourite/')

.get(getFavourite)

router.route('/done/')

.get(getDone)

router.route('/task/')

.get(getTask)

.post(add)

.put(edit)

.delete(remove);

module.exports = router;

user.js

class User {

constructor(email, username, password) {

this.UID = UID;

this.email = email;

this.username = username;

this.tasks = [ {} ]

}

}

module.exports = User;

task.js

const { addListener } = require("nodemon");

class Task {

constructor(title, description, date, group) {

this.title = title;

this.description = description

this.date = date;

this.group = group;

this.favourite = favourite;

// this.need\_to\_remind = need\_to\_remind;

// this.duration = duration;

}

}

module.exports = {

Task,

};

entry.js

class Entry {

constructor(timestamp, ip, username, data) {

this.timestamp = timestamp;

this.ip = ip;

this.username = username;

this.data = data;

}

}

module.exports = Entry;

fireStore.js

const { collection, doc, getDoc, getDocs, setDoc, addDoc, deleteDoc, query, where} = require('firebase/firestore');

const { db } = require('../config');

async function addItem(user\_id, task) {

try {

let docRef = await addDoc(collection(db, 'users', user\_id, 'tasks'), task);

console.log("Document written with ID: ", docRef.id);

return docRef.id;

} catch (e) {

console.error("Error adding document: ", e);

return null;

};

}

async function deleteItem(user\_id, task\_id) {

try {

await deleteDoc(doc(db, 'users', user\_id, 'tasks', task\_id));

console.log("Document successfully deleted!");

return 1;

} catch (e) {

console.error("Error removing document: ", e);

return null;

}

}

async function setItem(user\_id, task\_id, task) {

try {

setDoc(doc(db, 'users, UID'), user)

await setDoc(doc(db, 'users', user\_id, 'tasks', task\_id), task);

console.log("Document successfully updated!");

return 1;

} catch (e) {

console.error("Error updating document: ", e);

return null;

}

}

async function getItem(user\_id, task\_id) {

try {

const docRef = doc(db, 'users', user\_id, 'tasks', task\_id);

const docSnap = await getDoc(docRef);

if (docSnap.exists()) {

console.log("Document data:", docSnap.data());

return docSnap.data();

} else {

console.log("No such document!");

return null;

}

} catch (e) {

console.error("Error getting document: ", e);

return null;

}

}

async function getGroupFromBase(user\_id, group\_name) {

try {

const q = query(collection(db, 'users', user\_id, 'tasks'), where('group', '==', group\_name));

const querySnapshot = await getDocs(q);

querySnapshot.forEach((doc) => {

console.log(doc.id, " => ", doc.data());

});

let result = [];

querySnapshot.forEach((doc) => {

result.push(doc.data());

});

return result;

} catch (e) {

console.error("Error getting document: ", e);

return null;

}

}

async function getWeekTasksFromBase(user\_id, date) {

try {

const q = query(collection(db, 'users', user\_id, 'tasks'), where('date', '==', String(date)));

// console.log(date);

const querySnapshot = await getDocs(q);

let result = [];

querySnapshot.forEach((doc) => {

result.push(doc.data());

// console.log(doc.id, " => ", doc.data());

});

// console.log(result);

return result;

} catch (e) {

console.error("Error getting document: ", e);

return null;

}

}

async function getFavouriteFromBase(user\_id) {

try {

const q = query(collection(db, 'users', user\_id, 'tasks'), where('favourite', '==', true));

const querySnapshot = await getDocs(q);

querySnapshot.forEach((doc) => {

console.log(doc.id, " => ", doc.data());

});

let result = [];

querySnapshot.forEach((doc) => {

result.push(doc.data());

});

return result;

} catch (e) {

console.error("Error getting document: ", e);

return null;

}

}

async function getDoneFromBase(user\_id, is\_done) {

try {

if (is\_done == "true") {

is\_done = true;

} else {

is\_done = false;

}

const q = query(collection(db, 'users', user\_id, 'tasks'), where('done', '==', is\_done));

const querySnapshot = await getDocs(q);

querySnapshot.forEach((doc) => {

console.log(doc.id, " => ", doc.data());

});

let result = [];

querySnapshot.forEach((doc) => {

result.push(doc.data());

});

return result;

} catch (e) {

console.error("Error getting document: ", e);

return null;

}

}

module.exports = {

addItem,

deleteItem,

setItem,

getItem,

getGroupFromBase,

getWeekTasksFromBase,

getFavouriteFromBase,

getDoneFromBase,

};

taskFunctions.js

const { Task, } = require('../models/task');

function pin(req, res) {

let task = new Task("a", "b", "c", "d", "e");

res.json(task);

}

function notificate(req, res) {

let task = new Task("a", "b", "c", "d", "e");

res.json(task);

}

module.exports = {

pin,

notificate,

};

task.js

const { Task, } = require('../models/task');

const {addItem,

deleteItem,

setItem,

getItem,

getGroupFromBase,

getWeekTasksFromBase,

getFavouriteFromBase,

getDoneFromBase,

} = require('../data/fireStore');

async function add(req, res) {

user\_id = req.query.user\_id;

let task = req.body;

task\_id = await addItem(user\_id, task);

if (task\_id) {

console.log("Iam out!! Document written with ID: ", task\_id);

res.json({ task\_id: task\_id });

} else {

res.json({ error: 'Error adding task' })

}

}

async function remove(req, res) {

task\_id = req.query.task\_id;

user\_id = req.query.user\_id;

ok = await deleteItem(user\_id, task\_id);

if (!ok) {

res.json({ status: 'Error removing document, please try again' });

}

res.json({ status: 'ok' });

}

async function edit(req, res) {

task\_id = req.query.task\_id;

user\_id = req.query.user\_id;

let task = req.body;

ok = await setItem(user\_id, task\_id, task)

if (!ok) {

res.json({ status: 'Error updating document, please try again' });

}

res.json({ status: 'ok' });

}

async function getTask(req, res) {

task\_id = req.query.task\_id;

user\_id = req.query.user\_id;

let task = await getItem(user\_id, task\_id);

if (!task) {

res.json({ status: 'Error getting document, please try again' });

}

res.json(task);

}

async function getGroup(req, res) {

group\_name = req.params.name;

user\_id = req.query.user\_id;

let group = await getGroupFromBase(user\_id, group\_name);

if (!group) {

res.json({ status: 'Error getting group, please try again' });

}

res.json(group);

}

async function getWeekTasks(req, res) {

user\_id = req.query.user\_id;

let date = new Date(req.query.bgn\_date);

let task = [];

for(let i = 0; i < 7; i++) {

let r = await getWeekTasksFromBase(user\_id, date.getTime());

task.push(r);

date.setDate(date.getDate() + 1);

if (!task[i]) {

res.json({ status: 'Error getting document, please try again' });

}

console.log(task[i]);

}

res.json(task);

}

async function getFavourite(req, res) {

user\_id = req.query.user\_id;

let favour = await getFavouriteFromBase(user\_id);

if (!favour) {

res.json({ status: 'Error getting favour, please try again' });

}

res.json(favour);

}

async function getDone(req, res) {

user\_id = req.query.user\_id;

is\_done = req.query.is\_done;

let done = await getDoneFromBase(user\_id, is\_done);

if (!done) {

res.json({ status: 'Error getting favour, please try again' });

}

res.json(done);

}

module.exports = {

add,

remove,

edit,

getTask,

getGroup,

getWeekTasks,

getFavourite,

getDone,

};