E-COMMERCE CODE SAMPLE

Author: Diana Sormani

Introduction

This document provides an overview of an e-commerce system created as a code sample project. It's code is available at my GitHub, both backend and frontend.

This project implements a shopping cart. It provides features for:

- ❏ An administrator to create, update and delete products to be sold and upload it's information.
- ❏ A (buyer) user to navigates through those products.
- ❏ A (buyer) user to add and remove products to the shopping cart.
- ❏ A (buyer) user to analyse his/hers shopping cart.
- ❏ A (buyer) user to login and logout using his/her google account or create a new one.

Design and Architecture overview

The system was implemented with a frontend and a backend over a microservices architecture.

The backend provides the APIs required from the frontend to implement the e-commerce features. It also implements the data access layer to manipulate the information.

The frontend implements the forms,components, pages and styles necessary for the user interaction.

The system also provides **auth0** authorization, to allow the users to properly login and out of the system.

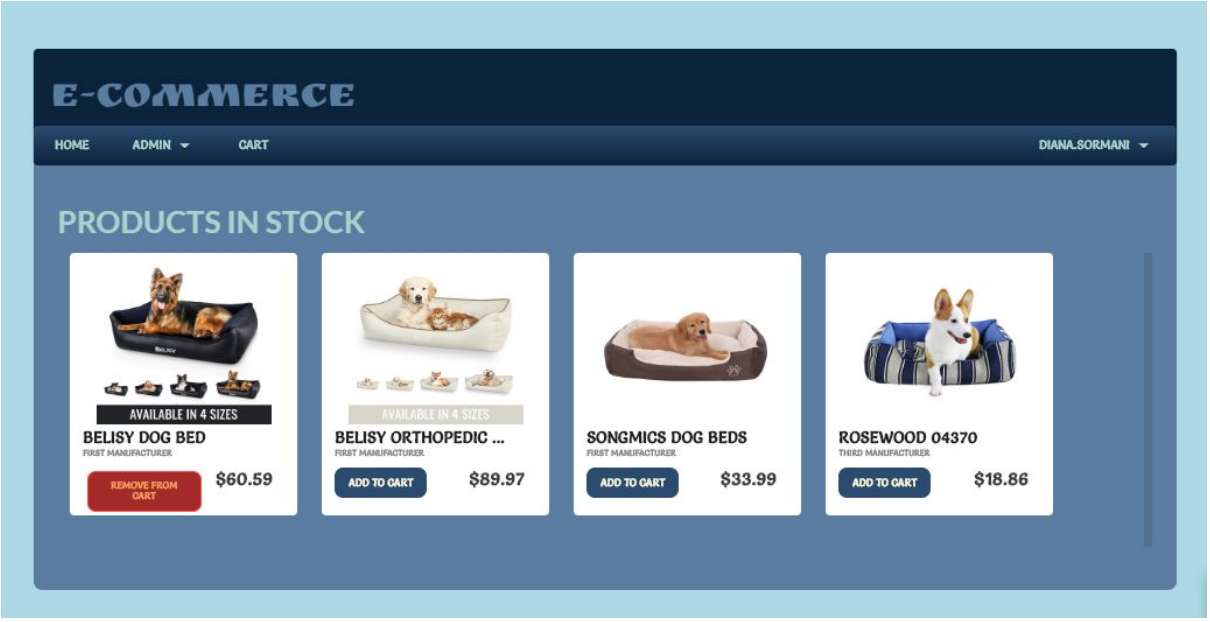
Frontend

The frontend was implemented with the following technologies and programing languages:

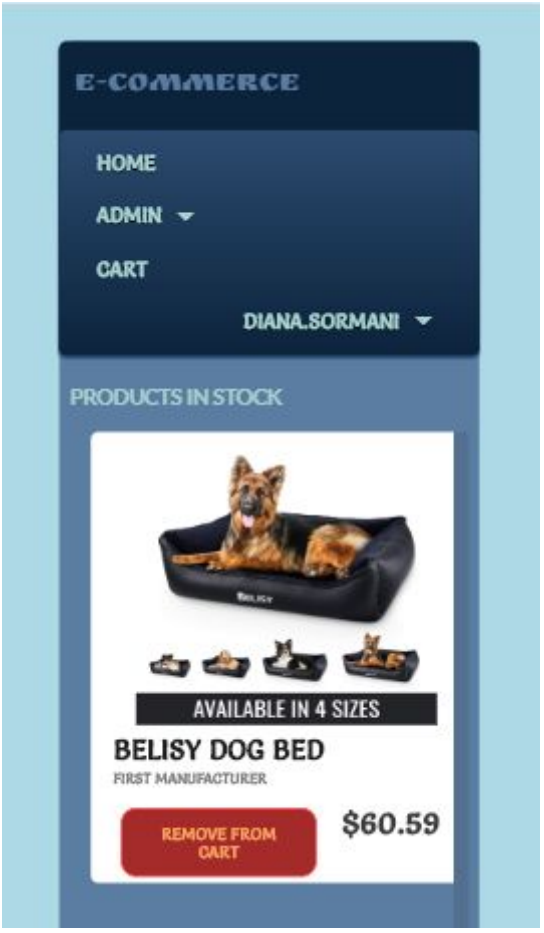
- ➔ **Vue.js**
- ➔ **Vuex**
- ➔ **CSS / HTML**
- ➔ **Auth0**
- ➔ **Git**
- ➔ **npm**

The frontend system requires user authentication, is responsive and has field to field validation. Here is a depiction of the different pages offered:

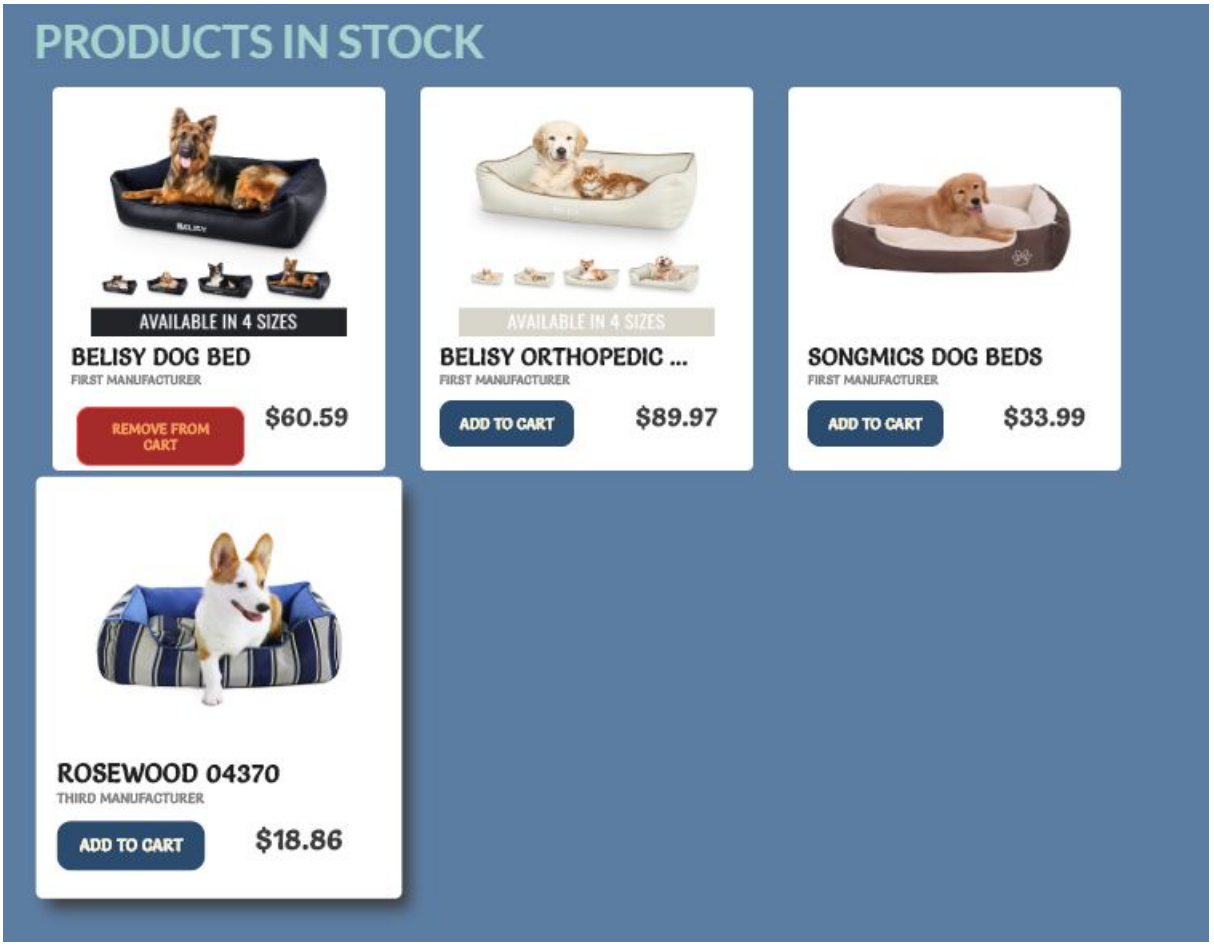
- ❏ The home page displays a list of available products. The products



The page is completely responsive: from every product component to its menus.

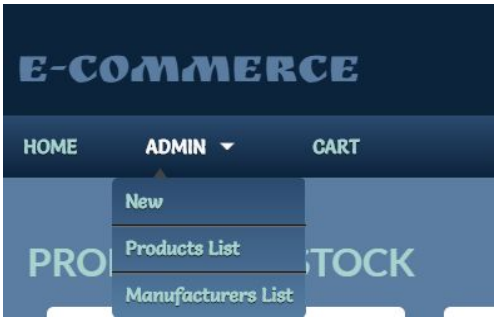


❏ This page also presents a little effect (CSS transition) to highlight the selected product:



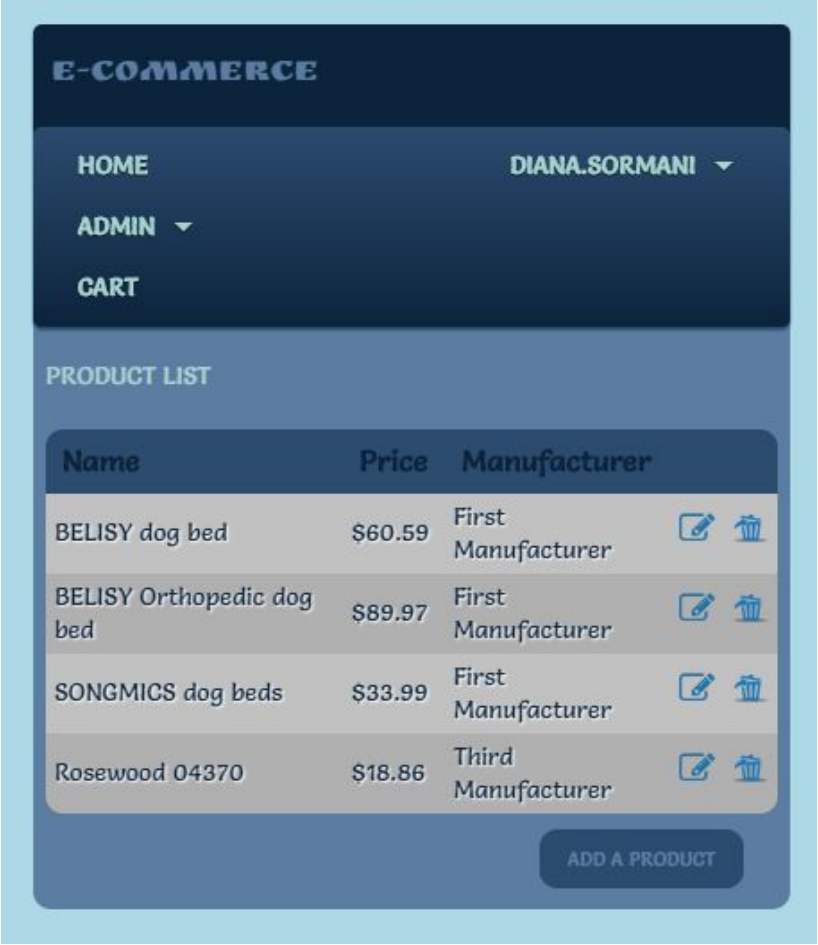
❏ The system provides a menu for the administrator. On this menu the administrator can access the following options:

- New: allows the administrator to create a new product.
- Product List: allows the administrator to manage his product list (create, update and delete products).
- List Manufacturers: allows the administrator to manage the manufacturers' list (create, update and delete manufacturers)

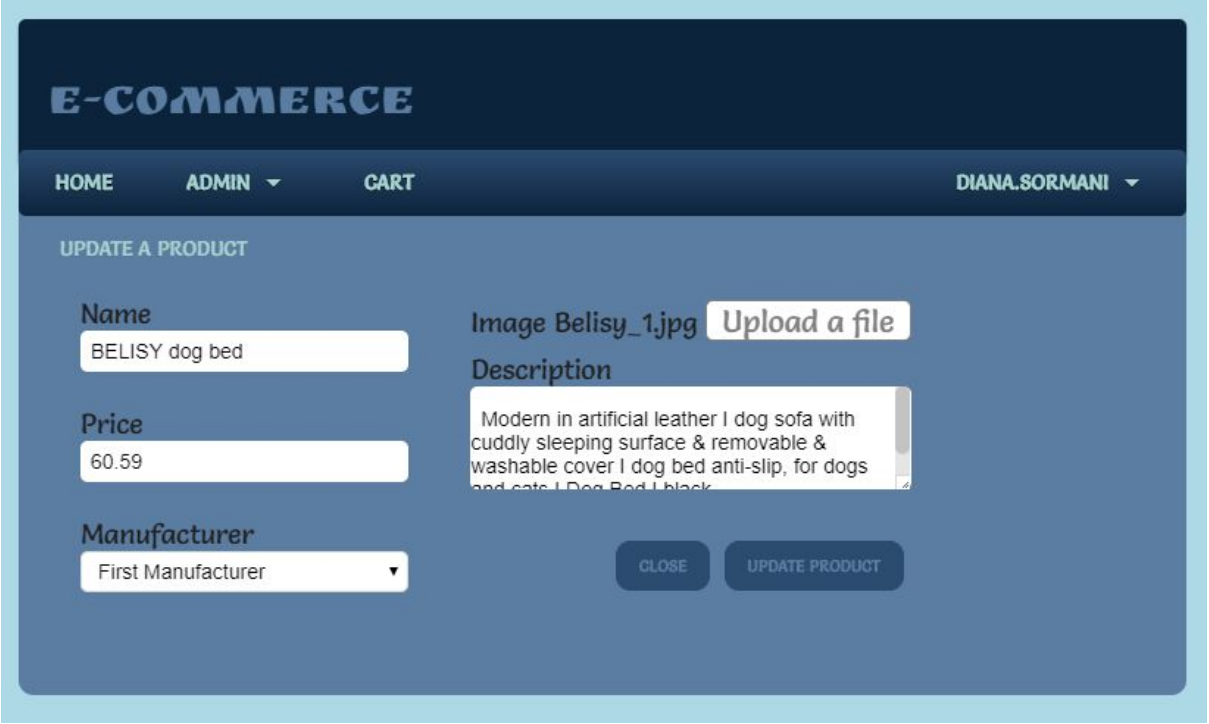


❏ The Products List page is as follows: a pretty standard responsive grid.

The same page in a smaller device:

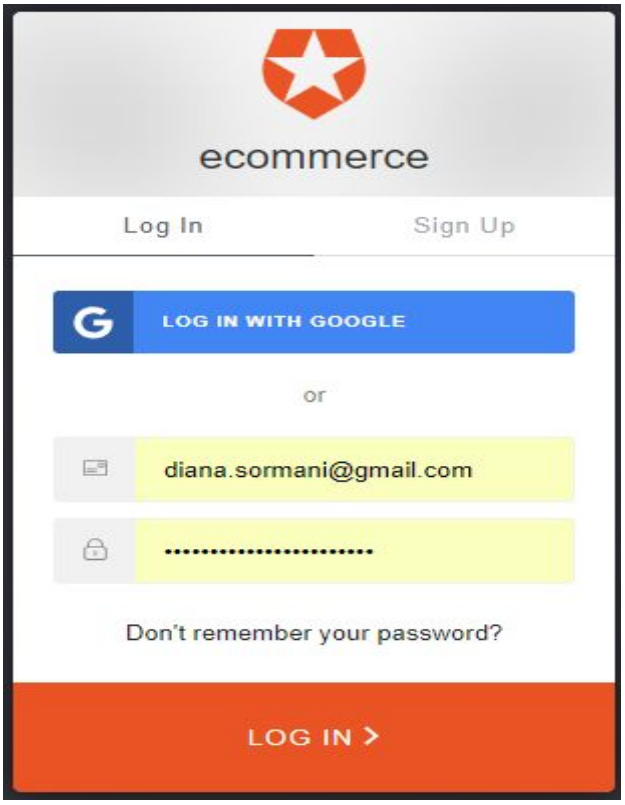


❏ The system also provides a form for product's CRUD functions:

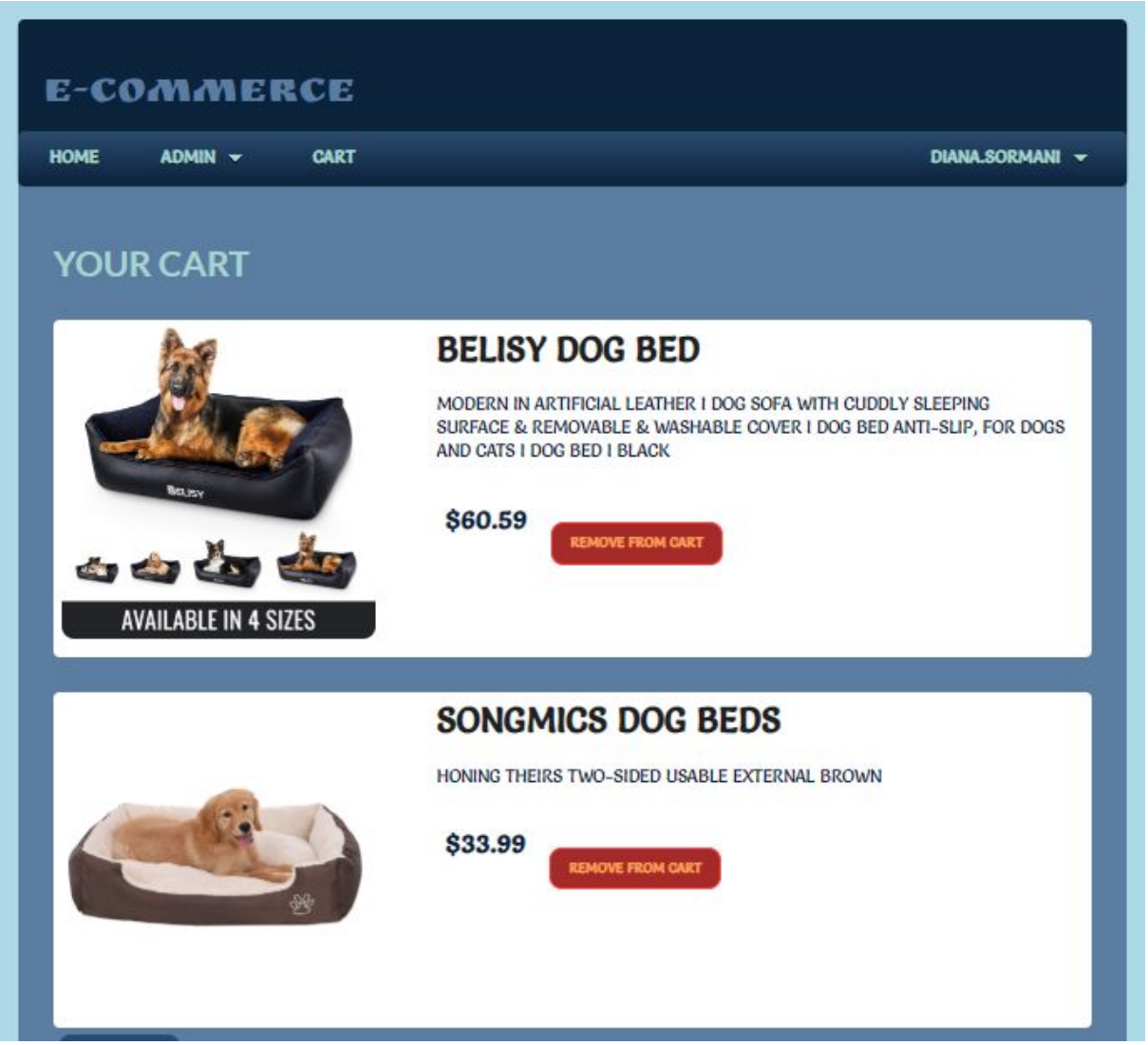


❏ Analogously, the system provides a grid for the manufacturer's information and a form to handle its CRUD functions.

❏ From the User's section of the menu, the system allows to logout. Afterwards, the system will require the user to login. The login is implemented with an auth0 component.



- ❏ Finally, the system offer a page for the user to check the selected items on his/her shopping cart:



Backend

The backend was implemented with the following technologies and programing languages:

- ➔ **Node.js**
- ➔ **Express**
- ➔ **MongoDB** (and mongoose)

It provides a layer for handling the APIs that provides the endpoints for the frontend. The api layer defines three controllers for providing the required services:

- ➔ cart.js
- ➔ manufacturer.js

- ➔ product.js

It also provides a data access layer for managing the interaction with a NoSQL database (MongoDB in this case). That layer defines the models for three data collections and also provides the proper function for accessing them:

- ➔ Manufacturers
- ➔ Products
- ➔ Carts

Installation and Execution

As mentioned above, the system's code can be obtained from my GibHub:

- ➔ Backend
- ➔ Frontend

It is necessary to clone both applications locally and run:

- **`npm install`**

To run the backend, once cloned locally,:

- cd to the application's folder
- run: **`npm start`**

To run the frontend, once cloned locally,:

- cd to the application's folder
- run: **`npm run dev`**