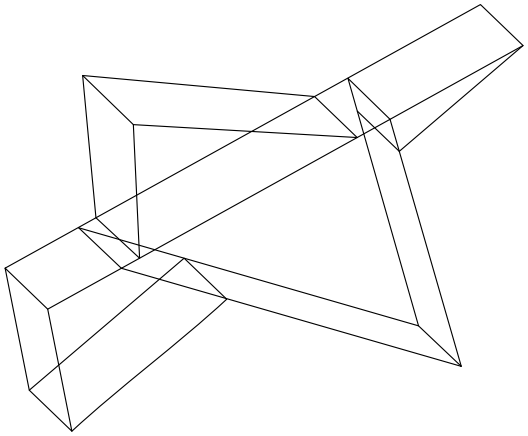

Assignment:

Habit Tracker

Stockholm 2024-10-23

netlight

1 Assignment

Your task is to develop a Habit Tracking API that allows a user to track their behaviors and improve their daily habits. By systematically tracking ones habits, users can identify patterns, set goals, and hold themselves accountable for their progress. This could lead to positive behavioral changes and a more structured and productive daily life! 💜

We recommend using GitHub to upload your project. A short guide can be found here: https://docs.github.com/en/get-started/start-your-journey/hello-world

1.1 End-Points

- GET `/habits`: Retrieve a list of all habits..
- GET `/habits/{id}`: Retrieve details of a specific habit based on its ID.
- POST `/habits`: Add a new habit.
- PATCH `/habits/{id}`: Update information for a specific habit.
- DELETE `/habits/{id}`: Delete a specific habit.
- POST `/habits/{id}/tracking`: Add a new tracking for a specific habit.
- GET `/habits/{id}/tracking`: Retrieve all trackings for a specific habit.

1.2 Data Model

A ”Habit” has the following properties:

- `id`: Unique Identifier (UUID)
- `name`: The name of the habit (string)
- `description`: Description of the habit (string)
- `frequency`: Frequency of the habit (e.g. daily/weekly) (enum/string)
- `startDate`: Date the habit is started (ISO-formated date[1])

One tracking entry has the following properties:

- `id`: Unique Identifier (UUID)
- `habitId`: Id of the habit the entry belongs to (UUID)Hab
- `timestamp`: Timestamp of the entry (ISO-formated timestamp)
- `note`: Optional note about the entry (string)

1.3 Extension

If the basic functionality is implemented and you desire an additional challenge, consider adding the following functionality:

- **Web UI:** Create a graphical interface in React ontop of the API.
- **Reminders** : Implement a feature to send reminders to users to perform their habits. This can be done via email or push notifications (websockets?).
- **Statistics and Data Visualization**: Add endpoints to generate statistics and visualizations, such as graphs showing progress over time.
- **Auth:** Implement authentication and authorization to ensure that only authorized users can add, update, or delete habits and their trackings.

[1] https://en.wikipedia.org/wiki/ISO_8601

1.4 Setup

1.4.1 Java/Spring Boot

Java is a classic choice found in almost every company. In the Java world Spring Boot is often the go-to framework for creating large-scale web applications in Java. To get started with Spring Boot, you can use the following steps:

1. Install an Integrated Development Environment (IDE):

   The most popular IDE for Java is IntelliJ IDEA.

2. Create a new Spring Boot-project:

   Go tol Spring Initializr and generate your project. Download and open it with your chosen IDE.

3. Start the application

   To start the server just run `./mvnw spring-boot:run` in the terminal.

1.4.2 C#/.NET

.NET is a versatile platform for building all types of applications. Here are the steps to get started with a web API in C#:

1. Install the .Net SDK:

   Download and install the .Net Core SDK from Microsofts official website.

2. Install an Integrated Development Environment (IDE):

   Visual Studio is the most used IDE for C#/.NET-development. An alternative could be Visual Studio Code with the C#-plugin for a more lightweight IDE.

3. Create a new .Net-Core project:

   - Open Visual Code and go to "Create a new project".
   - Pick the project type "ASP.NET Core Web Application".

4. Run your application:

   To run your application just hit F5 or click "Run".

1.4.3 Golang

Golang, or Go, is a statically typed programming language designed for simplicity and efficiency. To get started:

1. Install Go:

   Download the go cli tool from The official Go-website.

2. Install a integrated Development Environment (IDE):

   Popular IDEs for go development are usally Visual Studio Code with the Go-plugin, or GoLand from JetBrains.

3. **Create a Go Project**:

   Create a new folder for your project and a main.go file.

   A simple Starting point can be something like this:

```go
package main

import (
    "encoding/json"
    "net/http"
    "log"
)

func getHabits(w http.ResponseWriter, r *http.Request) {
  // Implement the  endpoint here
}

func main() {
    http.HandleFunc("/habits", getHabits)
    log.Println("Server is running on port 8080")
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

4. **Run the application**:
   Open your terminal and run `go run main.go`.