

AI Meets Node JS With Crafting a Custom Sports Activity Service

Anton Kalik

Lineup

Introduction

OpenAI

Setup Application

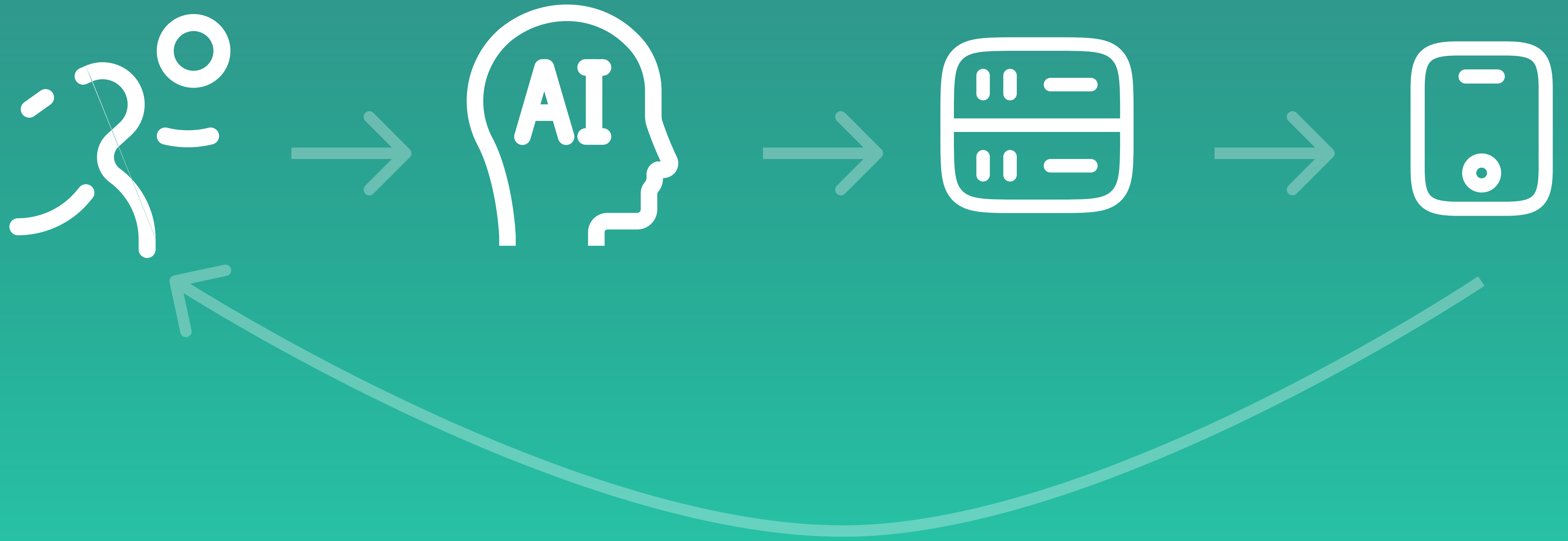
Connection

Prompting

Wrapping

Conclusion

Introduction





OpenAI







Setup OpenAI



ChatGPT



Interact with our flagship language models in a conversational interface

API



Integrate OpenAI models into your application or business

OPENAI_API_KEY="dm-PLCUj7YiTV9e80dAT26AI7BhcdkFJYZ3hz0p00H3927315y84"



Setup Node JS Service

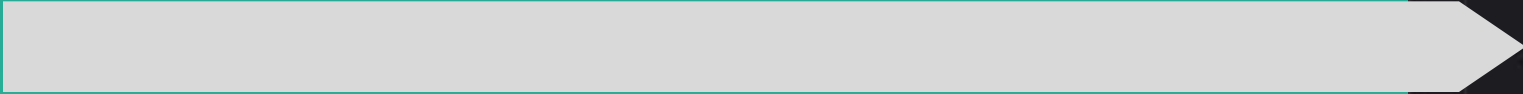
```
npx create-node-application pullap_ai_servise -fw=express
```


Dependencies

```
"dependencies": {  
  "bcrypt": "^5.1.1",  
  "body-parser": "^1.20.2",  
  "cors": "^2.8.5",  
  "dotenv": "^16.3.1",  
  "express": "^4.18.2",  
  "jsonwebtoken": "^9.0.2",  
  "knex": "^3.0.1",  
  "openai": "^4.14.2",  
  "pg": "^8.11.3",  
  "redis": "^4.6.10",  
  "validator": "^13.11.0"  
},
```

package.json

Dependencies



```
"dependencies": {  
  "bcrypt": "^5.1.1",  
  "body-parser": "^1.20.2",  
  "cors": "^2.8.5",  
  "dotenv": "^16.3.1",  
  "express": "^4.18.2",  
  "jsonwebtoken": "^9.0.2",  
  "knex": "^3.0.1",  
  "openai": "^4.14.2",  
  "pg": "^8.11.3",  
  "redis": "^4.6.10",  
  "validator": "^13.11.0"  
},
```

package.json

```
1 PORT=9999
2
3 # OpenAI
4 OPENAI_API_KEY="openai_api_key"
5
6 # DB
7 POSTGRES_HOST="localhost"
8 POSTGRES_PORT=5432
9 POSTGRES_DB="pullap_ai_development"
10 POSTGRES_USER="postgres_user"
11 POSTGRES_PASSWORD="postgres_password"
12
13 # User
14 DEFAULT_PASSWORD="default_password"
15 JWT_SECRET="jwt_secret"
16
17 # Test User
18 TEST_USER_EMAIL="your@email.com"
19 TEST_USER_FIRST_NAME="first_name"
20 TEST_USER_LAST_NAME="last_name"
21 TEST_USER_USERNAME="firstname_lastname"
22
23 # Redis
24 REDIS_HOST="localhost"
25 REDIS_PORT=6379
26 REDIS_DB=0
27 REDIS_PASSWORD="redis_password" |
```

.env

Knex Setup

knexfile.ts

```
1  require('dotenv').config();
2  require('ts-node/register');
3  import type { Knex } from 'knex';
4
5  const connection: Knex.ConnectionConfig = {
6    host: process.env.POSTGRES_HOST as string,
7    database: process.env.POSTGRES_DB as string,
8    user: process.env.POSTGRES_USER as string,
9    password: process.env.POSTGRES_PASSWORD as string,
10 };
11
12 const commonConfig: Knex.Config = {
13   client: 'pg',
14   connection,
15   migrations: {
16     directory: './database/migrations',
17   },
18   seeds: {
19     directory: './database/seeds',
20   },
21 };
22
23 export default {
24   development: {
25     ...commonConfig,
26   },
27   test: {
28     ...commonConfig,
29     connection: {
30       ...connection,
31       database: process.env.POSTGRES_DB_TEST as string,
32     },
33   },
34 };
35
```


Knex Migrations - Users

```
1 import { Knex } from 'knex';
2
3 const tableName = "users" as const;
4
5 export async function up(knex: Knex): Promise<void> {
6   await knex.schema.createTable(tableName, { callback: function (table: Knex.CreateTableBuilder): void {
7     table.increments( columnName: 'id').primary();
8     table.string( columnName: 'first_name').nullable();
9     table.string( columnName: 'last_name').nullable();
10    table.string( columnName: 'username').nullable();
11    table.string( columnName: 'email').nullable();
12    table.string( columnName: 'password').nullable();
13    table.enum( columnName: 'role', values: ['user', 'admin']).nullable();
14    table.timestamps( useTimestamps: true, defaultToNow: true);
15   });
16 }
17
18 export async function down(knex: Knex): Promise<void> {
19   await knex.schema.dropTable(tableName);
20 }
```

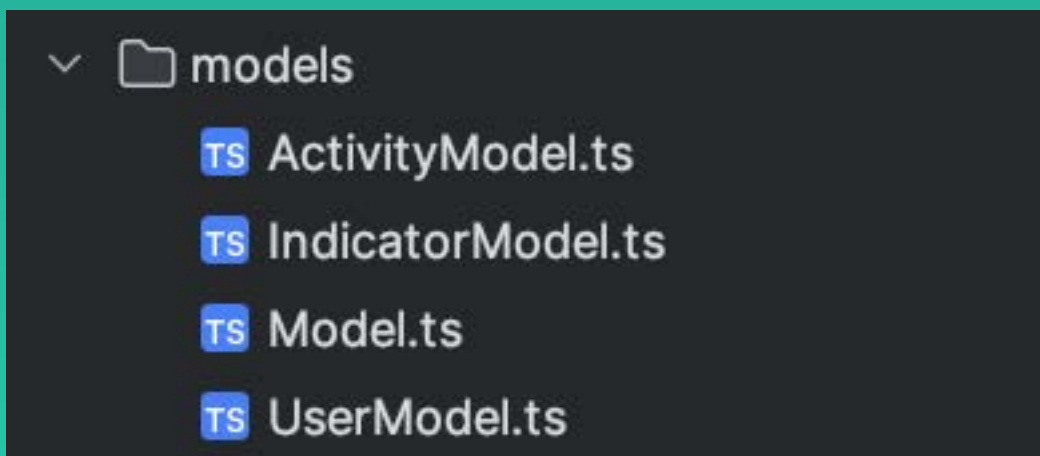
Knex Migrations - Indicators

```
1 > import ...
2
3
4 const tableName: "indicators" = 'indicators';
5
6 export async function up(knex: Knex): Promise<void> {
7   await knex.schema.createTable(tableName, { callback: function (table: Knex.CreateTableBuilder): void {
8     table.increments( columnName: 'id').primary();
9     table.integer( columnName: 'age').nullable();
10    table.integer( columnName: 'weight').nullable();
11    table.integer( columnName: 'height').nullable();
12    table.enum( columnName: 'life_style', Object.values(LifeStyle)).nullable();
13    table.integer( columnName: 'user_id').unsigned().nullable();
14    table.foreign( column: 'user_id').references( columnName: 'id').inTable( tableName: 'users').onDelete( command: 'CASCADE');
15    table.timestamps( useTimestamps: true, defaultToNow: true);
16   });
17 }
18
19 export async function down(knex: Knex): Promise<void> {
20   await knex.schema.dropTable(tableName);
21 }
```


Knex Migrations - Activities

```
1  import { Knex } from "knex";
2  import { activities } from "../../src/constants/activities";
3
4  export async function up(knex: Knex): Promise<void> {
5    await knex.schema.createTable('activities', { callback: function (table: Knex.CreateTableBuilder) : void {
6      table.increments('id').primary();
7      table.enum('activity_type', Object.values(activities)).notNullable();
8      table.string('duration').notNullable();
9      table.integer('user_id').unsigned().notNullable();
10     table.foreign('user_id').references('id').inTable('users').onDelete('CASCADE');
11     table.timestamps({ useTimestamps: true, defaultToNow: true });
12   });
13 }
14
15
16 export async function down(knex: Knex): Promise<void> {
17   await knex.schema.dropTable('activities');
18 }
```


Models



```
1 import { database } from 'root/database';
2
3 3 inheritors
4 export abstract class Model {
5   @protected static tableName?: string;
6
7   @protected static get table() {
8     if (!this.tableName) {
9       throw new Error('The table name must be defined for the model.');
```

User Model

```
1 import { Model } from 'src/models/Model';
2 import { Role, User, DefaultUserData } from 'src/@types';
3
4 export class UserModel extends Model {
5   static tableName = 'users';
6
7   public static async create<Payload>(data: Payload) : Promise<{id: number}> {
8     return super.insert<Payload & DefaultUserData>( data: {
9       ...data,
10      role: Role.User,
11    });
12  }
13
14   public static findByEmail(email: string): Promise<User | null> {
15     return this.findOneBy<
16       {
17         email: string;
18       },
19       User
20     >( data: { email });
21   }
22
23   public static findByUsername(username: string): Promise<User | null> {
24     return this.findOneBy<
25       {
26         username: string;
27       },
28       User
29     >( data: { username });
30   }
31 }
```


Indicator Model

```
1 import { Model } from 'src/models/Model';
2 import type { Indicator } from 'src/@types';
3
4 export class IndicatorModel extends Model {
5   static tableName = 'indicators';
6
7   public static async findAllByUserId(userId: number) : Promise<Indicator[]> {
8     return this.findAllBy<
9       {
10         user_id: number;
11       },
12       Indicator
13     >(data: {
14       user_id: userId,
15     });
16   }
17
18   public static async updateByUserId(userId: number, data: Partial<Indicator>) : Promise<{id: number}> {
19     return this.updateBy<
20       {
21         user_id: number;
22       },
23       Partial<Indicator>
24     >(
25       params: {
26         user_id: userId,
27       },
28       data
29     );
30   }
31 }
```

Activity Model

```
1 import { Model } from 'src/models/Model';
2 import type { Activity, Indicator } from "src/@types";
3
4 export class ActivityModel extends Model {
5   static tableName = 'activities';
6
7   public static async findAllByUserId(userId: number) : Promise<Activity[]> {
8     return this.findAllBy<
9       {
10         user_id: number;
11       },
12       Activity
13     >(data: {
14       user_id: userId,
15     });
16   }
17 }
```

Initial Point

```
1 import * as process from 'process';
2 require('dotenv').config();
3 > import ...
4
5
6
7
8
9 const app : Express = express();
10 const PORT : string | 9999 = process.env.PORT || 9999;
11
12 app.use(bodyParser.json());
13 app.use(cors());
14 app.use('/api/v1', router);
15
16 (async () : Promise<void> => {
17   try {
18     await connectToRedis();
19
20     app.listen(PORT, { callback: async () : Promise<void> => {
21       console.log(`Server is running on port ${PORT}`);
22     }});
23   } catch (error) {
24     console.error('Failed to start server:', error);
25     process.exit( code: 1);
26   }
27 })();
```


Redis

```
1 require('dotenv').config({ options: {
2   path: '../.env',
3 });
4 import process from 'process';
5 import * as Redis from 'redis';
6
7 const redisClient = Redis.createClient({ options: {
8   url: `redis://${process.env.REDIS_PASSWORD}@${process.env.REDIS_HOST}:${process.env.REDIS_PORT}`
9 });
10
11 redisClient.on('error', error => console.error('Redis Client Error', error));
12
13 const connectToRedis = async () : Promise<void> => {
14   try {
15     await redisClient.connect();
16     console.log('Connected to Redis');
17   } catch (err) {
18     console.error(`Could not connect to Redis: ${err}`);
19     process.exit(1);
20   }
21 };
22
23 export { redisClient, connectToRedis };
```

```
1 import { redisClient } from 'src/redis/index';
2
3 export class Redis {
4   public static setSession(userId: number, token: string) : Promise<string | null> = {
5     if (!userId) throw new Error('userId is required');
6     if (!token) throw new Error('token is required');
7     return redisClient.set({ args: `session:${userId}`, token });
8   }
9
10   public static getSession(userId: number) : Promise<string | null> = {
11     if (!userId) throw new Error('userId is required');
12     return redisClient.get({ args: `session:${userId}` });
13   }
14
15   public static deleteSession(userId: number) : Promise<number> = {
16     if (!userId) throw new Error('userId is required');
17     return redisClient.del({ args: `session:${userId}` });
18   }
19 }
```


Auth Middleware

```
1 import { jwt } from 'src/utils/jwt';
2 import { Redis } from 'src/redis/Redis';
3 import type { Request, Response, NextFunction } from 'express';
4 import type { UserSession } from 'src/@types';
5
6 export async function authMiddleware(req: Request, res: Response, next: NextFunction)
7 {
8   const authHeader: string | undefined = req.headers['authorization'];
9   const token: string | undefined = authHeader && authHeader.split(' ')[1];
10   const JWT_SECRET: string | undefined = process.env.JWT_SECRET;
11
12   if (!token) return res.sendStatus(401);
13
14   if (!JWT_SECRET) {
15     console.error('JWT_SECRET Not Found');
16     return res.sendStatus(500);
17   }
18
19   if (!token) return res.status(401).json({ error: 'Token not provided' });
20
21   try {
22     const userSession: UserSession = await jwt.verify<UserSession>(token);
23
24     if (!userSession) {
25       return res.sendStatus(401);
26     }
27
28     const storedToken: string | null = await Redis.getSession(userSession.id);
29
30     if (!storedToken || storedToken !== token) {
31       return res.sendStatus(401);
32     }
33
34     req.user = userSession;
35     next();
36   } catch (error) {
37     console.error('JWT_ERROR', error);
38     return res.sendStatus(401);
39   }
40 }
```


Routes

```
1 import { Router } from 'express';
2 import { authRouter } from 'src/routes/authRouter';
3 import { healthController } from 'src/controllers/healthController';
4 import { sessionController } from 'src/controllers/sessionController';
5 import { authMiddleware } from 'src/middlewares/authMiddleware';
6 import { userRouter } from 'src/routes/userRouter';
7 import { suggestionRouter } from 'src/routes/suggestionRouter';
8 import { indicatorRouter } from 'src/routes/indicatorRouter';
9
10 export const router : Router = Router( options: { mergeParams: true });
11
12 router.get( path: '/health', healthController);
13 router.use('/auth', authRouter);
14 router.use(authMiddleware);
15 router.get( path: '/session', sessionController);
16 router.use('/user', userRouter);
17 router.use('/suggestion', suggestionRouter);
18 router.use('/indicator', indicatorRouter);
```


Expectations

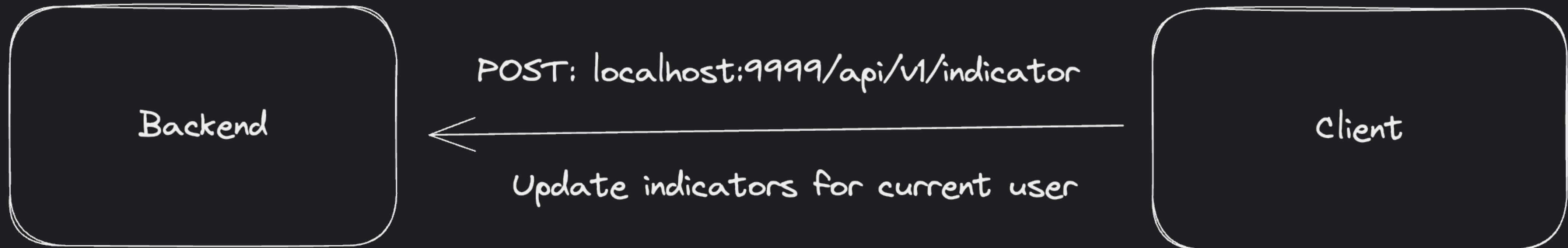
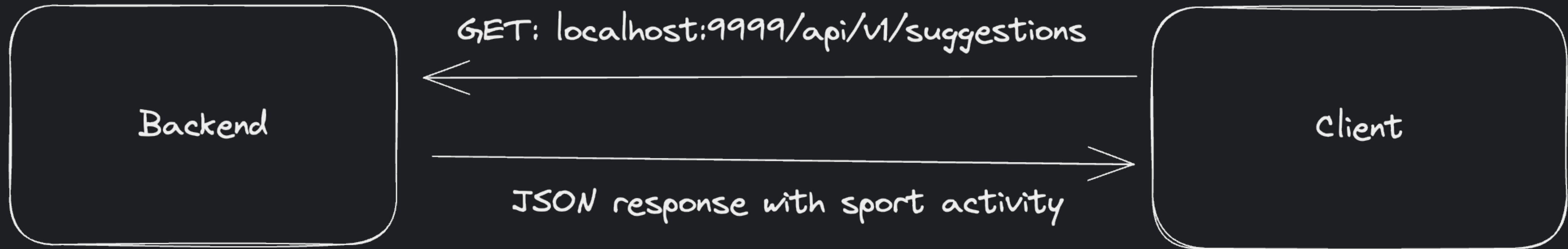
GET: localhost:9999/api/v1/suggestions

```
Body Cookies Headers (8) Test Results Status: 200 OK Time: 37.94 s Size: 1.42 KB Save as example
Pretty Raw Preview Visualize JSON
1 {
2   "data": [
3     {
4       "activity_type": "cycling",
5       "duration": 45,
6       "description": "A low-impact cardiovascular activity that can be done outdoors or on a stationary bike.",
7       "water_consume": 750,
8       "explanation_why": "Cycling is a great activity choice for you as it provides both cardiovascular and strength benefits. It is a low-impact exercise that puts less stress on your joints, making it suitable for all fitness levels. Cycling also helps to improve cardiovascular endurance, build leg muscles, and burn calories.",
9       "plan": [
10        "Start by warming up for 5 minutes with a light pedal at an easy pace.",
11        "Increase your pace and intensity for the next 30 minutes, aiming to maintain a moderate level of effort.",
12        "If you're cycling outdoors, find a route with varied terrain to challenge your leg muscles and add some resistance to your workout. If you're using a stationary bike, increase the resistance level periodically throughout the workout.",
13        "Cool down for the last 5 minutes by gradually decreasing your pace and intensity.",
14        "Remember to stay hydrated throughout the workout. Take sips of water every 10-15 minutes.",
15        "After the workout, stretch your leg muscles to prevent tightness and promote recovery."
16      ]
17     }
18   ]
19 }
```


Expectations

GET: localhost:9999/api/v1/suggestions

```
Body Cookies Headers (8) Test Results Status: 200 OK Time: 14.29 s Size: 1.07 KB Save as example
Pretty Raw Preview Visualize JSON ↕
1 {
2   "data": [
3     "activity_type": "skiing",
4     "duration": 60,
5     "description": "Hit the slopes and enjoy skiing down the mountain. Skiing is a great full-body workout that improves cardiovascular fitness
6       and strengthens the lower body muscles.",
7     "water_consume": 750,
8     "explanation_why": "Based on your previous activity of crossfit and your super active lifestyle, skiing is a great activity for you. It
9       provides a combination of cardiovascular exercise and muscle strengthening, helping to improve your overall fitness level.",
10    "plan": [
11      "Put on your ski gear and make sure everything is properly adjusted.",
12      "Find a suitable slope for your skiing level.",
13      "Start skiing down the slope, maintaining a good posture and bending your knees.",
14      "Continue skiing for the duration of your workout, taking breaks as needed.",
15      "Enjoy the experience and have fun!"
16    ]
17  ]
18 }
```

OpenAI API config

```
1 import OpenAI from 'openai';  
2  
3 export const openai : OpenAI = new OpenAI({  
4   apiKey: process.env.OPENAI_API_KEY,  
5 });
```

Controller

```
1  import { Request, Response } from 'express';
2  import { IndicatorModel } from 'src/models/IndicatorModel';
3  import { ActivityModel } from 'src/models/ActivityModel';
4  import { getSportActivitySuggestion } from 'src/helpers/getSportActivitySuggestion';
5
6  export const getSuggestionController = async (req: Request, res: Response) : Promise<void> => {
7    try {
8      const [indicator : Indicator ] = await IndicatorModel.findAllByUserId(req.user.id);
9      const [lastActivity : Activity ] = await ActivityModel.findAllByUserId(req.user.id);
10
11     const result : Activity & ActivityDescription = await getSportActivitySuggestion(indicator, lastActivity);
12
13     await ActivityModel.insert( data: {
14       activity_type: result.activity_type,
15       duration: result.duration,
16       user_id: req.user.id,
17     });
18
19     res.json( body: {
20       data: result,
21     });
22   } catch (error: unknown) {
23     console.log(error);
24   } res.status( code: 500 ).json( body: {
25     message: 'Internal Server Error',
26   });
27 }
28 }
```


System Prompt

```
1 import { Indicator, Activity, ActivityDescription } from 'src/@types';
2 import { activities } from 'src/constants/activities';
3 import { openai } from 'src/configs/openai';
4
5 export async function getSportActivitySuggestion(
6   indicator: Indicator,
7   lastActivity?: Activity
8 ): Promise<Activity & ActivityDescription> {
9   const systemPrompt: string = `
10   You will be provided with indicators as age, weight, height, lifestyle
11   and previous activity type with duration if the exist,
12   and your task is to return sports activity plan in JSON format depends on those indicators.
13   The plan should include the type of activity, the duration in minutes, a good description
14   of how to do the activity, the recommended water consumption in milliliters,
15   and the plan like step by step what to do during the activity.
16
17   Example of the response:
18   {
19     "activity_type": "${activities.sort( compareFn: () => Math.random() -- 0.5).join(' || ')},
20     "duration": 30,
21     "description": "A continuous run at a moderate pace to improve cardiovascular endurance.",
22     "water_consume": 500
23     "distance": 3000,
24     "explanation_why": "The explanation why this activity is good for you based on you previous activity and indicators",
25     "plan": ["First step description", "Second step description", "Third step description"]
26   }
27 `;
```

User Prompt

```
28
29 ··const lastActivityMessage : string = lastActivity
30 ··?`
31 ··  -- Last activity type: ${lastActivity.activity_type}
32 ··  -- Last activity duration: ${lastActivity.duration}
33 ··  `
34 ··: '';
35
36 ··const userPrompt : string = `
37 ··  -- Age: ${indicator.age}
38 ··  -- Weight: ${indicator.weight}
39 ··  -- Height: ${indicator.height}
40 ··  -- Lifestyle: ${indicator.life_style}
41 ··  ${lastActivityMessage}
42 ··  `;
43
44 ··console.log('userPrompt', userPrompt);
```


Completions

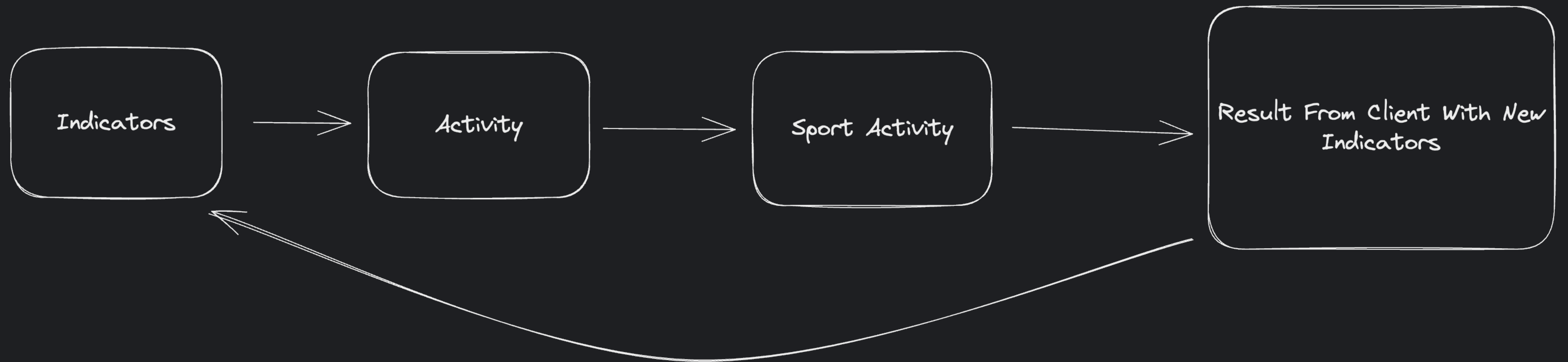
```
45
46   const completion : ChatCompletion = await openai.chat.completions.create( body: {
47     messages: [
48       { role: 'system', content: systemPrompt },
49       { role: 'user', content: userPrompt },
50     ],
51     model: 'gpt-3.5-turbo',
52     temperature: 0.9,
53   });
54
55   return JSON.parse( text: completion.choices[0].message.content || '{}');
56 }
```

Suggestion Controller

Update Activity by AI result

```
2 import { IndicatorModel } from 'src/models/IndicatorModel';
3 import { ActivityModel } from 'src/models/ActivityModel';
4 import { getSportActivitySuggestion } from 'src/helpers/getSportActivitySuggestion';
5
6 export const getSuggestionController = async (req: Request, res: Response) : Promise<void> => {
7   try {
8     const [indicator : Indicator] = await IndicatorModel.findAllByUserId(req.user.id);
9     const [lastActivity : Activity] = await ActivityModel.findAllByUserId(req.user.id);
10
11    const result : Activity & ActivityDescription = await getSportActivitySuggestion(indicator, lastActivity);
12
13    await ActivityModel.insert( data: {
14      activity_type: result.activity_type,
15      duration: result.duration,
16      user_id: req.user.id,
17    });
18
19    res.json( body: {
20      data: result,
21    });
22  } catch (error: unknown) {
23    console.log(error);
24    res.status( code: 500 ).json( body: {
25      message: 'Internal Server Error',
26    });
27  }
28 }
```

Update Indicator



Update Indicator

```
1  import { Request, Response } from 'express';
2  import { IndicatorModel } from 'src/models/IndicatorModel';
3
4  export const updateIndicatorController = async (req: Request, res: Response) : Promise<void> => {
5    try {
6      const { id: number } = await IndicatorModel.updateByUserId(req.user.id, data: {
7        age: req.body.age,
8        weight: req.body.weight,
9        height: req.body.height,
10       life_style: req.body.life_style,
11     });
12
13     res.json( body: {
14       id,
15     });
16   } catch (error: unknown) {
17     console.log(error);
18     res.status( code: 500 ).json( body: {
19       message: 'Internal Server Error',
20     });
21   }
22 };
```

POST: localhost:9999/api/v1/indicator

Activities

WHERE ORDER BY

	id	activity_type	duration	user_id	created_at	updated_at
1	1	crossfit	60	11	2023-11-07 22:00:44...	2023-11-07 2
2	2	crossfit	60	11	2023-11-07 22:01:27...	2023-11-07 2
3	3	crossfit	45	11	2023-11-07 22:03:33...	2023-11-07 2
4	4	crossfit	60	11	2023-11-07 22:06:35...	2023-11-07 2
5	5	running	30	11	2023-11-07 22:07:55...	2023-11-07 2
6	6	cycling	45	11	2023-11-07 22:09:04...	2023-11-07 2
7	7	skiing	60	11	2023-11-07 22:47:08...	2023-11-07 2

GitHub

<https://github.com/antonkalik/pullap-ai-server>

Thank you
for the
attention