

Polar Edge Analytics

Documentation for our scouting website.

- Getting Started
 - What Do I Need to Know?
 - Discord OAuth
 - Creating Your Development Setup
- System Architecture

Getting Started

Resources for getting started developing Polar Edge Analytics locally.

Getting Started

What Do I Need to Know?

<https://wiki.yetirobotics.org/books/web-development/page/web-development-quickstart>

Getting Started

Discord OAuth

Creating Your Development Setup

This guide discusses setting up the Polar Edge Analytics repository for local development.

Prerequisites

This guide assumes you have the following already set up (TODO: add references on how to get these configured)

- Node.js installed; we recommend using nvm (MacOS/Linux) or fnm (Windows) to manage your Node.js version
- pnpm installed
- Docker Desktop installed

Local Setup

Setting up the Polar Edge Analytics repository requires running a local instance of the database. We currently provide a shell script located at `apps/web/lib/database/scripts/db-start.sh` to do so, with plans to run this script automatically in the future.

If the root directory, run `pnpm install` to install dependencies across the entire repository. We recommend installing the `turbo` CLI globally to make running commands easier. This can be done by running `pnpm i -g turbo`.

For developing the scouting site, only one `.env` file is currently required. For convenience, we provide a `.env.example` file within the `apps/web` directory. Copy the contents of this file into another called `.env.local`.

“ NOTE: this file is currently unavailable, it should be shortly...

Set Up Your .env.local

There are 6 Values that you need to put in your `.env.local` file.

`DATABASE_URL=postgres://postgres:postgres@localhost:5432/polar_edge`

`AUTH_DISCORD_ID= #application id in discord developer portal`

`AUTH_DISCORD_SECRET= #oauth secret in discord developer portal`

AUTH_SECRET= #generate with openssl rand -base64 32 in terminal

TBA_API_KEY= #api key from tba if using tba features

ADMIN_USERS= #your_discord_user_name_here

To get AUTH_DISCORD_ID and AUTH_DISCORD_SECRET:

1.Go to <https://discord.com/developers/applications>

2.Click on New Application and Make a New Application

3.Go to OAuth2

4.Client ID is found under the Client Information Tab, and Reset Secret for Client Secret.

To get AUTH_SECRET, go on terminal and run "openssl rand -base64 32 in terminal".
Make sure to put this in "".

To get TBA_API_KEY, go on <https://www.thebluealliance.com/account> and make a new API Key.

ADMIN_USERS is your discord username.

After .env.local is Set Up

Once dependencies are installed and the `.env.local` files are configured, the development server can be run via:

```
turbo dev # if turbo CLI installed globally
```

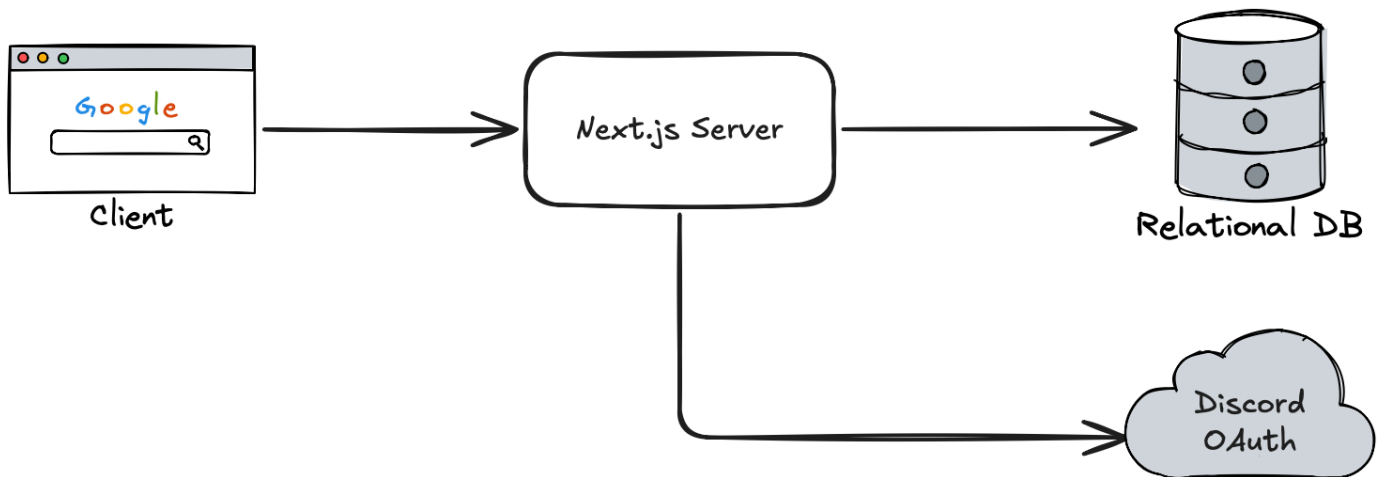
```
pnpm run dev # if turbo CLI not installed globally
```

System Architecture

NOTE: this is a living document and may not always be up to date.

Documentation created with AI support.

System Overview



The scouting site is currently comprised of the following components:

- A Next.js Server
- Relational Database (PostgreSQL)

Component Overviews

Next.js Server

Framework documentation: <https://nextjs.org/docs>

Next.js is a full-stack React framework that enables server-side rendering and seamless client-server communication. The Next.js server acts as the central hub for handling client requests. It authenticates users via Discord OAuth, interacts with the PostgreSQL database to fetch or store scouting data, and serves the frontend UI to users. This setup ensures that only authorized users can access or submit data, while also maintaining a clean separation between the client, backend logic, and persistent storage.

PostgreSQL Database

PostgreSQL Documentation: <https://www.postgresql.org>

The relational database in this architecture serves as the persistent storage layer for the scouting site. It stores all essential data such as team information, match statistics, user profiles, and scouting submissions. The Next.js server acts as an intermediary, handling data validation, access control, and business logic before reading from or writing to the database. By using a relational database, the system benefits from structured schemas, efficient querying, and strong data integrity—ensuring that all scouting information is reliable and organized.

DrizzleORM

DrizzleORM Documentation: <https://orm.drizzle.team/docs/overview>

To interact with the database and manage migrations, we utilize DrizzleORM within the Next.js application (ORM - "Object-relationship manager"). Through Drizzle, we are able to easily manage database migrations, and write type-safe queries in our Next.js server. Drizzle also provides [Drizzle Studio](#), an easy way to view and interact with your data locally.

Other Core Technologies

Docker

Docker Documentation: <https://docs.docker.com>

Docker is a platform that allows developers to package applications and their dependencies into lightweight, portable containers. These containers run consistently across different environments—whether on a developer's laptop, a test server, or in production. With Docker, you can define your app's environment in a Dockerfile, ensuring that everything from the operating system to libraries and runtime is standardized.

Our services are deployed using [Docker Compose](#). We currently publish a Docker image of the Next.js server to the [GitHub Container Registry](#). In development, we encourage using Docker to work with Postgres. This reduces "works on my machine" issues, and more closely simulates the production environment. The repository comes with scripts to properly start up a Docker container running Postgres locally.