Project Title:

Automated Order-to-Shipping Workflow (WooCommerce → GLS)

Project Description:

This project is a browser-based automation system designed to eliminate manual data entry when creating shipping orders through the GLS web interface. The core problem addressed is the lack of a public API for GLS shipment creation, which forces users to manually input customer data for every order. This process is time-consuming, repetitive, and prone to human error.

To solve this, I developed a lightweight automation pipeline that connects WooCommerce order data with the GLS shipping form without relying on any external APIs, backend services, or automation frameworks such as Selenium.

The system works as follows: when viewing an order inside WooCommerce, a custom button labeled “GLS AUTO” appears. This button is generated using a PHP snippet hooked into the WooCommerce admin interface. When clicked, it constructs a URL containing all relevant order data (name, address, city, ZIP code, phone, and email) and sends it to a Cloudflare Worker.

The Cloudflare Worker acts as a data processing layer. It normalizes and prepares the data before redirecting the user to the official GLS shipping page with all parameters embedded in the URL. The Worker performs several key transformations, including splitting full names into first and last names, extracting street names and house numbers from a single address field, cleaning phone numbers into a valid format, and generating fallback names from email addresses if necessary.

The final step is handled by a custom Chrome extension (located in the “gls-extension” folder), which reads the URL parameters and automatically fills in the GLS form fields. Since the GLS form is built using Vue.js, direct DOM value assignment is not sufficient. To ensure proper reactivity, the extension uses native input setters and dispatches input and change events to simulate real user interaction. It also includes logic to wait for dynamically loaded elements and correctly target multiple form instances (sender vs recipient).

This approach effectively creates a “no-API automation layer” that replicates user behavior in a reliable and fast way. The result is a significant improvement in efficiency: the time required to create a shipment is reduced from approximately two minutes to just a few seconds, with near-zero input errors.

Key Features:

- Fully automated form filling on a third-party platform without API access
- Seamless integration with WooCommerce order data
- Intelligent data normalization (name parsing, address splitting, phone formatting)
- Vue.js compatibility via event simulation
- Zero external infrastructure beyond a lightweight Cloudflare Worker
- Fast and reliable execution entirely within the browser

Technologies Used:

JavaScript (DOM manipulation and event handling), Cloudflare Workers, Chrome Extension APIs, PHP (WooCommerce hooks), URL parameter data transport, frontend reverse engineering (Vue.js behavior handling)

Cloudflare Worker Code:

```
export default {
  async fetch(request) {

    const url = new URL(request.url);

    const name = url.searchParams.get("name") || "";
    const address = url.searchParams.get("address") || "";
    const city = url.searchParams.get("city") || "";
    const zip = url.searchParams.get("zip") || "";
    const email = url.searchParams.get("email") || "";

    const phone = (url.searchParams.get("phone") || "")
      .replace(/\D/g, '')
      .replace(/^385/, '')
      .slice(-9);

    // IME / PREZIME
    let parts = name.trim().split(/\s+/);
    let first = parts[0] || "";
```

```
let last = parts.slice(1).join(" ") || "";

if (!first || !last) {
  const emailName = email.split("@")[0];

  const emailParts = emailName
    .replace(/[0-9]/g, '')
    .split(/[._-]+/);

  if (!first) first = capitalize(emailParts[0] || "");
  if (!last) last = capitalize(emailParts[1] || "");
}

function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}

// ADRESA SPLIT
function splitAddress(full) {
  if (!full) return { street: "", number: "" };

  const match = full.match(/^(.*?)[\s,]+(\d+[a-zA-Z]?)/);

  if (match) {
    return {
      street: match[1].trim(),
      number: match[2].trim()
    };
  }

  return {
    street: full,
    number: ""
  };
}

const { street, number } = splitAddress(address);

// GLS URL
const glsUrl = new URL("https://www.paket.hr/posalji-paket-pl");

// KLJUČNO (ovo ti je falilo)
glsUrl.searchParams.set("name", name);

// postojeće
glsUrl.searchParams.set("first", first);
glsUrl.searchParams.set("last", last);
glsUrl.searchParams.set("address", address);
```

```
    // novo
    glsUrl.searchParams.set("street", street);
    glsUrl.searchParams.set("house", number);

    glsUrl.searchParams.set("city", city);
    glsUrl.searchParams.set("zip", zip);
    glsUrl.searchParams.set("phone", phone);
    glsUrl.searchParams.set("email", email);

    return Response.redirect(glsUrl.toString(), 302);
  }
};
```

WooCommerce Integration Snippet (PHP):

```
add_action('woocommerce_order_item_add_action_buttons', 'gls_worker_button');

function gls_worker_button($order) {

  if (!$order) return;

  $first = $order->get_shipping_first_name() ?: $order->get_billing_first_name();
  $last  = $order->get_shipping_last_name()  ?: $order->get_billing_last_name();

  $url = "https://glsautomate.opgbranko1.workers.dev?" . http_build_query([
    'name'    => trim($first . ' ' . $last),
    'address' => $order->get_shipping_address_1(),
    'city'    => $order->get_shipping_city(),
    'zip'     => $order->get_shipping_postcode(),
    'phone'   => $order->get_billing_phone(),
```

```
    'email'   => $order->get_billing_email()
  ]);


  echo '<a href="'.$url.'" target="_blank" class="button button-primary" style="margin-left:10px;">
    GLS AUTO
  </a>';
}
```

Extension Location:

The Chrome extension responsible for autofilling the GLS form is located in the folder: gls-extension