

# Project Status & Debugging Report: Reliserv API

## 1. System Architecture

The project is built on a containerized microservices architecture:

- **Backend:** Node.js server running in a development environment via ts-node-dev.
- **Database:** PostgreSQL 15+ hosted within a Docker container.
- **ORM:** Prisma, which acts as the bridge between the TypeScript code and the SQL database.
- **API Testing:** Postman extension integrated into VS Code for endpoint validation.

## 2. Technical Specifications

The following configuration is defined in the apps/api/.env file:

- **Port:** 4000.
- **Database URL:** postgresql://reliserv:reliserv@localhost:5432/reliserv?schema=public.
- **Credentials:** Username reliserv and Password reliserv.

## 3. Timeline of Actions & Findings

### *Phase 1: Environment Stabilization*

- **Observation:** The API was unable to read configuration due to incorrect file naming or placement.
- **Action:** Validated and renamed the .env file within the apps/api directory.
- **Result:** The API now successfully initializes and listens on <http://localhost:4000>.

### *Phase 2: Database Infrastructure Reset*

- **Observation:** Repeated P1000 Authentication errors indicated a credential mismatch between the code and the running Docker container.
- **Action:** Executed a "hard reset" using docker compose -f infra/docker-compose.yml down -v to purge persistent volumes.
- **Finding:** A "volume not found" error during cleanup confirmed that previous data volumes were successfully removed, leaving a clean slate for the next deployment.

### *Phase 3: Schema Synchronization Attempt*

- **Observation:** Running npx prisma db push from the root directory failed because the schema.prisma file is localized within the apps/api/prisma directory.

- **Action:** Navigated to apps/api to execute the push command.
- **Result:** Continued P1000 errors suggest the Docker container is still not accepting the reliserv password despite volume purging.

## 4. Root Cause Analysis (Pending Verification)

Given the persistence of the P1000 error after 50+ reset attempts, the following "Invisible Blockers" are the primary suspects:

Suspect	Description
<b>Orphaned Database</b>	A local installation of PostgreSQL (e.g., pgAdmin/Postgres.app) is running on the host machine, occupying Port 5432 and blocking Docker.
<b>Hardcoded Config</b>	The file infra/docker-compose.yml may have POSTGRES_PASSWORD hardcoded to something other than reliserv.
<b>Caching</b>	Docker Desktop may be caching image layers that contain old environment variables.

## 5. Proposed Resolution Strategy

To close this report, we recommend a "**Port & Config Audit**":

1. **Check Port 5432:** Run `netstat -ano | findstr :5432` to see if a non-Docker process is using the port.
2. **Verify Compose File:** Manually inspect `infra/docker-compose.yml` to ensure the environment variables match the `.env` file exactly.
3. **Manual Login:** Attempt to log into the container via terminal to see if it accepts the password manually.

```

PS C:\Users\psaig\OneDrive\Desktop\ASE Project\reiserv> cd apps/api
PS C:\Users\psaig\OneDrive\Desktop\ASE Project\reiserv\apps\api> npx prisma generate
Loaded Prisma config from prisma.config.ts.

Prisma schema loaded from prisma\schema.prisma.

✓ Generated Prisma Client (v7.3.0) to ./node_modules@prisma/client in 153ms
Start by importing your Prisma Client (See: https://pris.ly/d/importing-client)

PS C:\Users\psaig\OneDrive\Desktop\ASE Project\reiserv\apps\api> npm run dev
> reiserv-api@1.0.0 dev
> ts-node-dev --transpile-only src/server.ts
[INFO] 16:18:24 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.9.3)
API running on http://localhost:4000

```

0.33% / 800% (8 CPUs available)      23.36MB / 7.52GB

Containers							
	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	infra	-	-	-	0.33%	16 seconds ago	<span style="color: blue;">...</span> <span style="color: red;">Delete</span>
<input type="checkbox"/>	reiserv_redis	d852f09e9088	redis:7	6379:6379	0.3%	16 seconds ago	<span style="color: blue;">...</span> <span style="color: red;">Delete</span>
<input type="checkbox"/>	reiserv_postgres	d63f8b985393	postgres:16	5432:5432	0.03%	16 seconds ago	<span style="color: blue;">...</span> <span style="color: red;">Delete</span>

New HTTP Request

Default workspace

http://localhost:4000

GET /

Params Headers (20) Body Scripts Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (20) Test Results

Pretty Raw Preview HTML

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Error</title>
6   </head>
7   <body>
8     <pre>Cannot GET /</pre>
9   </body>
10 </html>
11
12
13

```

Status: 404 Not Found Time: 28 ms Size: 846 B

Build with Agent

Start by importing your Prisma Client (See: https://pris.ly/d/importing-client)

PS C:\Users\psaig\OneDrive\Desktop\ASE Project\reiserv> npm run dev
> reiserv-api@1.0.0 dev
> ts-node-dev --transpile-only src/server.ts
[INFO] 17:05:00 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.9.3)
API running on http://localhost:4000
GET / 404 5.399 ms: 139

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:4000/v1/auth/signup?Content-Type=Content-Type`. The request body is a JSON object:

```
1 {  
2   "name": "Test User",  
3   "email": "test@example.com",  
4   "password": "Password123!",  
5   "role": "CUSTOMER"  
6 }
```

The response status is "Could not send request" with the error message "Error: read ECONNRESET". Below the request, the terminal shows the following code and an error message:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE  
Start by importing your Prisma Client (See: https://pris.ly/d/importing-client)  
16 const { name, email, phone, password, role } = parsed.data;  
17  
→ 18 const exists = await prisma.user.findUnique(  
Authentication failed against the database server, the provided database credentials for 'reliserv' are not valid
```

CHAT CHAT + ...

Build with AI AI responses may be included. Generate Agent Instructions onto your codebase.

SUGGESTED ACTIONS Build Workspace Show Options

Add Context... Describe what to build next