Builder Catalogue Project

A deep dive into the technical aspects of the project.

29 August 2023 | Asela Sandaruwan

Builder Catalogue: A Glimpse

- The Builder Catalogue is a digital service tailored for “Brick” enthusiasts.
- It bridges the gap between a user's inventory of “Brick” pieces and the vast array of “Brick” sets available.

Objective & Challenge

Objective

- Empower users to explore their existing “Brick” collections.
- Assess and determine which “Brick” sets can be constructed using pieces they already possess.

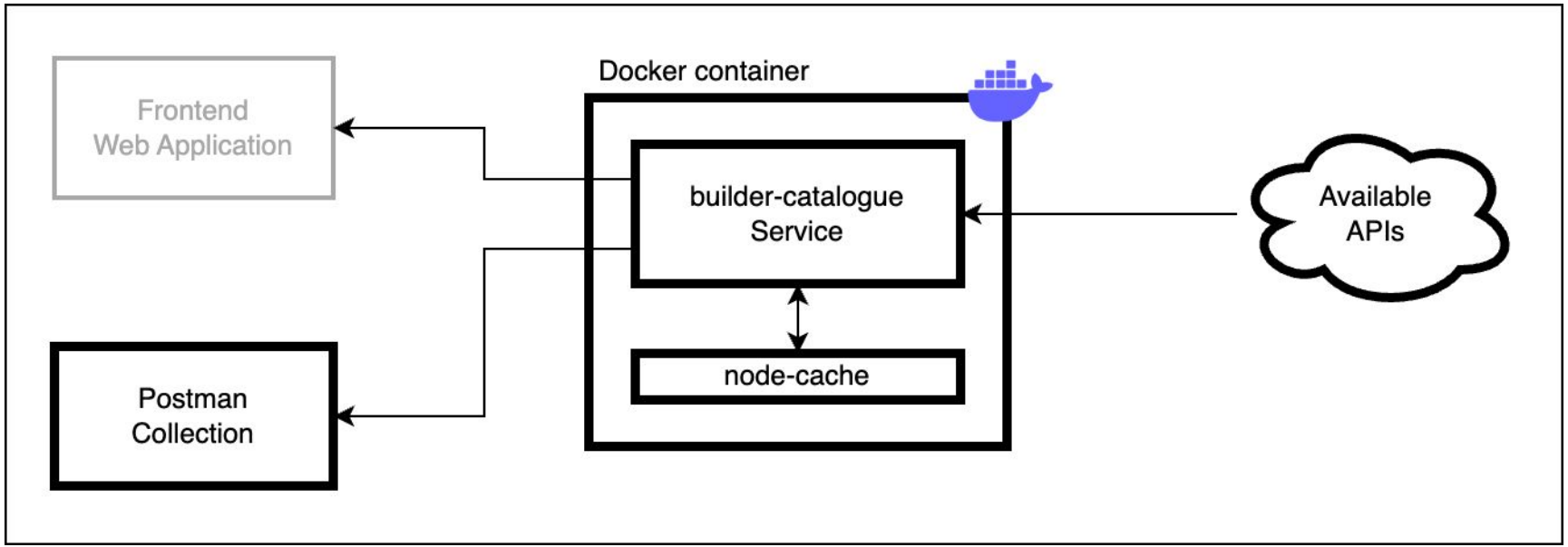
Challenge

- The core challenge was to answer: "Which sets can a user, like 'brickfan35', construct with their current inventory of pieces?"
- This required a deep dive into user inventories, set requirements, and efficient matching algorithms.

Approach

- Developed a backend service using Node.js and Express.
- Leveraged APIs to fetch user inventories and set details.
- Implemented algorithms to match user pieces with set requirements.
- Containerized the backend service using Docker.
- Established a Postman collection for seamless interaction with the builder-catalogue APIs.

Project Architecture



Developed Features

Set Building Assessment:

- Algorithms evaluate which sets a user can construct based on their current inventory.
- Provides insights into missing pieces for desired sets.
- Algorithms identify users who can contribute the missing pieces to complete a given set.

Key API Endpoints:

- /user/:username/buildable-sets: Determine buildable sets from user's inventory.
- /user/:username/set/:setname/missing-pieces: Identify missing pieces for a specific set.
- /user/:username/set/:setname/collaborators: Find potential collaborators for building a set.

Missing Features & Reasons

---

1. Frontend Web Application

Status: Initially planned, but not developed.

Reasons:

- Time Constraints: The development timeline was tight, and prioritizing the backend ensured a functional product was delivered.
- Learning Curve: Introducing a new frontend technology would require additional time for learning and implementation. Given the project's scope and timeline, it was more feasible to focus on strengthening the backend.

Missing Features & Reasons

---

2. Stretch Goals

Status: Partially completed.

Reasons for Skipping:

- Performance Optimization: Before diving into complex features, it was essential to ensure the application's core performance was optimal.
- DevOps Prioritization: Time was invested in creating Dockerfiles.

Future Enhancements & Improvements

Frontend Development:

- Originally planned to develop a frontend interface.
- Paused due to time constraints and the learning curve associated with new frontend technologies.

Hosting on GCP CloudRun:

- Evaluate CloudRun's compatibility with in-memory node-cache.
- If unsuitable, consider alternatives like Kubernetes or other serverless platforms.

CI/CD with CloudBuild:

- Automate Build, Test and Deployment of the backend and frontend services.

Infrastructure as Code:

- Implement Terraform to automate and manage infrastructure.

Authentication:

- Introduce authentication mechanisms for enhanced security, especially if a frontend is developed.

Collaborative Features:

- Enhance collaboration features, allowing users to combine inventories for joint builds.

Technical Challenges & Solutions

Data Integration:

- **Challenge**: Consistency across multiple APIs.
- **Solution**: Error handling, caching with node-cache.

Set Matching Algorithm:

- **Challenge**: Complex matching with color substitutions.
- **Solution**: Efficient data structures, task breakdown.

Docker & Deployment:

- **Challenge**: Consistency and scalability.
- **Solution**: Dockerization, future-ready for Kubernetes/CloudRun.

Frontend Development:

- **Challenge**: Time constraints.
- **Solution**: Prioritized backend, open for future frontend integrations.

DEMO

Q&A