Community Tracker API Setup

Backend Stack:

Languages: Go

Framework Documentation: Go Fiber

Repository Link: Github Repo

ORM Documentation: GORM

Database: PostgreSQL

Build Tool: Docker

Docker Setup:

Download docker desktop and install it on your local machine. The project uses docker for development so no other software needs to be installed on your machine and no admin rights are needed when running services.

On docker installation, make sure to enable Hyper-V in windows, found in windows features. Unclick wsl2 when installing docker, wls2 cannot be installed so Hyper-V will be used by docker for virtualization. Verify in the settings that wsl2 in unchecked before proceeding to clone the project.

Environment Setup:

Clone the project in GitHub, once cloned create a copy of .env.example and name it to .env, this is where we will put database credentials and other secret keys. Ask for a sample copy of the .env file, to help set up the project.

Sample .env file

Do not run the project in docker without creating an env file. The values in env will be used by docker to set up the containers and use its values as credentials. For the DB_HOST, use the name of the container in the docker-compose.yml file. For the **JWT_SECRET**, you can generate your own from grc.com and use the 64 random hexadecimal characters. The jwt secret will be used to sign tokens for authentication.

Running the API Service:

Once the env file has been set, you can now run the project through docker. Open a terminal in the project root directory and run the command.

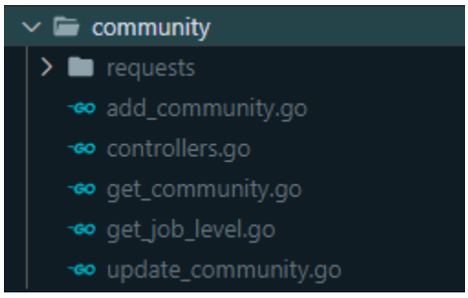
```
docker compose up --build
```

This command will download and build the necessary containers for the project to run, so no other software is needed for you to install.

Please read the README.md file in the repository for other commands when running the container and building the project and for code changes.

Coding:

For the development, all files should be inside the pkg directory. For creating endpoints, create a folder named after the resource name. Each resource should have a controller.go file to handle the routes. Register the route the main.go file located in the cmd directory.



Package structure

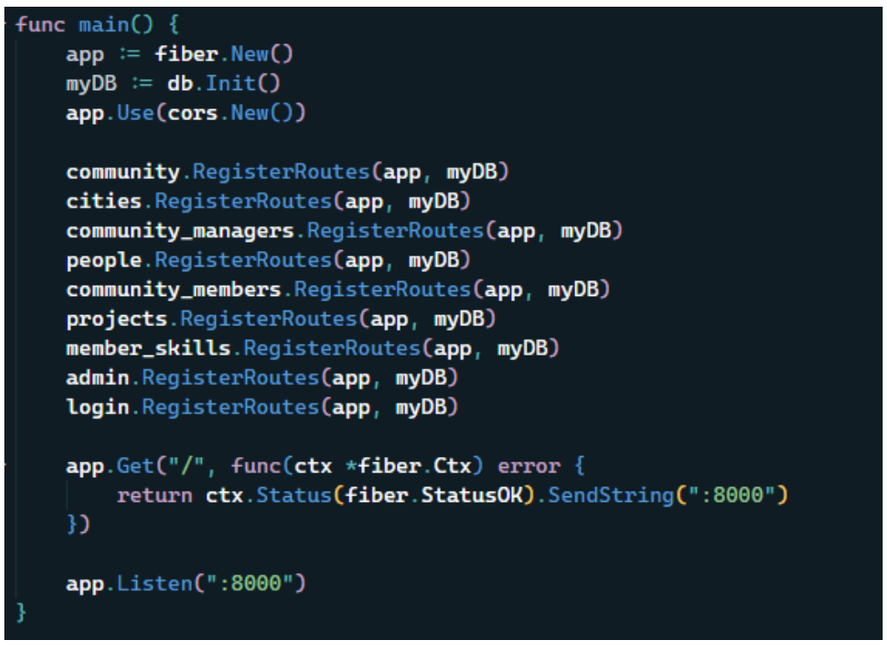
```go
package community

import (
    "github.com/VncntDzn/community-tracker-api/pkg/middleware"      You, yesterday • Imple
    "github.com/gofiber/fiber/v2"
    "gorm.io/gorm"
)

Vincent, last month | 1 author (Vincent)
type handler struct {
    DB *gorm.DB
}

func RegisterRoutes(app *fiber.App, db *gorm.DB) {
    h := &handler{
        DB: db,
    }
    communityRoutes := app.Group("/api/community")
    communityRoutes.Get("/job-level", h.GetJobLevel)
    communityRoutes.Get("/", h.GetCommunity)
    communityRoutes.Post("/", middleware.AuthMiddleware, h.AddCommunity)
    communityRoutes.Put("/:communityid", middleware.AuthMiddleware, h.UpdateCommunity)
}
```

controller.go



```go
func main() {
    app := fiber.New()
    myDB := db.Init()
    app.Use(cors.New())

    community.RegisterRoutes(app, myDB)
    cities.RegisterRoutes(app, myDB)
    community_managers.RegisterRoutes(app, myDB)
    people.RegisterRoutes(app, myDB)
    community_members.RegisterRoutes(app, myDB)
    projects.RegisterRoutes(app, myDB)
    member_skills.RegisterRoutes(app, myDB)
    admin.RegisterRoutes(app, myDB)
    login.RegisterRoutes(app, myDB)

    app.Get("/", func(ctx *fiber.Ctx) error {
        return ctx.Status(fiber.StatusOK).SendString(":8000")
    })

    app.Listen(":8000")
}
```

main.go

Register the resource package in the main.go file.

Golang Tutorial: [The Net Ninja - Golang Tutorial]