

W11-P1: Supabase Database Setting

The screenshot shows the Supabase Settings interface. The left sidebar has a 'Database' section selected. The main area is titled 'Database Settings' and contains a 'Connection info' section with fields for Host, Database name, Port, User, and Password. A red box highlights this 'Connection info' section. Below it are sections for 'Database password' and 'Connection string', with links for various database drivers at the bottom.

Connection info

Host	db.esqwkehuhxspbqsektof.supabase.co	<input type="button" value="Copy"/>
Database name	postgres	<input type="button" value="Copy"/>
Port	5432	<input type="button" value="Copy"/>
User	postgres	<input type="button" value="Copy"/>
Password	[The password you provided when you created this project]	

Database password

You can use this password to connect directly to your Postgres database.

Reset Database Password

Connection string

PSQL URL Golang JDBC .NET Nodejs PHP Python

bda6064 aosihfvdps Wed Apr 26 13:23:42 2023 +0800 Merge branch 'main' of

<https://github.com/aosihfvdps/1112-2A-db-demo-410410319>

47ff584 aosihfvdps Wed Apr 26 13:20:38 2023 +0800 this is w09

W11-P2: Connecting Supabase

The screenshot shows a Visual Studio Code interface with two code files open:

- app.js**:

```
1 var createError = require('http-errors');
2 var express = require('express');
3 var path = require('path');
4 var cookieParser = require('cookie-parser');
5 var logger = require('morgan');

6 const dotenv = require('dotenv');
7 dotenv.config();

8 const db = require('../utils/database');

11 //const testDBConnection = require('../utils/test-db');

13 var indexRouter = require('../routes/index');
14 var usersRouter = require('../routes/users');

16 const crownRouter_19 = require('../routes/crown_19');
17 const crown2Router_19 = require('../routes/crown2_19');

19 var app = express();
20
21 // view engine setup
22 app.set('views', path.join(__dirname, 'views'));
23 app.set('view engine', 'ejs');
```
- database.js**:

```
1 const { Pool } = require('pg');
2 let pool;
3
4 if(process.env.DATABASE === 'SUPABASE'){
5   pool = new Pool({
6     user: 'postgres',
7     host: process.env.SUPABASE_HOST,
8     port: '5432',
9     database: 'postgres',
10    password: process.env.PASSWORD,
11  });
12  console.log(`Connecting Supabase PostgreSQL running on ${pool.options.host}`);
13}
14 else{
15   pool = new Pool({
16     user: 'postgres',
17     host: 'localhost',
18     port: '5432',
19     database: 'crown_19',
20     password: '0000',
21   });
22  console.log(`Connecting local PostgreSQL running on ${pool.options.host}`);
23}
24
```

The terminal output shows the connection status:

```
Error: connect ECONNREFUSED ::1:5432
  at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1487:16) {
  errno: -4078,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '::1',
  port: 5432
}
[nodemon] restarting due to changes...
[nodemon] starting `node ./bin/www`
Connecting supabase PostgreSQL running on postgres database
Web server running on port 3000
```

The status bar indicates the current time is 2023-04-26 02:14.

W11-P3: Put md img file into Supabase

The screenshot shows a Visual Studio Code interface with two code files open:

- app.js**:

```
1 ## W11-P1: Supabase Database Setting
2
3 [[w11-p1.png]](https://esqkuehuhxspbqsektof.supabase.co/storage/v1/object/public/demo-19/md_2A_img/w11-p1.png?t=2023-04-26T06%3A36%3A24.540Z)
4
5 bda6064 aosishfvdps Wed Apr 26 13:23:42 2023 +0800 Merge branch 'main' of https://github.com/aosishfvdps/1112-2A-db-demo-410410319
6 47ff584 aosishfvdps Wed Apr 26 13:20:38 2023 +0800 this is w09
7
8 ## W11-P2: Connecting Supabase
9
10 [[w11-p2.png]](https://esqkuehuhxspbqsektof.supabase.co/storage/v1/object/public/demo-19/md_2A_img/w11-p2.png?t=2023-04-26T06%3A36%3A24.540Z)
```
- w11.md**:

```
W11-P1: Supabase Database Setting
[[w11-p1.png]](https://esqkuehuhxspbqsektof.supabase.co/storage/v1/object/public/demo-19/md_2A_img/w11-p1.png?t=2023-04-26T06%3A36%3A24.540Z)

W11-P2: Connecting Supabase
[[w11-p2.png]](https://esqkuehuhxspbqsektof.supabase.co/storage/v1/object/public/demo-19/md_2A_img/w11-p2.png?t=2023-04-26T06%3A36%3A24.540Z)
```

The terminal output shows the connection status:

```
Connecting supabase PostgreSQL running on postgres database
Web server running on port 3000
```

The right side of the interface shows a preview of the Supabase database settings and a terminal window. The status bar indicates the current time is 2023-04-26 02:39.

```
git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-4-25"
```

W12-P1: Create foreign key from cat_id (shop2_xx) to id (category2_xx)

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like FTS Configurations, FTS Dictionaries, and Tables. The 'Tables' section is expanded, and the 'shop2_19' table is selected. A red box highlights the 'Constraints' section under 'shop2_19', which contains two entries: 'fkey1' and 'shop_xx_pkey'. The right pane is the 'shop2_19' object editor. The 'Constraints' tab is active, showing a table with three columns: 'Name', 'Columns', and 'Referenced Table'. The 'fkey1' row is highlighted with a red box and has the following values:

Name	Columns	Referenced Table
fkey1	(cat_id) -> (id)	public.category2_19

Below the table are buttons for 'Close', 'Reset', and 'Save'.

W12-P2: give SQL to get products based on a category

The screenshot shows the pgAdmin interface. On the left, the database browser displays two tables: 'category2_19' and 'shop2_19'. The 'shop2_19' table is currently selected. Both tables have six columns: id, name, size, remote_url, local_url, and link_url. A red box highlights the columns for both tables. In the center, the query editor contains the following SQL code:

```
1 select C.name as category, S.id, S.name, price, S.local_url
2 from category2_19 as C, shop2_19 as S
3 where S.cat_id = C.id
4 and C.name = 'womens'
```

A red box highlights this entire query. Below the query editor, the data output shows the results of the query:

category	id	name	price	local_url
womens	28	Blue Tanktop	25	/img/womens/blue-tank.png
womens	29	Floral Blouse	20	/img/womens/floral-blouse.png
womens	30	Floral Dress	80	/img/womens/floral-skirt.png
womens	31	Red Dots Dress	80	/img/womens/red-polka-dot-dress.png
womens	32	Striped Sweater	45	/img/womens/striped-sweater.png
womens	33	Yellow Track Suit	135	/img/womens/yellow-track-suit.png
womens	34	White Blouse	20	/img/womens/white-vest.png

Total rows: 7 of 7 Query complete 0:00:00.429 Ln 1, Col 50

W12-P3: implement route /crown2_xx/shop2_xx/:category, and show json data regarding some category in Browser

The screenshot shows a Visual Studio Code environment. On the left, the Explorer sidebar shows a project structure with files like 'crown2_19.js', 'shop2_19.ejs', and 'shop2_19.sql'. The 'crown2_19.js' file is open in the editor, showing a Node.js route definition:

```
router.get('/shop2_19/:category', async function (req, res) {
  let results = await db.query(`select C.name as category, S.id, S.name, S.size, S.remote_url, S.local_url, S.link_url from category2_19 as C, shop2_19 as S where S.cat_id = C.id and C.name = ${req.params.category}`);
  res.json(results.rows);
})
```

A red box highlights the database query in the code. To the right, a browser window shows the URL `localhost:3000/crown2_19/shop2_19/womens`. The page displays the following JSON data:

```
[{"category": "womens", "id": 28, "name": "Blue Tanktop", "size": "S", "remote_url": "/img/womens/blue-tank.png", "local_url": "/img/womens/blue-tank.png"}, {"category": "womens", "id": 29, "name": "Floral Blouse", "size": "M", "remote_url": "/img/womens/floral-blouse.png", "local_url": "/img/womens/floral-blouse.png"}, {"category": "womens", "id": 30, "name": "Floral Dress", "size": "L", "remote_url": "/img/womens/floral-skirt.png", "local_url": "/img/womens/floral-skirt.png"}, {"category": "womens", "id": 31, "name": "Red Dots Dress", "size": "XL", "remote_url": "/img/womens/red-polka-dot-dress.png", "local_url": "/img/womens/red-polka-dot-dress.png"}, {"category": "womens", "id": 32, "name": "Striped Sweater", "size": "S", "remote_url": "/img/womens/striped-sweater.png", "local_url": "/img/womens/striped-sweater.png"}, {"category": "womens", "id": 33, "name": "Yellow Track Suit", "size": "M", "remote_url": "/img/womens/yellow-track-suit.png", "local_url": "/img/womens/yellow-track-suit.png"}, {"category": "womens", "id": 34, "name": "White Blouse", "size": "S", "remote_url": "/img/womens/white-vest.png", "local_url": "/img/womens/white-vest.png"}]
```

W12-P4: implement route /crown2_xx/shop2_xx/:category, and show in shop2_xx.ejs

localhost:3000/crown2_19/shop2_19/hats

HATS
Chen Ji Wei 410410319

Brown Brim	25	Blue Beanie	18	Brown Cowboy	35	Grey Brim	25
Green Beanie	18	Palm Tree Cap	14	Red Beanie	18	Wolf Cap	14

localhost:3000/crown2_19/shop2_19/womens

WOMENS
Chen Ji Wei 410410319

Blue Tanktop	25	Floral Blouse	20	Floral Dress	80	Red Dots Dress	80
	25		20		80		80

VS Code screenshot showing the implementation of the route /crown2_xx/shop2_xx. The code uses an async function to query the database for products and render them into a template.

```

    router.get('/shop2_19', async function (req, res, next) {
      try {
        let results = await db.query(`SELECT * FROM shop2_19`);
        console.log('results', JSON.stringify(results));
        res.render('crown2_19/shop2_19', {
          data: results.rows,
          category: 'All products',
          name: 'Chen Ji Wei',
          ID: '410410319'
        });
      } catch(error) {
        console.log(error);
      }
    });

    router.get('/shop2_19/:category', async function (req, res, next) {
      try {
        let results = await db.query(`SELECT * FROM shop2_19`);
        console.log('results', JSON.stringify(results));
        res.render('crown2_19/shop2_19', {
          data: results.rows,
          category: req.params.category,
          name: 'Chen Ji Wei',
          ID: '410410319'
        });
      } catch(error) {
        console.log(error);
      }
    });
  
```

Browser screenshot showing the resulting product listing page titled "ALL PRODUCTS" by Chen Ji Wei. It displays four categories of hats: Brown Brim, Blue Beanie, Brown Cowboy, and Grey Brim, each with a small thumbnail image.

W12-P5: implement route /crown2_xx/shop2_xx to show all products in shop2_xx.ejs

VS Code screenshot showing the implementation of the route /crown2_xx/shop2_xx. The code is identical to the previous one, using an async function to query the database and render the results into a template.

```

    router.get('/shop2_19', async function (req, res, next) {
      try {
        let results = await db.query(`SELECT * FROM shop2_19`);
        console.log('results', JSON.stringify(results));
        res.render('crown2_19/shop2_19', {
          data: results.rows,
          category: 'All products',
          name: 'Chen Ji Wei',
          ID: '410410319'
        });
      } catch(error) {
        console.log(error);
      }
    });

    router.get('/shop2_19/:category', async function (req, res, next) {
      try {
        let results = await db.query(`SELECT * FROM shop2_19`);
        console.log('results', JSON.stringify(results));
        res.render('crown2_19/shop2_19', {
          data: results.rows,
          category: req.params.category,
          name: 'Chen Ji Wei',
          ID: '410410319'
        });
      } catch(error) {
        console.log(error);
      }
    });
  
```

Browser screenshot showing the resulting product listing page titled "ALL PRODUCTS" by Chen Ji Wei. It displays four categories of hats: Brown Brim, Blue Beanie, Brown Cowboy, and Grey Brim, each with a small thumbnail image.

W12-P6: make each category link works in /crown2_xx page

The screenshot shows two windows side-by-side. On the left is a code editor in Visual Studio Code with the file `index.ejs` open. The code is EJS (Embedded JavaScript) and displays a menu item loop. A red box highlights the section of code that generates links:

```
<a href="<%= data[i].link_url %>" class="content">
  <h1 class="title"><%= data[i].name.toUpperCase() %></h1>
  <span class="subtitle">SHOP NOW</span>
</a>
```

On the right is a "Table editor" window from Supabase. It shows a table named `category_xx` with one column `link_url` and five rows of data. A red box highlights the data in this table:

link_url
/crown2_19/shop2_19/hats
/crown2_19/shop2_19/mens
/crown2_19/shop2_19/womens
/crown2_19/shop2_19/sneakers
/crown2_19/shop2_19/jackets