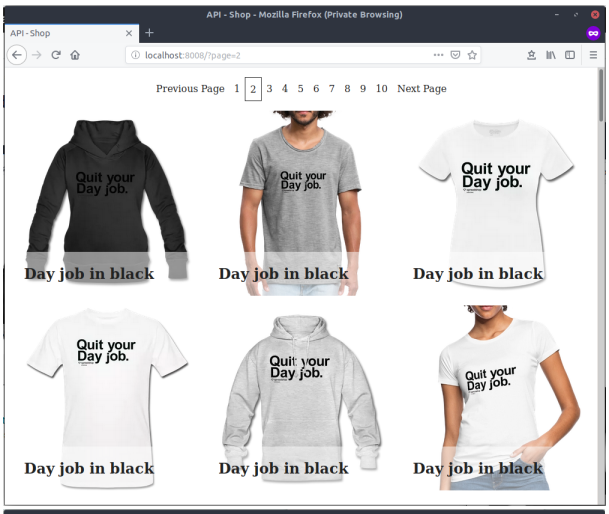
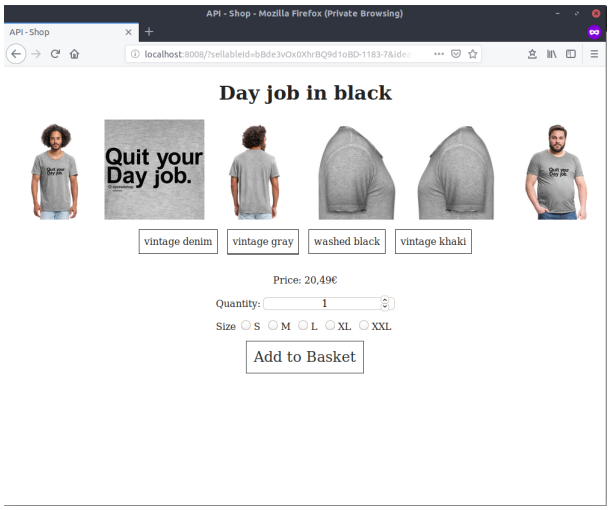
Build your own shop system

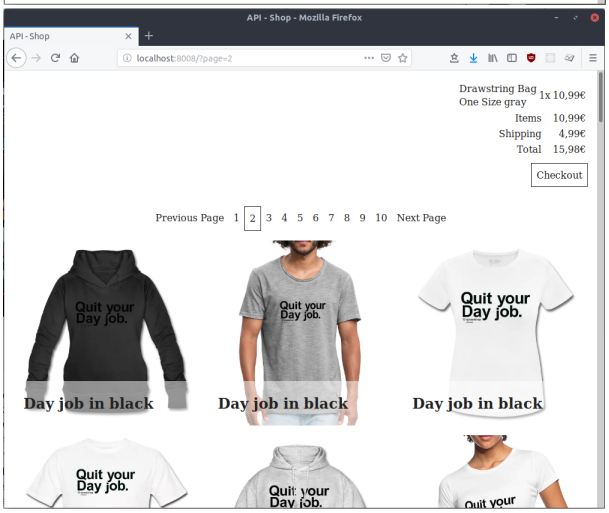
Intro

If a standalone or JavaScript Integrated Shop does not fulfill your requirements and you absolutely must have a custom shop this tutorial aims to be a starting point for your journey.

Before we start you can take a look at a **working (minimal) example** that we have built in order to demonstrate the capabilities of the Public Shop API. It shows a draft of a shop system written in php that might be helpful to understand the (list-page -> detail-page -> add-to-basket) workflow.







Prerequisites

We have to stress that you have to have a certain amount of time and technical knowledge in order to build your own shop based on the Public Shop API.

The following topics will not be covered by this tutorial and will be treated as given:

1. A Spreadshop with one or more ideas which you configured in the Partner Area
2. An API Key
3. A solid understanding of HTTP, HTML, CSS, client-side and server-side programming.

Step by step

1. Manage your assets in the Partner Area

There are no endpoints available that allow uploading or changing the content of your shop. Use the partner area to upload designs and manage your assortment. Note that not only the content you upload in the Partner Area, but also the shop settings affect the output of the sellable resources. For example, enabling model images affects the image URLs returned by the API.

2. Build a scheduled sellable data import

To use the full potential of a custom shop implementation it is recommended to import the entries of the sellable list resource into your own database every 24 hours. The example integration uses the Public Shop API as a direct data source for simplicity, but that is not the recommended approach. The import based solution yields better performance and allows you to introduce all kinds of filtering and sorting options (which are NOT built into the Public Shop API). Once you have imported this data you can build a frontend that meets your requirements.

3. Build the list page

Using the sellable data from your data base, you can now build your own list page giving an overview over the available sellables. Use the "previewImage. url" field directly in your <img> tags. You are free to build any navigational structure and filtering mechanisms you deem necessary. Depending on your needs, the ProductType Departments and Categories resource might be useful.

4. Build the detail page

When assembling a detail page on your server, request the sellable detail resource allowing you to render all available images as well as sizes. If you want to display things like name and description of the product type, appearance names and colors, etc., the ProductType resource is useful. The response of that resource is slow and rarely changes, so it is a good candidate for caching. In a somewhat generic implementation, the Price Formatting Guide might be helpful.

5. Build a basket

Once a user clicks the add-to-basket button, use the "Create a basket" resource to create a Spreadshirt basket with the selected sellable in it. Associate the returned basket id with the customer's session. You can now use the fields from the basket payload to assemble a drop-down UI component or something similar. Use the "Update a basket" resource for subsequent customer interactions.

6. Redirect to the Spreadshirt checkout

Finally, you can redirect the customer to the Spreadshirt checkout using the "shop checkout" link from the basket response. We will take over from there.