A 100DEVS PRESENTATION

MVC

A SOFTWARE ARCHITECTURE PARADIGM

By Borja Martí

MVC

MODEL - VIEW - CONTROLLER

- Architectural paradigm
- Separates user interface from business logic

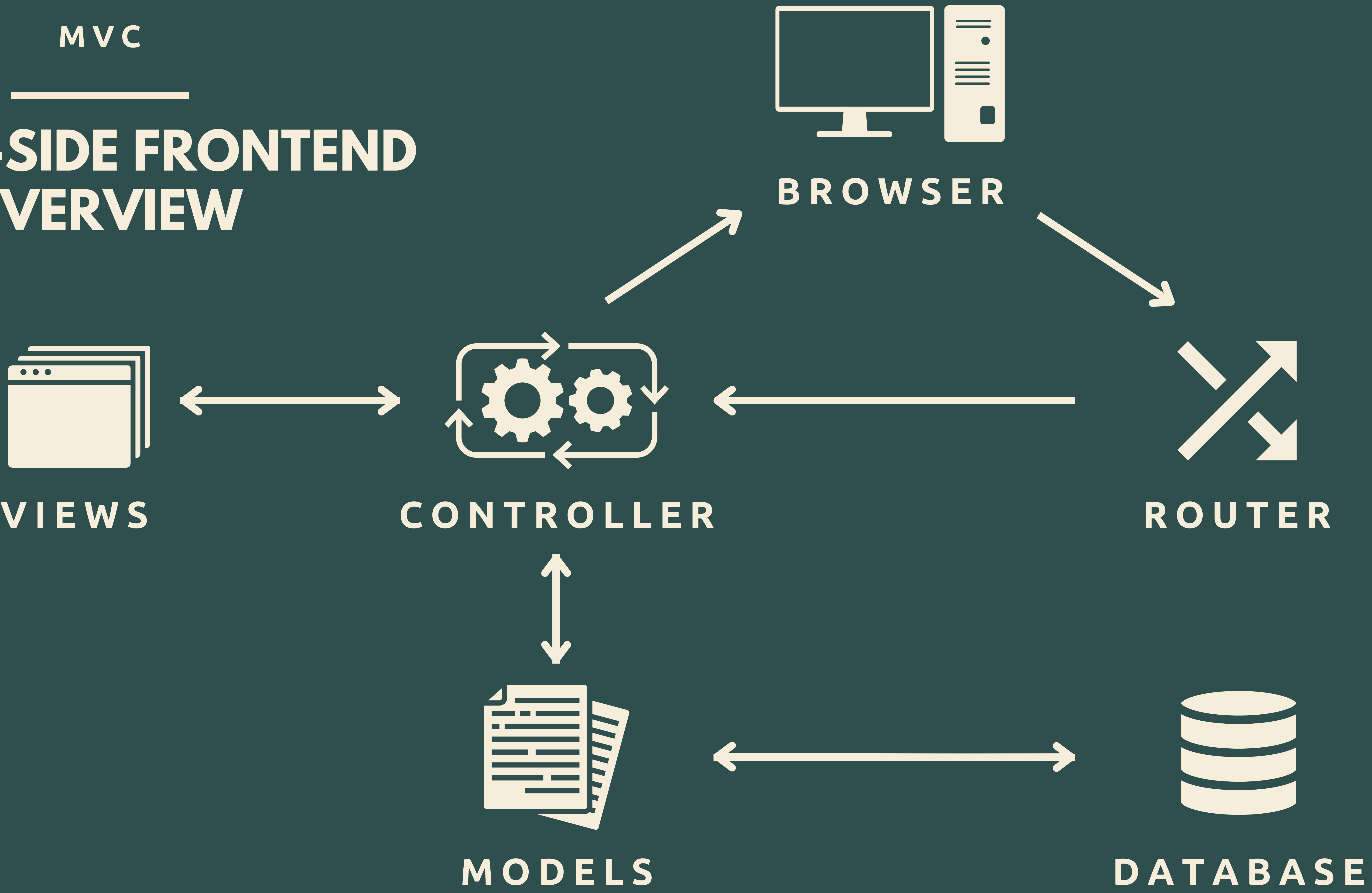
MVC

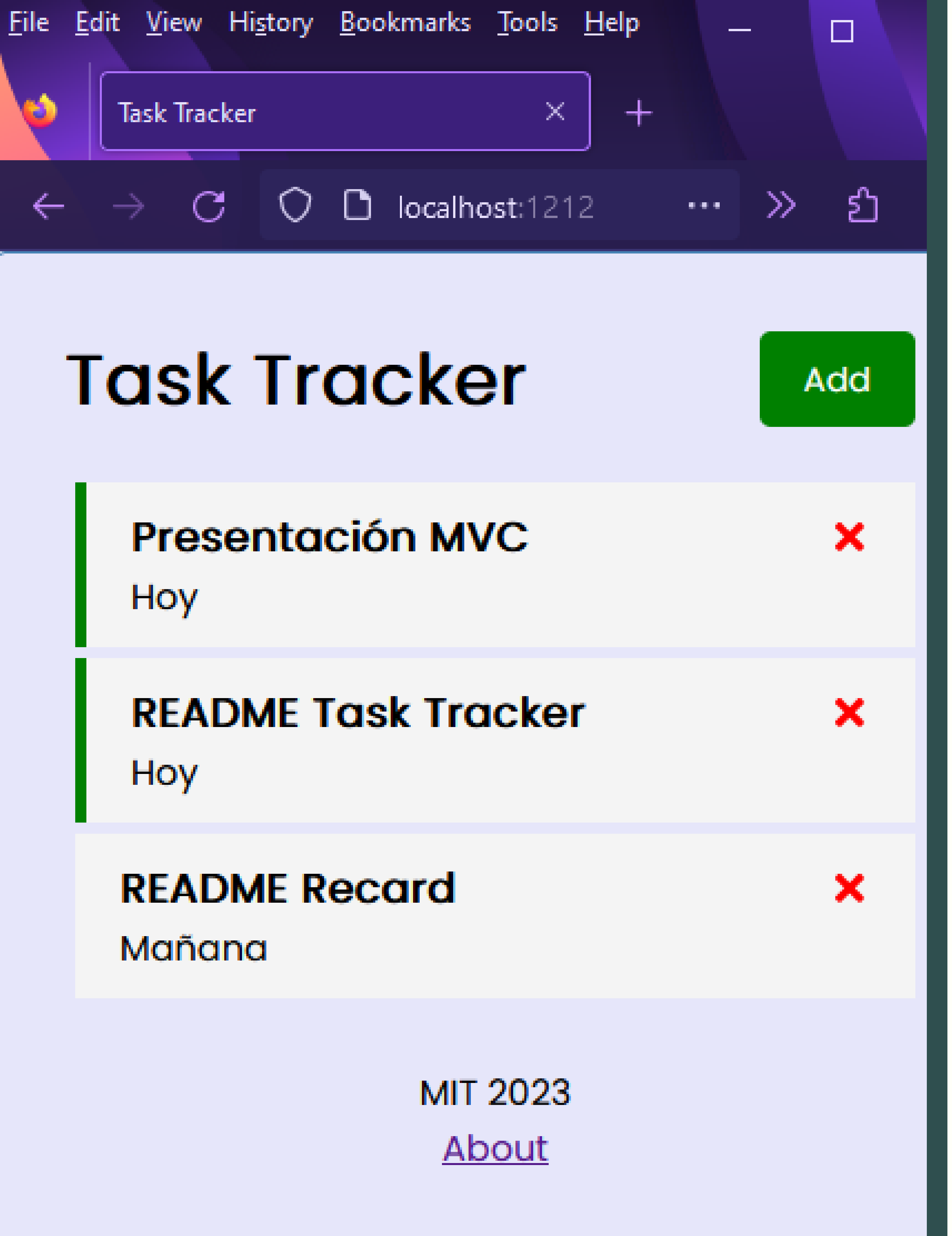
WHY?

- Separation of concerns:
  - Modularity and reusability
  - Readability and maintainability

MVC

SERVER-SIDE FRONTEND OVERVIEW





Task Tracker

Add

Presentación MVC
Hoy

README Task Tracker
Hoy

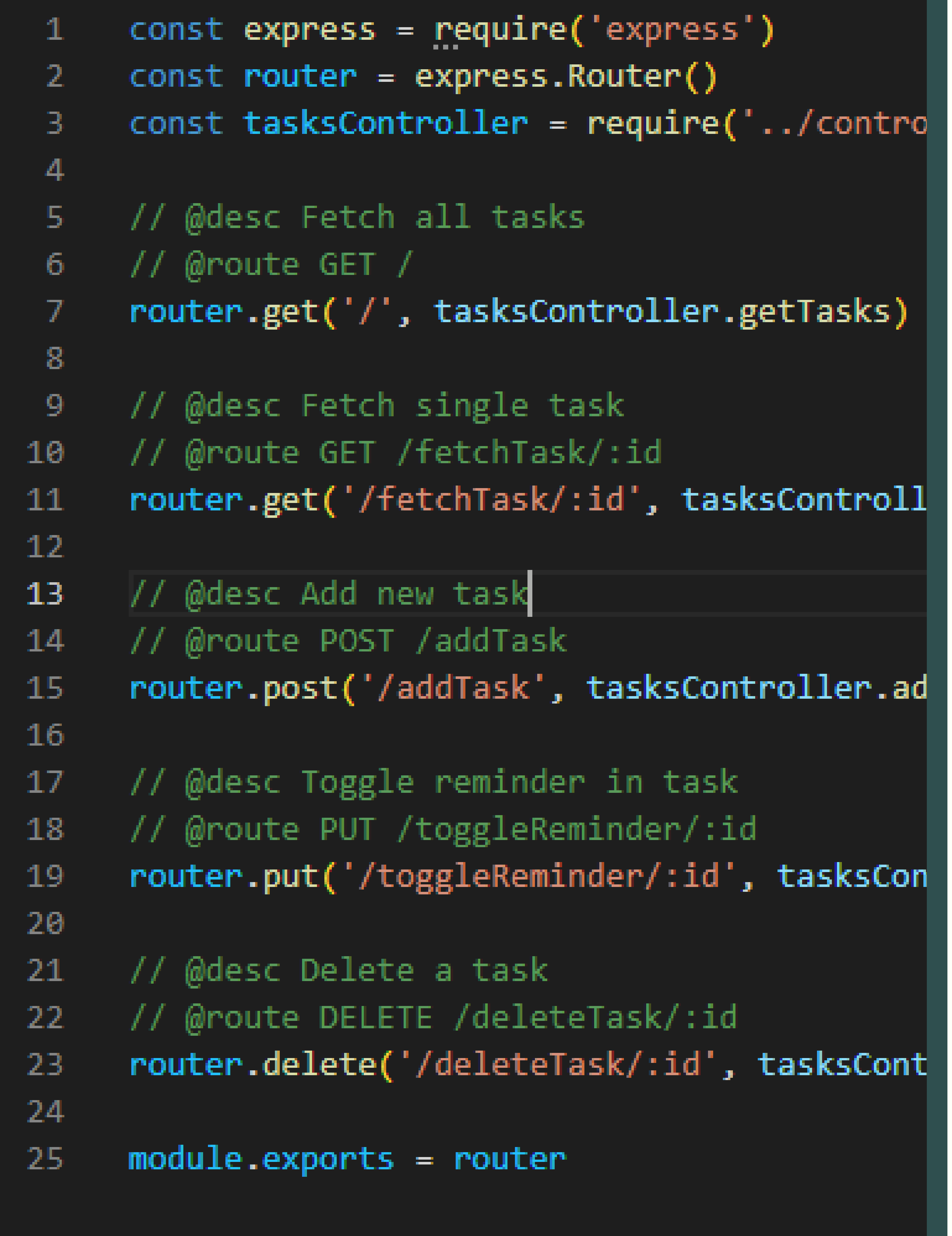
README Recard
Mañana

MIT 2023
About

MVC

BROWSER

- Send requests
- Receive responses
- Display content

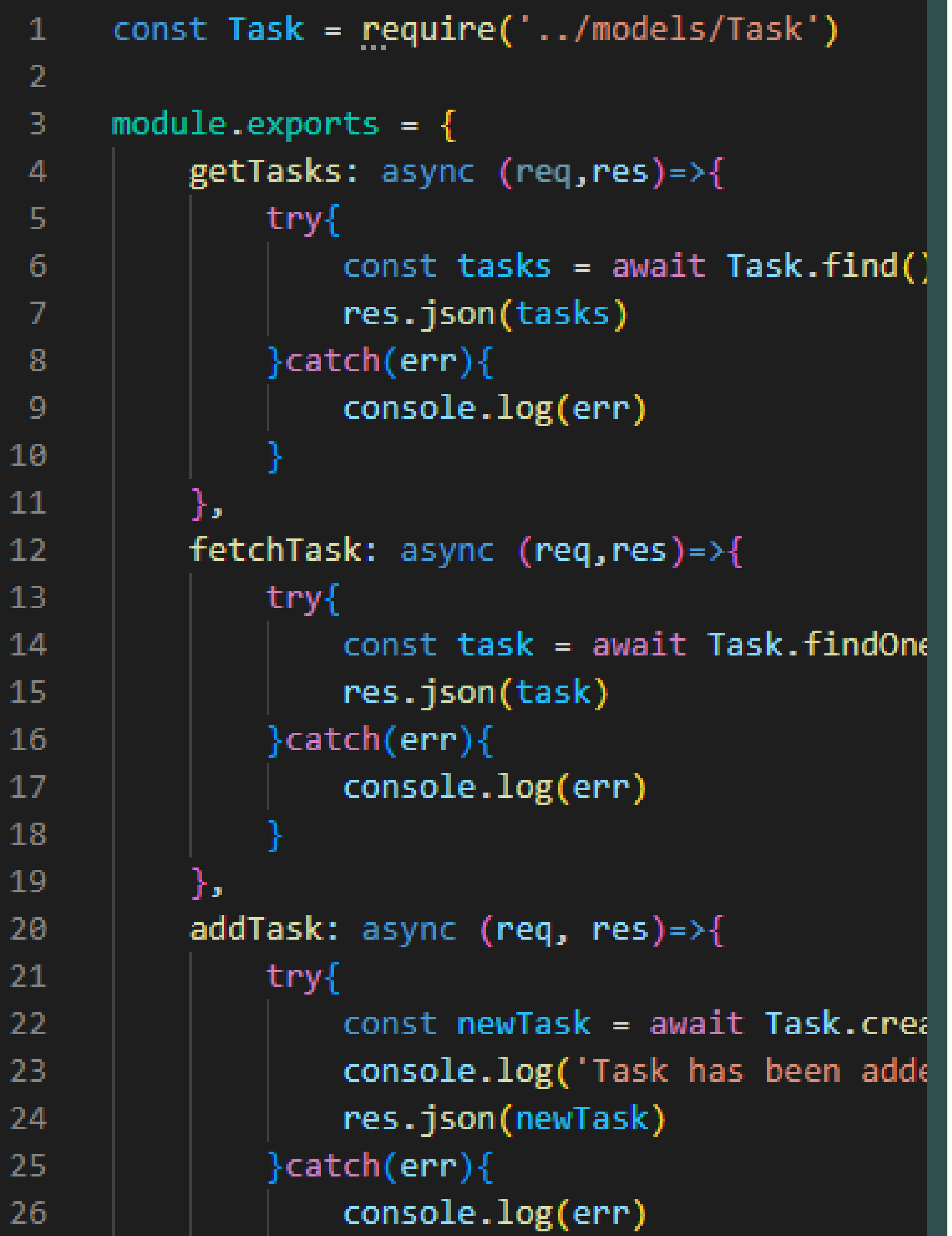


```
const express = require('express')
const router = express.Router()
const tasksController = require('../contro

// @desc Fetch all tasks
// @route GET /
router.get('/', tasksController.getTasks)

// @desc Fetch single task
// @route GET /fetchTask/:id
router.get('/fetchTask/:id', tasksControll

// @desc Add new task
// @route POST /addTask
router.post('/addTask', tasksController.ad

// @desc Toggle reminder in task
// @route PUT /toggleReminder/:id
router.put('/toggleReminder/:id', tasksCon

// @desc Delete a task
// @route DELETE /deleteTask/:id
router.delete('/deleteTask/:id', tasksCont

module.exports = router
```

MVC

ROUTER

- Parses requests
- Directs them to handlers

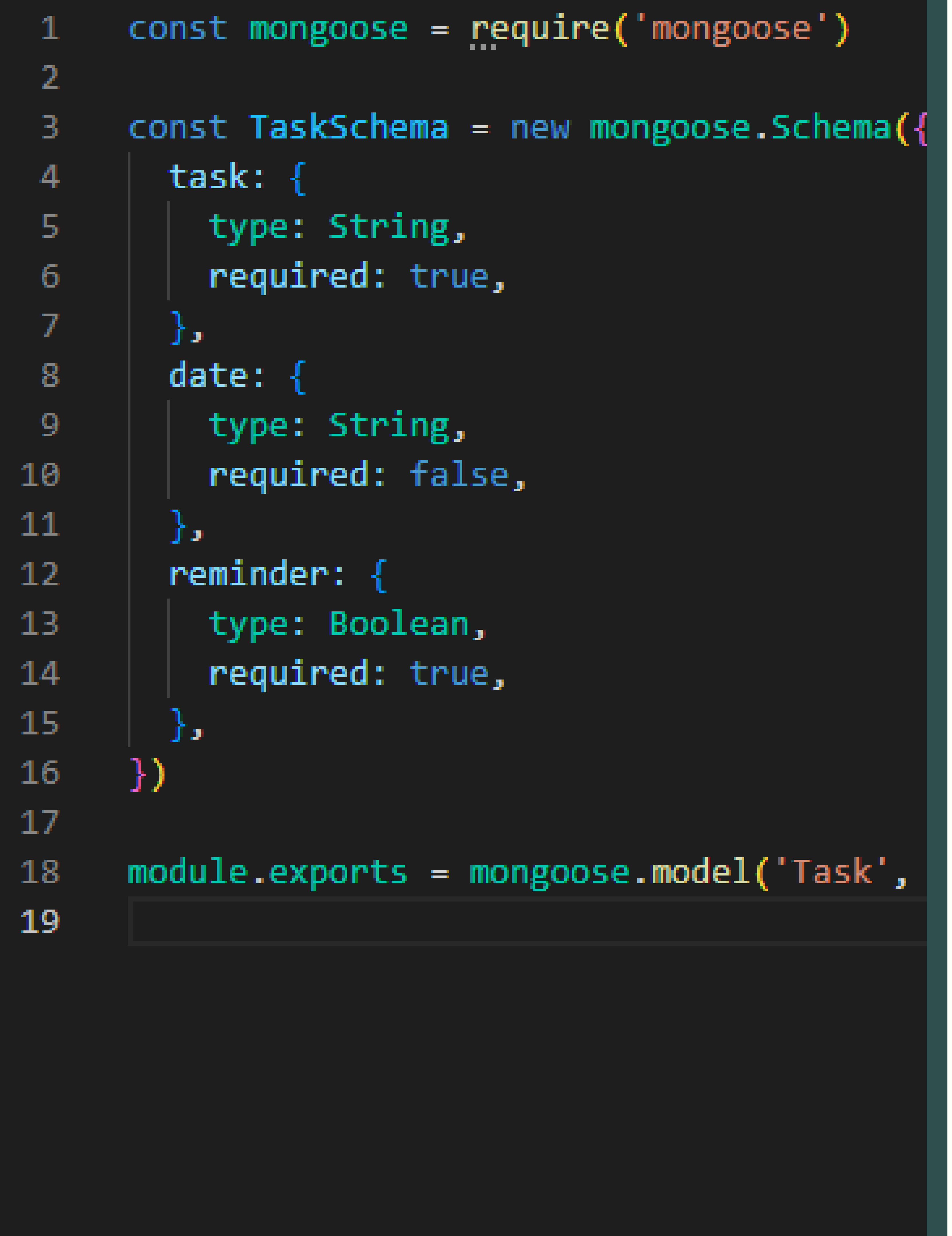


```
const Task = require('../models/Task')

module.exports = {
    getTasks: async (req,res)=>{
        try{
            const tasks = await Task.find()
            res.json(tasks)
        }catch(err){
            console.log(err)
        }
    },
    fetchTask: async (req,res)=>{
        try{
            const task = await Task.findOne
            res.json(task)
        }catch(err){
            console.log(err)
        }
    },
    addTask: async (req, res)=>{
        try{
            const newTask = await Task.crea
            console.log('Task has been adde
            res.json(newTask)
        }catch(err){
            console.log(err)
```

MVC

CONTROLLER

- Mediates Model and View
- Directs flow of data

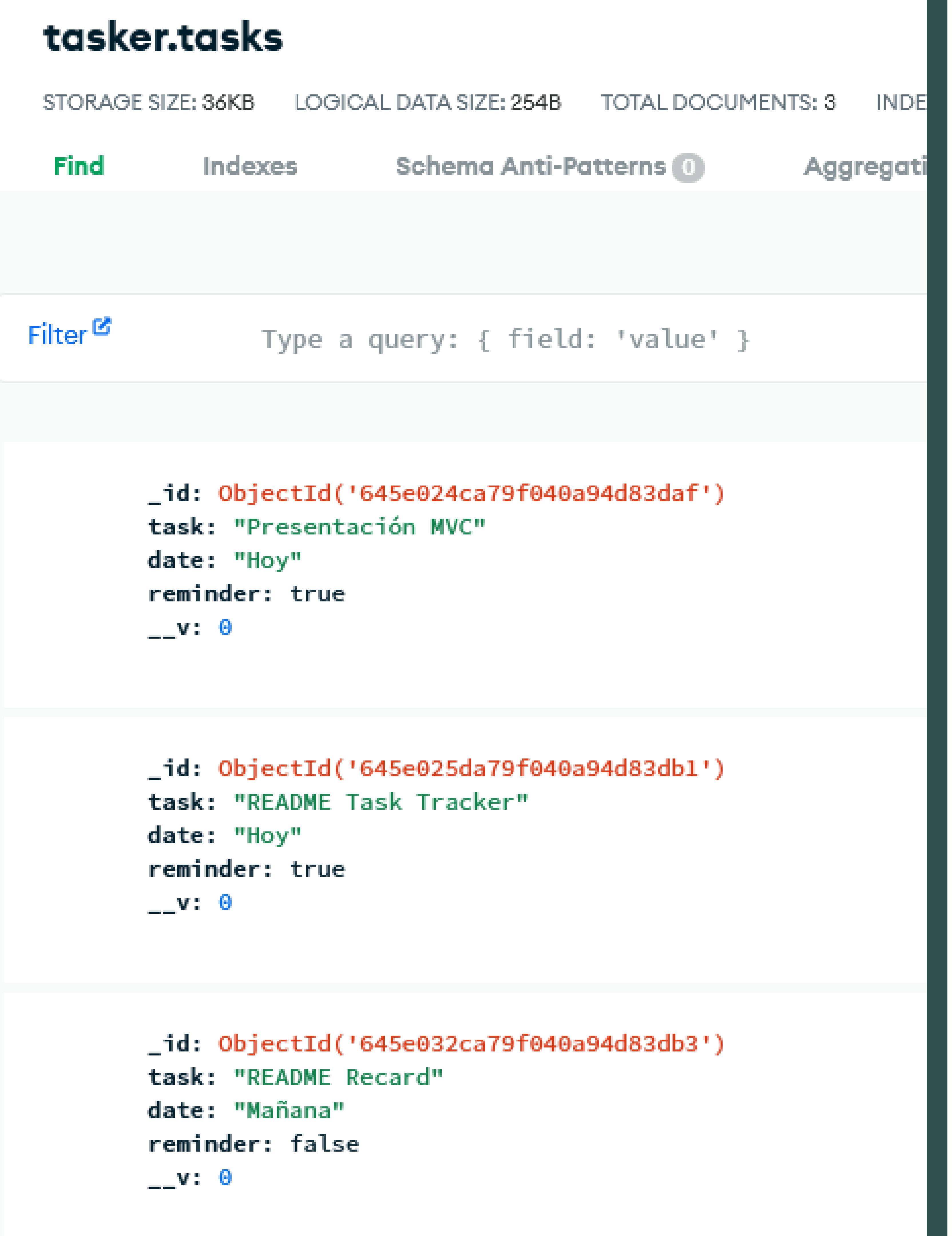


```
const mongoose = require('mongoose')

const TaskSchema = new mongoose.Schema({
  task: {
    type: String,
    required: true,
  },
  date: {
    type: String,
    required: false,
  },
  reminder: {
    type: Boolean,
    required: true,
  },
})

module.exports = mongoose.model('Task',
```

MVC

MODELS

- Handles data access
- Data management
- Data validation

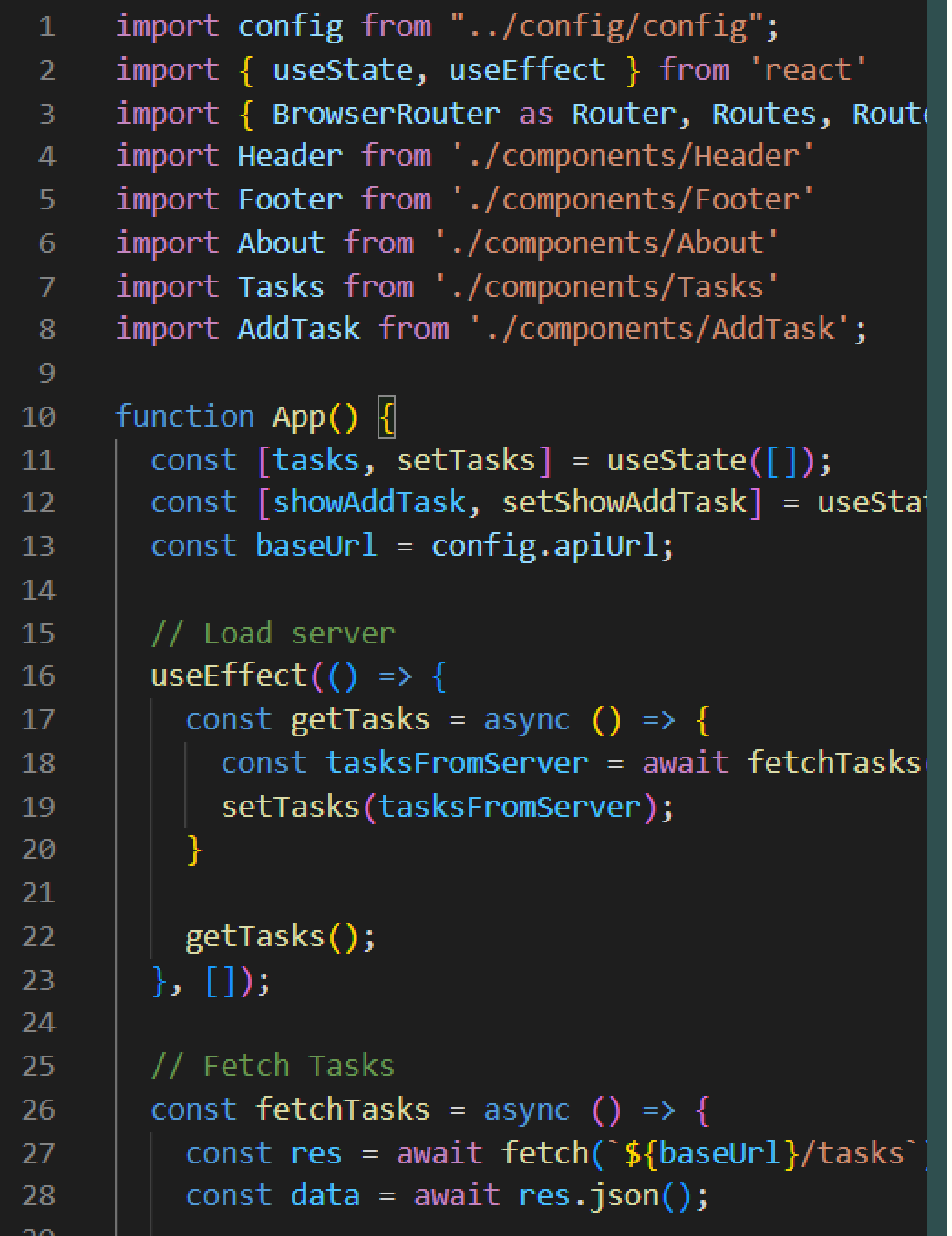


```
_id: ObjectId('645e024ca79f040a94d83daf')
task: "Presentación MVC"
date: "Hoy"
reminder: true
__v: 0

_id: ObjectId('645e025da79f040a94d83db1')
task: "README Task Tracker"
date: "Hoy"
reminder: true
__v: 0

_id: ObjectId('645e032ca79f040a94d83db3')
task: "README Recard"
date: "Mañana"
reminder: false
__v: 0
```

MVC

DATABASE

- Data storage
- Data security



```
import config from "../config/config";
import { useState, useEffect } from 'react'
import { BrowserRouter as Router, Routes, Rout
import Header from './components/Header'
import Footer from './components/Footer'
import About from './components/About'
import Tasks from './components/Tasks'
import AddTask from './components/AddTask';

function App() {
  const [tasks, setTasks] = useState([]);
  const [showAddTask, setShowAddTask] = useSta
  const baseUrl = config.apiUrl;

  // Load server
  useEffect(() => {
    const getTasks = async () => {
      const tasksFromServer = await fetchTasks
      setTasks(tasksFromServer);
    }

    getTasks();
  }, []);

  // Fetch Tasks
  const fetchTasks = async () => {
    const res = await fetch(`${baseUrl}/tasks`
    const data = await res.json();
```

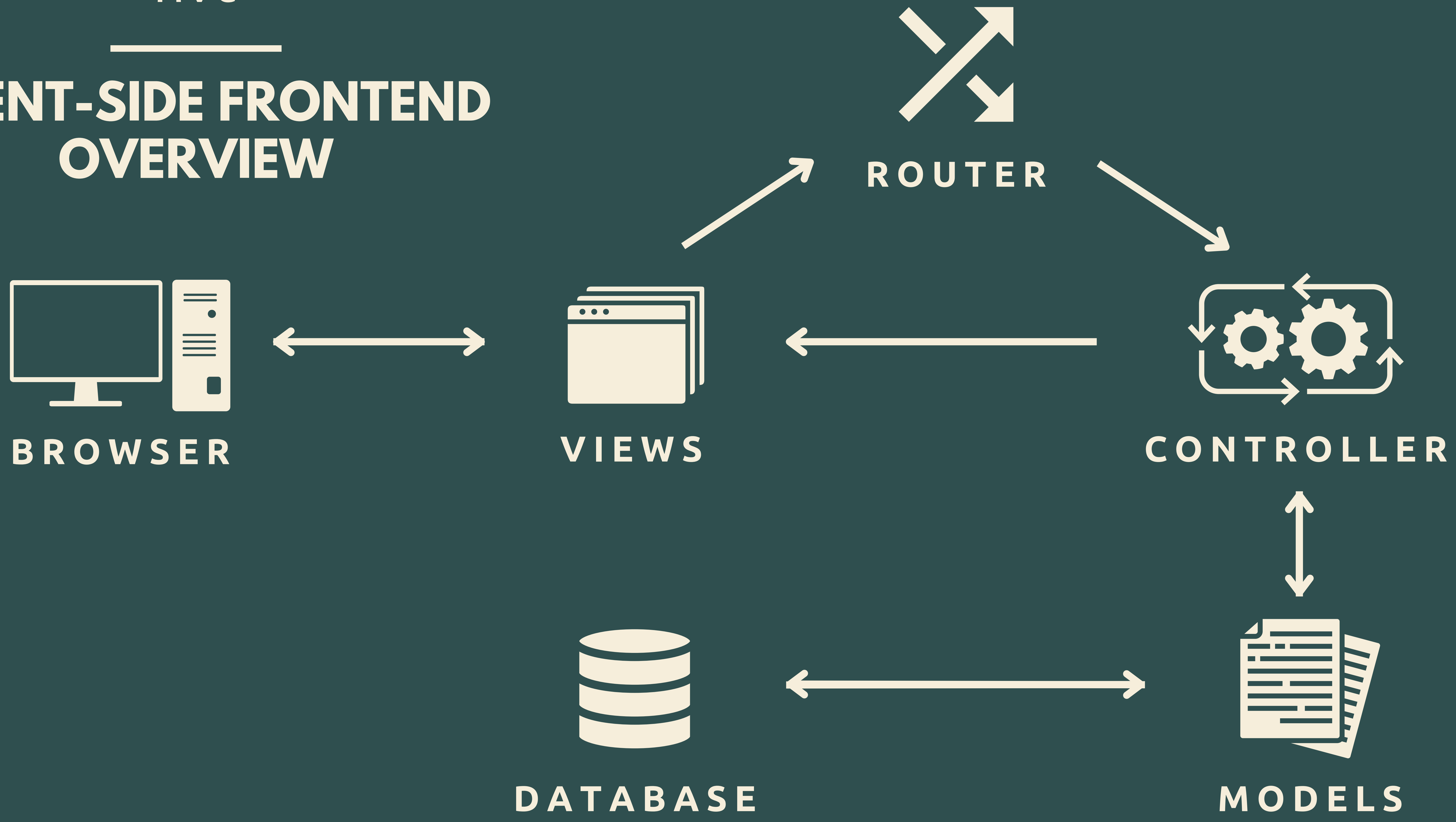
MVC

VIEWS

- Renders the User Interface
- Presents data

MVC

CLIENT-SIDE FRONTEND OVERVIEW



A 100DEVS PRESENTATION

HAPPY CODING!

MVC

github.com/borjaMarti