To do list for to do list lol

* Set up backend environment
* Create backend directory
* Go to backend
* Start up npm package, npm init -y, for me bun init
* Modify package JSON
* Install express
* Add express.json() in index.ts
* Setup app.config to host all env

*import* express *from* "express";

*import* { host, port } *from* "./app.config";

*const* app = *express*();

app.*use*(express.*json*());

app.*listen*(port, host, () *=>* {

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

});

Set up simple app.get route to send message to client side

app.*use*(express.*urlencoded*({ extended: **true** })); to use url encoded

setup router:

create router folder

create a router.ts

*const* router = express.*Router*();

router.*use*("/todo", todoRoute);

*export* *default* router;

create todoRoute

*import* express *from* "express";

*const* router = *express*();

router.*route*("/").*get*(getAllTodos);

router.*route*("/:todo").*patch*(updateTodo).*delete*(deleteTodo);

*export* *default* totoRouter

set up todo router

*import* *type* { Request, Response } *from* "express";

*export* *const* *getAllTodos* = *async* (req: Request, res: Response) *=>* {

  res.*send*("this gets all todos");

};

*export* *const* *updateTodo* = *async* (req: Request, res: Response) *=>* {

  res.*send*("this updates one todo");

};

*export* *const* *deleteTodo* = *async* (req: Request, res: Response) *=>* {

  res.*send*("this deletes one todo");

};

Install mongoose

Create folder db

Create file connection.ts

*import* mongoose *from* "mongoose";

*const* *connectDB* = (uri: string) *=>* {

*return* mongoose.*connect*(uri);

};

*export* *default* connectDB;

modify server

*const* *startUpServer* = *async* () *=>* {

*await* *connectDB*(mongo\_uri);

  console.*log*("database connection successful");

  app.*listen*(port, host, () *=>* {

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

  });

};

Setup todo schema models -> todo.model.ts

*import* mongoose *from* "mongoose";

*const* TodoSchema = new mongoose.*Schema*({

  content: {

    type: String,

    required: **true**,

  },

  conmplete: {

    type: Boolean,

    default: **false**,

  }

});

*const* Todo = mongoose.*model*("Todo", TodoSchema);

*export* *default* Todo;

*export* *const* *getAllTodos* = *async* (req: Request, res: Response) *=>* {

*const* todos = *await* Todo.*find*();

  res.*status*(StatusCodes.OK).*json*(todos);

};

Add http-status-codes

Create validation schema dtos -> todo -> create\_todo.dto.ts

*import* Joi *from* "joi";

*export* *const* createTodoSchema = Joi.*object*({

  contect: Joi.*string*().*required*(),

  completed: Joi.*boolean*().*default*(**false**),

});

Send updated error handler and middlewares

Next is validating the schema sent for createTodo

*export* *const* *createTodo* = *async* (

  req: Request,

  res: Response,

  next: NextFunction

) *=>* {

*try* {

*const* { error, value } = createTodoJoiSchema.*validate*(req.body);

*if* (error) *throw* new *BadRequest*(error.message);

    console.*log*(value);

  } *catch* (error) {

*next*(error);

  }

};

Create a new todo and store in the database

*export* *const* *createTodo* = *async* (

  req: Request,

  res: Response,

  next: NextFunction

) *=>* {

*try* {

*const* { error, value } = createTodoJoiSchema.*validate*(req.body);

*if* (error) *throw* new *BadRequest*(error.message);

*const* newTodo = new *Todo*(value);

*await* newTodo.*save*();

    res.*status*(StatusCodes.OK).*json*({ message: "Todo added successfully" });

  } *catch* (error) {

*next*(error);

  }

};

Next add update dto to extend create dto

*import* { createTodoJoiSchema } *from* "./create\_todo.dto";

*export* *const* updateTodoJoiSchema = createTodoJoiSchema.*fork*(

  Object.*keys*(createTodoJoiSchema.*describe*().keys),

  (schema) *=>* schema.*optional*()

);

Then implement the update function

*export* *const* *updateTodo* = *async* (

  req: Request,

  res: Response,

  next: NextFunction

) *=>* {

*try* {

*const* { error, value } = updateTodoJoiSchema.*validate*(req.body);

*if* (error) *throw* new *BadRequest*(error.message);

*const* updatedTodo = *await* Todo.*findByIdAndUpdate*(req.params.\_id, value, {

      new: **true**,

      runValidators: **true**,

    });

*if* (!updatedTodo) *throw* new *RequestNotFound*("Todo not found");

    res

      .*status*(StatusCodes.OK)

      .*json*({ message: `Todo with id ${updatedTodo.\_id} successfully` });

  } *catch* (error) {

*next*(error);

  }

};

For the delete, copy the update and make some changes

Frontend time

Npm create vite or bun im using bun

Then add the proxy to access backend

server: {

    proxy: {

      "/api": {

        target: "http://localhost:3000", *// Backend server URL*

        changeOrigin: **true**, *// Needed for virtual hosted sites*

*rewrite*: (path) *=>* path.*replace*(/*^*\/api/, ""), *// Remove /api prefix*

      },

    },

  },