Guide to Creating a Live Sales Counter Using a Raspberry Pi

Ivan Santagati

May 2024

1 Introduction

This guide will help you create a live counter that displays how many products have been sold in your Shopify store. We will use a Raspberry Pi to create a server that will fetch data from the Shopify API and display it on your website. This guide is detailed to help even those with no programming experience.

2 Setting Up Your Shopify API Access

To fetch sales data, you need to create a private app in Shopify to get API credentials.

2.1 Creating a Private App in Shopify

1. Log in to your Shopify admin panel.
2. Go to **Apps** in the left-hand menu.
3. Scroll down and click **Develop apps for your store**.
4. Click **Create an app**.
5. Enter a name for your app (e.g., "Sales Counter App") and your email address.
6. Click **Create app**.

2.2 Setting API Permissions

1. In your newly created app, go to the **Configuration** tab.
2. Click **Configure Admin API scopes**.
3. Check the box for **Read access** under **Orders**.
4. Click **Save**.

2.3 Obtaining API Credentials

1. In the app settings, go to the **API credentials** tab.
2. You will see your **API key** and **API secret key**. Note these down.
3. Click **Reveal token once** to get your **Access token**. Note this down as well.

3 Preparing the Raspberry Pi

First, we need to prepare the Raspberry Pi to act as a server.

3.1 Installing Node.js and npm

Connect to your Raspberry Pi via terminal or SSH and install Node.js and npm:

```
sudo apt update
sudo apt install nodejs npm -y
```

3.2 Initializing a New Node.js Project

Create a new folder for your project and initialize a Node.js project:

```
mkdir sales-counter
cd sales-counter
npm init -y
```

3.3 Installing Express

Install Express, a popular framework for Node.js:

```
npm install express
```

4 Creating a Server to Fetch Data from Shopify API

Now we will create a server that will fetch sales data from the Shopify API.

4.1 Creating `server.js` File

In your project folder, create a file named `server.js` and add the following code:

```
const express = require('express');
const fetch = require('node-fetch');
const app = express();
const port = process.env.PORT || 3000;

const SHOPIFY_API_KEY = 'your_api_key';
const SHOPIFY_PASSWORD = 'your_password';
const SHOP_NAME = 'your_shop_name';

app.get('/sales-count', async (req, res) => {
    try {
        const response = await fetch(`https://${SHOPIFY_API_KEY}:${SHOPIFY_PASSWORD}@${SHOP_NAME}.myshopify.com/admin/api/2023-04/orders.json?status=any`, {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json'
            }
        });
        const data = await response.json();
        const totalSoldProducts = data.orders.reduce((acc, order) => acc + order.line_items.reduce((sum, item) => sum + item.quantity, 0), 0);
        res.json({ sales_count: totalSoldProducts });
    } catch (error) {
        console.error('Error fetching sales data:', error);
        res.status(500).json({ error: 'Error fetching sales data' });
    }
});

app.listen(port, () => {
    console.log(`Server running on port ${port}`);
});
```

Make sure to replace `your_api_key`, `your_password`, and `your_shop_name` with your Shopify API credentials and store name.

4.2 Starting the Server

Start the server with the following command:

```
node server.js
```

5 Adding the Counter to Shopify Theme

Now we will add the code to your Shopify theme to display the counter.

5.1 Adding the HTML Element

Go to your Shopify admin, then to **Online Store ¿ Themes ¿ Actions ¿ Edit code**. Find the `theme.liquid` file and add the following HTML element where you want the counter to appear:

```html
<div id="sales-counter">0</div>
```

5.2 Adding the JavaScript Code

Create a new JavaScript file (e.g., `custom.js`) in your theme and add the following code:

```javascript
async function updateSalesCounter() {
    try {
        const response = await fetch('https://your-server-url/sales-count'); // Replace with your server URL
        const data = await response.json();
        const salesCount = data.sales_count;

        document.getElementById('sales-counter').innerText = salesCount;
    } catch (error) {
        console.error('Error fetching sales data:', error);
    }
}

// Update the counter every 10 seconds
setInterval(updateSalesCounter, 10000);

// Update the counter when the page loads
updateSalesCounter();
```

Replace `https://your-server-url/sales-count` with your server's URL.

5.3 Including the JavaScript File in the Theme

Find the `theme.liquid` file and add the following line before the closing `</body>` tag:

```html
<script src="{{ 'custom.js' | asset_url }}"></script>
```

6 Testing

Check if the server is working:

- Open your browser and go to your server's URL (`https://your-server-url/sales-count`). You should see a JSON response with the number of products sold.

6.1 Example of a JSON Response

Here is an example of what the JSON response might look like:

```json
{
    "sales_count": 25
}
```

Check if the counter is working on your website:

- Open your online store and see if the counter updates correctly.

7 Conclusion

By following these steps, you have successfully set up a live sales counter in your Shopify store using a Raspberry Pi as a server. Make sure your server is secure and does not slow down your website.