Update architecture diagram and add additional features to read me file
Ali Al-Kubaisi authored 1 week ago

README.md 4.64 KiB

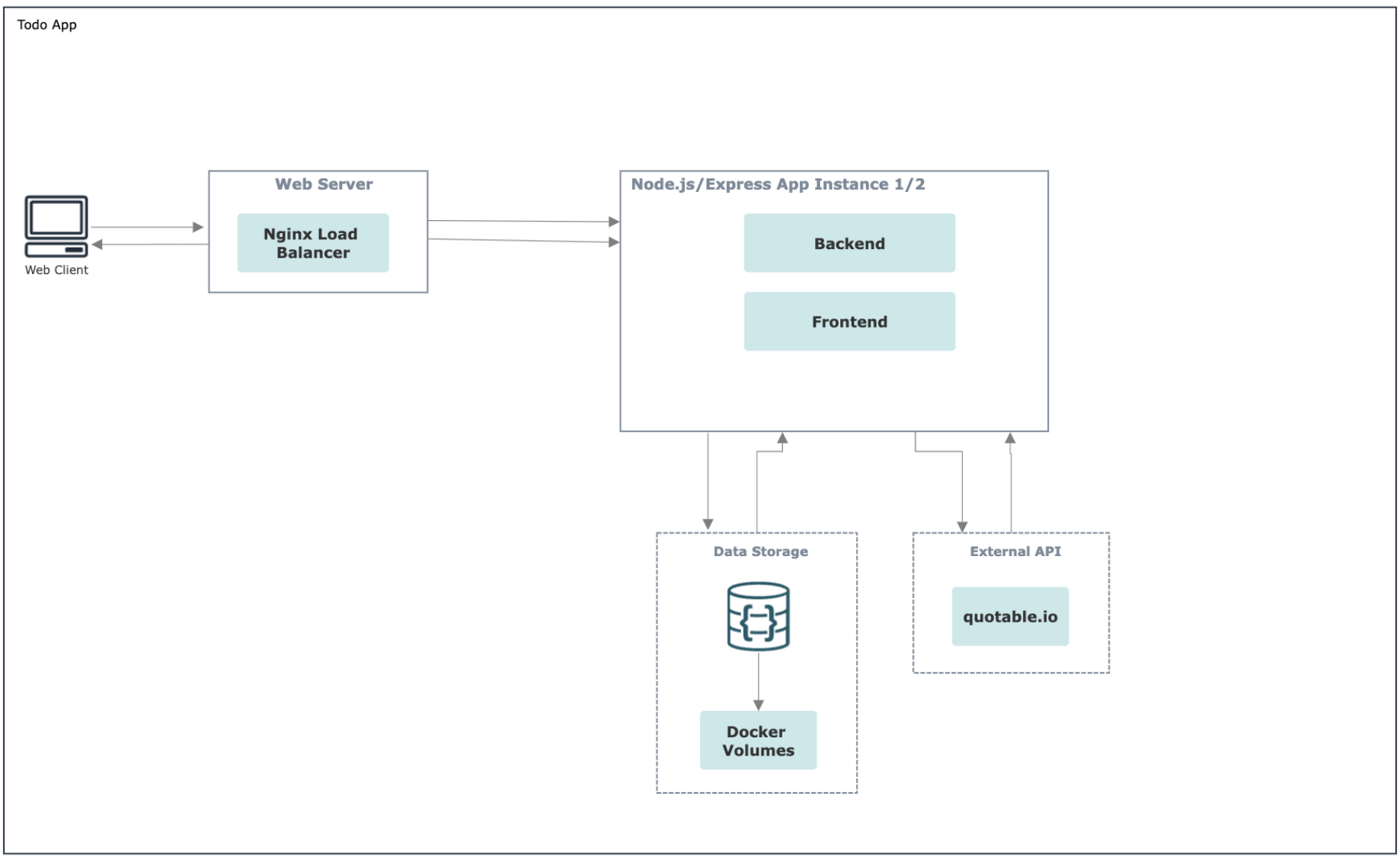
Todo Web Application

Introduction

This project is a web application designed for managing notes effectively. It offers a range of features including the ability to create, display, edit, filter, and sort notes. The application emphasizes user experience with a responsive design that adapts to different screen sizes and allows users to switch between different themes or styles for personalization.

Architecture

The application follows a client-server architecture, with the frontend and backend components decoupled for modularity and scalability. The frontend is built using Handlebars for templating and CSS for styling. The backend is powered by Node.js and Express.js, providing a robust server environment for handling requests, routing, and data management.



Features

- **Note Management**: Users can effortlessly manage their notes, with functionalities to create new notes, view them in an organized manner, edit existing notes, and delete notes no longer needed.
- **Sorting and Filtering**: Notes can be sorted in ascending or descending order based on title, date, or priority. Additionally, users can filter out completed notes to focus on tasks that require attention.
- **Style Switching**: The application supports multiple visual themes, allowing users to customize their experience according to their preference, such as switching between a light and dark mode.
- **Responsive Design**: Designed with mobile-first principles, the application ensures a seamless experience across various devices, from smartphones to desktops.
- **Accessibility**: Accessibility is a key focus, with the application being fully navigable using the keyboard, catering to users with different accessibility needs.

Additional Features

Load Balancer

To ensure high availability and effective load distribution, the application is configured with a load balancer. This setup improves response times and fault tolerance by distributing incoming network traffic across multiple server instances. Currently, the load balancer is configured to use Nginx.

Persistent Database Storage

The application uses NeDB, a lightweight JavaScript database, configured for persistent storage. This ensures that all data remains intact across application restarts and deployments. Data is stored in a file system, mounted as a Docker volume, which guarantees that updates and deletions within the application are durable and survive container restarts.

Getting Started

To get the application up and running on your local machine, follow these simple steps:

1. Clone the repository to your local machine.
2. Navigate to the project directory and run `npm install` to install the necessary dependencies.
3. Start the application by running `npm start`. The application will be available on `http://0.0.0.0:3000/` by default.

Running with Docker Compose and Load Balancer

To run the application using Docker Compose and the Nginx load balancer, follow these steps:

1. Clone the repository to your local machine.
2. Navigate to the project directory and run `docker-compose up --build` to build and start the application.
3. Access the application on `http://localhost/` in your browser.
4. To stop the application, run `docker-compose down`.

Technologies Used

- **Frontend**: Handlebars, CSS (with responsive design techniques such as Media Queries, Flexbox, and CSS Grid).
- **Backend**: Node.js for the server environment, Express.js for server-side routing, TypeScript for type-safe development and enhancing code quality.
- **Database**: NeDB, a lightweight JavaScript database, for persisting notes.
- **Templating**: Handlebars.js for dynamic content rendering on the frontend.
- **Styling**: CSS.
- **Load Balancer**: Nginx, used as a reverse proxy to distribute incoming traffic across multiple server instances effectively. Nginx enhances the application's scalability and availability by ensuring that traffic loads are evenly distributed, and by providing failover mechanisms to maintain service continuity in case one of the server instances becomes unavailable.
- **External API**: Quotable API for fetching motivational quotes, enhancing the user interface with dynamic content.

Docker Configuration

Docker is used to containerize the application, ensuring consistent environments across development, testing, and production. Docker Compose orchestrates multiple containers (application, load balancer, and persistent storage) to simplify deployment and scaling.