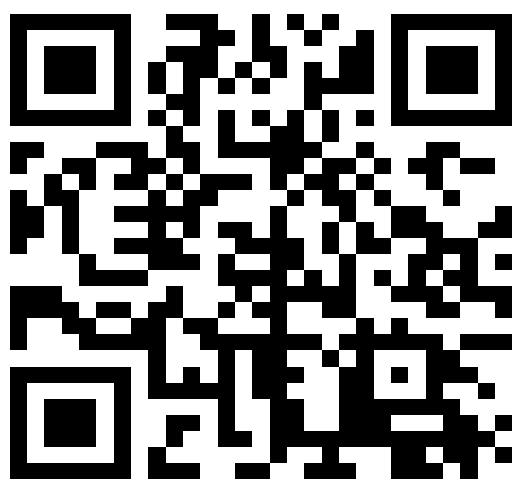


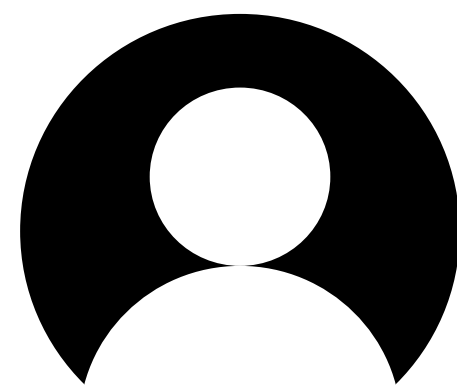
Aggre-Gator RSS

Gus Johannesen, Chris Ross,
Ellis Weaver-Kreider, Yanxi Wei



Aggre-Gator RSS is a feed aggregator that centralizes content from multiple web sources in one convenient location. Users can subscribe to their favorite websites and receive real-time updates. Built with WCU's purple and gold theme, Aggre-Gator RSS features a minimalist interface developed using TypeScript and styled with Tailwind CSS for a clean and straightforward UX design.

arrows denote TCP connections



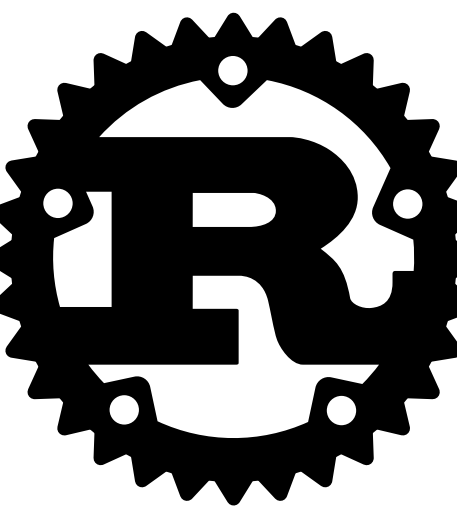
Vite

Compiles our Typescript, tree-shakes the code, and optimizes our static assets, resulting in static files we can serve with Nginx



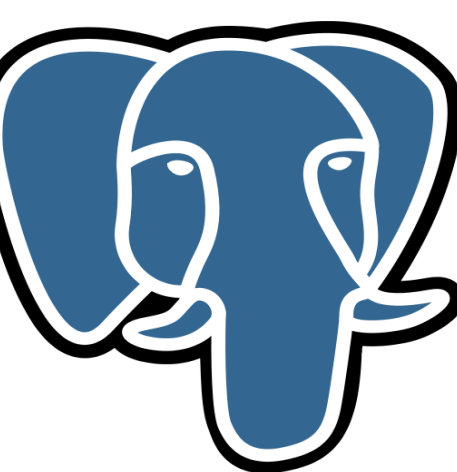
NGINX

Serves the static bundle and proxies `/api` to the `backend-api` container



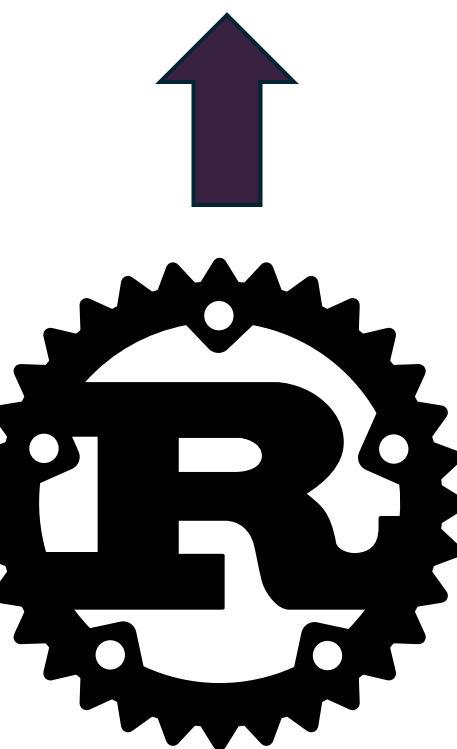
Backend API

Authenticates users and provides a REST API to access the articles and other data stored in the database



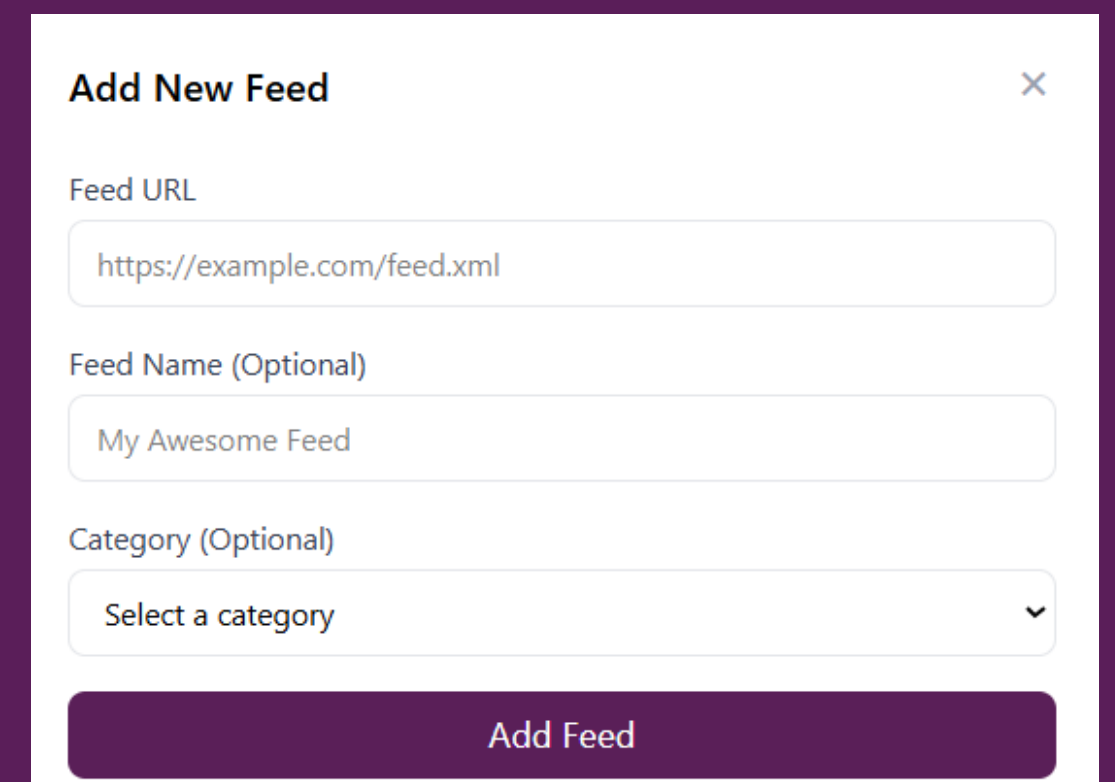
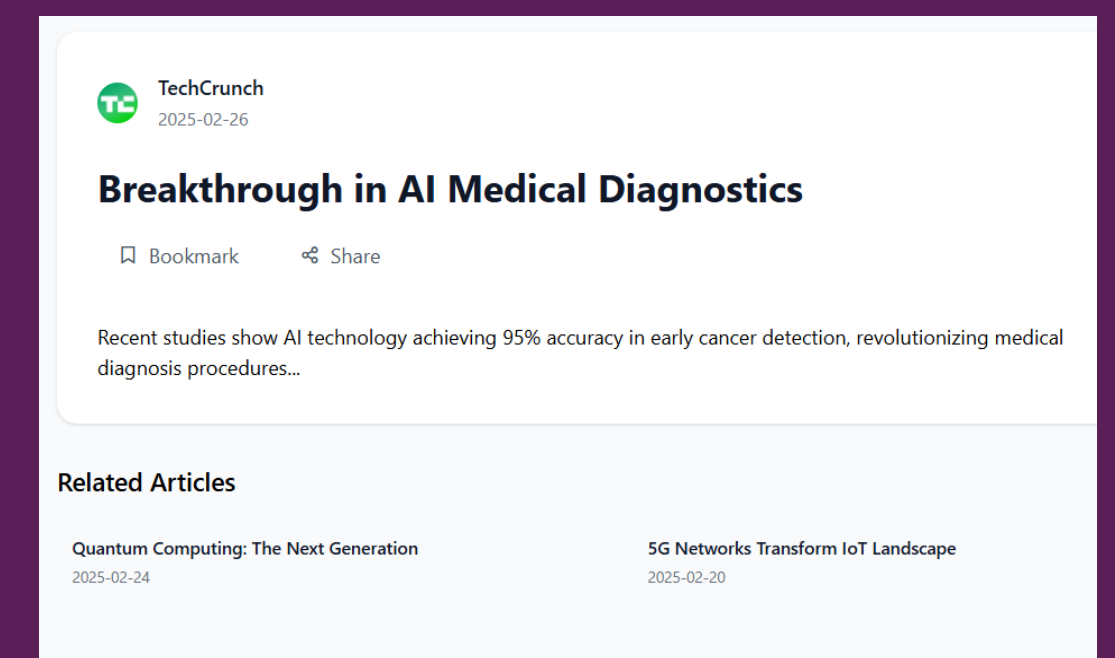
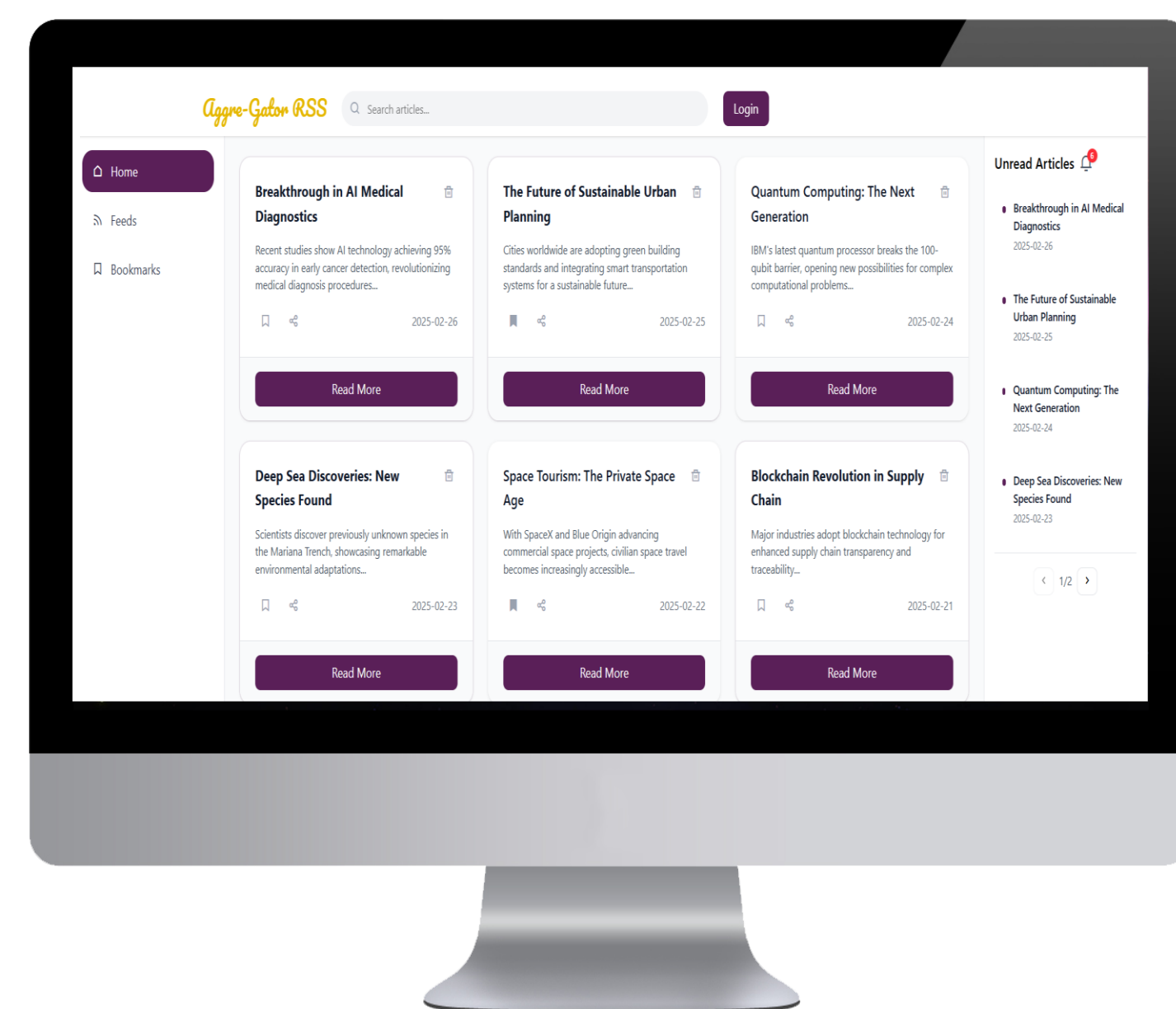
PostgreSQL

Stores articles, feeds, and extra data needed for user authentication



Feed Fetcher

Periodically fetches each feed, parses the XML, and updates the database with any new articles



Deployment

We use Nix instead of Docker to build our container images because of its focus on reproducibility and to better integrate with NixOS.

Nix allows us to define tests and checks that are run hermetically in the same way builds are, ensuring they are not affected by user-specific configuration. This means that we as developers can easily run our CI checks locally.

Nix's hermeticity allows it to cache build and check outputs when their inputs stay the same. This means that CI only evaluates checks that could have been impacted by code changes, without any manual configuration.

```
backend-api-container-stream = { /*...*/ }: dockerTools.streamLayeredImage {
  name = "backend-api";
  tag = "latest";

  config = {
    Entrypoint = [
      "${lib.getExe backend-api}"
    ];
    Env = [
      "DB_HOST=db"
      "LISTEN_PORT=80"
    ];
    ExposedPorts = {
      "80/tcp" = { };
    };
  };
};
```



The code to generate our `backend-api` container