### C:\\_zhangyidaFiles\C02\_Study\207\_Projects\Project250514\_nodejs\system> npm run dev npm : 无法加载文件 C:\\_zhangyidaFiles\C06\_LanguageEnvironment\JavaScript\nodejs\npm.ps1，因为在此系统上禁止运行脚本。有关详细信息，请参阅 https :/go.microsoft.com/fwlink/?LinkID=135170 中的 about\_Execution\_Policies。 所在位置 行:1 字符: 1 + npm run dev + ~~~ + CategoryInfo : SecurityError: (:) []，PSSecurityException + FullyQualifiedErrorId : UnauthorizedAccess

### ****方法 2：永久更改执行策略（适合开发环境）****

如果想长期允许运行脚本（仅限当前用户）：

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

* **RemoteSigned**：允许运行本地脚本，远程脚本需签名（安全推荐）。

**CurrentUser**：仅影响当前用户，不影响系统全局。

完成后，再次运行：

npm run dev

好的，这太棒了！您提供了数据库结构（schema）和连接字符串，这些是设置后端以保存分数的关键部分。

要将玩家数据（用户名和分数）保存到您在 Neon 上的 PostgreSQL 数据库中，您需要创建一个后端 API 端点。您的前端 React 代码已经配置为向 /api/save-score 发送 POST 请求。我们现在需要实现这个端点。

鉴于您的前端设置（使用了 "use client"; 以及 axios 调用相对路径 /api/...），您很可能正在一个 **Next.js 项目** 中工作。Next.js API 路由 (API Routes) 是在您的项目中直接创建此后端端点的绝佳方式。

以下是您需要做的：

**1. 安装 PostgreSQL 客户端库：** 如果您在后端使用 Node.js（Next.js API 路由就是基于 Node.js），您需要一个 PostgreSQL 客户端库。最常用的是 pg。

在您的项目根目录打开终端并运行：

Bash

npm install pg# 或者

yarn add pg

如果您使用 TypeScript，您可能还需要它的类型定义：

Bash

npm install --save-dev @types/pg# 或者

yarn add --dev @types/pg

**2. 创建 API 路由 (**/api/save-score**):** 在您的 Next.js 项目中，在以下路径创建文件：pages/api/save-score.ts (如果您使用 TypeScript) 或 pages/api/save-score.js (如果您使用 JavaScript)。

以下是 pages/api/save-score.ts 的示例实现：

TypeScript

// pages/api/save-score.tsimport { NextApiRequest, NextApiResponse } from 'next';import { Pool } from 'pg'; // 从 'pg' 模块导入 Pool 类

// 重要：将您的连接字符串存储在环境变量中！// 不要将其直接硬编码到代码中，尤其是密码。const connectionString = process.env.POSTGRES\_URL;

// 初始化一个连接池。// 通常最好在处理函数外部创建连接池，以便在多个请求之间重用。let pool: Pool;

if (!connectionString) {

console.error("POSTGRES\_URL 环境变量未设置！");

// 在实际应用中，您可能会阻止应用启动或更优雅地处理此问题。

// 目前，如果未设置，我们允许它在运行时可能失败。

} else {

pool = new Pool({

connectionString: connectionString,

// 您 Neon 连接字符串中的 `sslmode=require` 应该会指示 `node-postgres` 使用 SSL。

// 如果遇到 SSL 问题，您可能需要在此处添加一个 ssl 对象，例如：

// ssl: {

// rejectUnauthorized: false, // 请谨慎使用，主要用于开发/调试

// },

// 如果需要，请参阅 Neon 的文档以获取最安全的 SSL 配置。

});

}

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

if (!pool) {

return res.status(500).json({ error: '数据库连接未配置。' });

}

if (req.method === 'POST') {

const { player\_name, score } = req.body;

// 后端基础验证 (务必在后端进行验证!)

if (!player\_name || typeof player\_name !== 'string' || player\_name.trim().length === 0 || player\_name.trim().length > 255) {

return res.status(400).json({ error: '无效的玩家名称。它必须是一个非空字符串，最多255个字符。' });

}

if (typeof score !== 'number' || score < 0 || !Number.isInteger(score)) {

return res.status(400).json({ error: '无效的分数。它必须是一个非负整数。' });

}

try {

// 使用参数化查询以防止 SQL 注入

const query = `

INSERT INTO public.player\_score (player\_name, score)

VALUES ($1, $2)

RETURNING player\_name, score;

`;

const values = [player\_name.trim(), score];

const result = await pool.query(query, values);

res.status(201).json({

message: '分数保存成功！',

data: result.rows[0], // 新插入的行

});

} catch (error: any) {

console.error('将分数保存到数据库时出错：', error);

res.status(500).json({ error: '保存分数失败。', details: error.message });

}

} else {

// 处理任何其他 HTTP 方法

res.setHeader('Allow', ['POST']);

res.status(405).end(`方法 ${req.method} 不允许`);

}

}

**3. 配置环境变量：** 您的 PostgreSQL 连接字符串包含敏感信息（您的密码）。您**永远不应该**将其直接硬编码到您的前端或后端源代码中，尤其是那些会提交到代码仓库的代码。

在您的 Next.js 项目的根目录创建一个名为 .env.local 的文件（如果它尚不存在）。将您的连接字符串添加到此文件中：

代码段

# .env.local

# 将 YOUR\_ACTUAL\_PASSWORD 替换为您的 Neon 连接字符串中的真实密码部分

POSTGRES\_URL="postgresql://neondb\_owner:YOUR\_ACTUAL\_PASSWORD@ep-late-lab-a41enk0r-pooler.us-east-1.aws.neon.tech/neondb?sslmode=require"

* 将 YOUR\_ACTUAL\_PASSWORD 替换为实际的密码（您之前隐去的那部分 npg\_\*\*\*\*\*\*\*\*\*）。
* .env.local 文件应列在您的 .gitignore 文件中，以防止其被提交到版本控制。
* 当您部署您的 Next.js 应用程序时（例如，部署到 Vercel、Netlify 或您自己的服务器），您需要在您的托管平台的设置中设置此 POSTGRES\_URL 环境变量。

**API 路由代码说明：**

* import { Pool } from 'pg';: 从 pg 库导入 Pool（连接池）。连接池比为每个请求创建新连接更高效。
* process.env.POSTGRES\_URL: 从环境变量中读取连接字符串。
* new Pool(...): 创建一个新的连接池实例。
  + 您 Neon 连接字符串中的 sslmode=require 非常重要，pg 库应该会正确处理它以确保安全连接。
* handler(req: NextApiRequest, res: NextApiResponse): 这是标准的 Next.js API 路由处理函数。
* if (req.method === 'POST'): 确保此逻辑仅针对 POST 请求运行，这与您的前端发送请求的方式一致。
* const { player\_name, score } = req.body;: 从您的前端发送的数据中提取 player\_name 和 score。
* **验证**: 代码中包含了对 player\_name 和 score 的基本后端验证。这一点至关重要，因为您不能信任来自客户端的数据，即使您在前端进行了验证。
* **参数化查询**: SQL 查询使用 $1 和 $2作为占位符。实际值通过一个数组传递。这对于防止 SQL 注入攻击**极其重要**。
* pool.query(query, values): 执行 SQL 查询。
* res.status(201).json(...): 发送状态码为 201 (Created) 的成功响应，并附带已保存的数据。
* **错误处理**: 包含一个 try...catch 块来处理潜在的数据库错误，并发送适当的错误响应。

**您还需要了解/执行的事项：**

1. **完整密码：** 确保在您的 .env.local 文件中（以及在您的部署环境变量中）使用完整且正确的密码。npg\_\*\*\*\*\*\*\*\*\* 只是一个占位符。
2. **Neon 数据库激活：** 确保您的 Neon 数据库正在运行并且可以访问。
3. **表创建：** 您已经提供了 CREATE TABLE 语句。请确保这个 public.player\_score 表存在于您 Neon 上的 neondb 数据库中。
4. **部署：** 当您部署应用程序时，请记住在您的托管平台上配置 POSTGRES\_URL 环境变量。
5. **错误监控/日志记录：** 对于生产环境的应用程序，您可能需要为您的后端 API 设置更强大的错误日志记录和监控。
6. **前端代码：** 您在 SnakeGamePage.tsx 中的现有前端 saveScore 函数已经正确设置为调用此 /api/save-score 端点。前端代码无需更改。

有效期三十天的git令牌：（250515起）

ghp\_ElDlLL8fmqI0CyIgn7vdy6PUHrAIFe2AWZ07

美国圣何赛03

git config --global http.proxy http://127.0.0.1:7897

PS C:\\_zhangyidaFiles\C02\_Study\207\_Projects\Project250515\_nodejs\sysyem02> git push

Enumerating objects: 7, done.

Counting objects: 100% (7/7), done.

Delta compression using up to 16 threads

Compressing objects: 100% (4/4), done.

Writing objects: 100% (4/4), 500 bytes | 500.00 KiB/s, done.

Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (3/3), completed with 3 local objects.

To https://github.com/xiaxiaoyu8/Community-and-Organization-Management-System.git

bb9f551..fa00626 main -> main

PS C:\\_zhangyidaFiles\C02\_Study\207\_Projects\Project250515\_nodejs\sysyem02>