app.js - runtime

```
const express = require('express');
const bodyParser = require('body-parser');
const config = require('./config');
const cors = require('cors');
const app = express();

//BodyParser Settings
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

//CORS settings
app.use(cors());

//#region Main routes
app.use('/', require("./routes/info"));
app.use('/todo', require("./routes/todos"));
//#endregion

//Default error message for unknown request - HTTP 404!
app.use(function(req, res) {
  res.status(404).json({ success: false, result: null, message: `Not
found!`,  timestamp: Date.now()});
});

//API Runtime
app.listen(config["http"]["port"], () => {
  console.log(`[INFO] ${new Date().toLocaleString()} - RESTful API Webserver
started on ${config["http"]["port"]} port!`);
});
```

config.js

```
module.exports = {
    database: {
      host: '',
      user: '',
      password: '',
      database: '',
    },
    http: {
      port: 3000
    }
  };
```

db.js – base

```
const mysql = require('mysql2/promise');
const config = require('./config');

class Database {
    constructor() {
        this.connection = null;
        this.active = false;
        this.handleDisconnect = this.handleDisconnect.bind(this);
        this.handleConnect = this.handleConnect.bind(this);
    }}
//#region Database connection optimalize - 2024.01.27.
const database = new Database();
//#endregion
module.exports = database;
```

db.js – async methods in „Database" class

```
async connect() {
      try {
            //Create connection
            this.connection = await mysql.createConnection(config.database);
            //Handling Log setup
            this.connection.on('end', this.handleDisconnect);
            this.connection.on('error', this.handleDisconnect);
            this.connection.on('connect', this.handleConnect);
            this.active = true;
        } catch (error) {
            console.error(`[ERROR] ${new Date().toLocaleString()} - Error in
MySQL connection: ${error.code}`);
        }}
```

db.js – „filtermethod" and handlers in „Database class"

```
isConnected() {
        if (this.connection == null) {return false;}
        return this.active; }
handleDisconnect() {
        console.error(`[ERROR] ${new Date().toLocaleString()} - The database
connection has been lost!`);
        this.active = false;}
handleConnect() {
        console.log(`[INFO] ${new Date().toLocaleString()} - The database
connection has been restored!`);
        this.active = true;}
```

Route sample – GET method with parameter (optional)

```
const express = require('express');
const router = express.Router();
const database = require('../db');

router.get('/:id', async (req, res) => {
    const id = req.params.id ?? null;
    return res.status(200).json({success: true});
});
module.exports = router;
```

Route – sample – POST method with JSON RAW parameters checking in body

```
router.post('/', async (req, res) => {
    //Request neccessary input datas
    const required_keys = ["title", "details", "completed"];
    try {
        //Check if JSON data is sent in the request body
        if (!req.is('application/json')) {
            //Invalid content type in body

        }
        //POST parameters in body (keys convert to lowercase)
        const values = Object.fromEntries(Object.entries(req.body).map(([key,
value]) => [key.toLowerCase(), value]));
        //Check required post keys
        if (required_keys.filter(key => !(key in values)).length > 0) {
            //Missing required keys in data
        }
      catch (error) {
             //Exception error
      }
});
```

Database request example – GET method

```
router.get('/', async (req, res) => {
    try {
        //Connecting database, if needed
        if (!database.isConnected()) {
            await database.connect();
        }
        //SQL execute
        const [result] = await database.connection.execute("SELECT
`id`,`title`,`details`,`completed`,`created_at`,`modified_at` FROM `todos`;");
        //Results processing
        if (result.length > 0) {
            return res.status(200).json({ success: true, result: result,
message: `Number of tasks: ${result.length}`,  timestamp: Date.now()});
        }
        else {
            return res.status(204).json({success: false, result: null,
message: `The todolist is empty!`});
        }
    } catch (error) {
        return res.status(500).json({ success: false, result: null, message:
`Internal error in request! (${error.code})`,  timestamp: Date.now()});
    }
});
```

HTTP Status Codes		
Status Code	**Description**	**Meaning**
200	OK	The request has succeeded.
201	Created	The request has been fulfilled and a new resource has been created.
204	No Content	The server successfully processed the request but is not returning any content.
400	Bad Request	The server cannot process the request due to client error, such as malformed syntax or invalid request.
401	Unauthorized	The request requires user authentication.
403	Forbidden	The server understood the request, but refuses to authorize it.
404	Not Found	The server cannot find the requested resource.
405	Method Not Allowed	The method specified in the request is not allowed for the resource identified by the request URI.
500	Internal Server Error	A generic error message indicating that the server encountered an unexpected condition preventing it from fulfilling the request.
503	Service Unavailable	The server is currently unable to handle the request due to temporary overload or maintenance.