MAGENTO 2 CASE STUDY

A stunning makeover: how PWA transcended website performance for a beauty retail leader

BROCARD®

PARFUMS

ABOUT THE CLIENT

Brocard reigns as the ultimate destination for luxury cosmetics and perfumery in Ukraine. It stands out for its exquisite collections of skincare and beauty products and accessories: Brocard is the biggest retail chain that sells worldwide famous brands such as Chanel, Dior, Lancôme, Guerlain, Burberry, Yves Saint Laurent, Givenchy, Dolce&Gabbana, Giorgio Armani, Gucci, etc.

Brand: Brocard

Industry: Beauty & Cosmetics

Region: Ukraine

Website: www.brocard.ua

CLIENT IN FIGURES

1997

Date of establishment

70

Physical stores in 20 cities

60

Niche brands in the product line

CHALLENGE

We had been working with the client for quite a long time, diligently involved in the development, support, and maintenance of their store. However, as an increasing number of clients flocked to their website, accompanied by a growing assortment of products, we encountered a stumbling block in the form of performance issues. Notably, the website experienced slow page loading, particularly when loading the product catalog on the product listing page.

PageSpeed Insights indicated that the website's loading time fell within the unsatisfactory yellow zone, which is particularly concerning for the store because it signifies slower performance and potential negative impacts on user experience and conversion rates.

Yet, Brocard wanted to put a strong emphasis on Search Engine Optimization, and performance issues were a solid blocker for it: search engines consider page load speed as an important ranking factor. A slow-loading website can result in higher bounce rates and lower organic search rankings.

On top of that, Brocard had low PageSpeed Insights metrics on mobile devices. And search engines prioritize mobile-friendly websites. Thus, Brocard was falling behind its competitors in the very rivalrous niche of the eCommerce beauty, and this was a crucial point to optimizing performance.

DISCOVERY STAGE

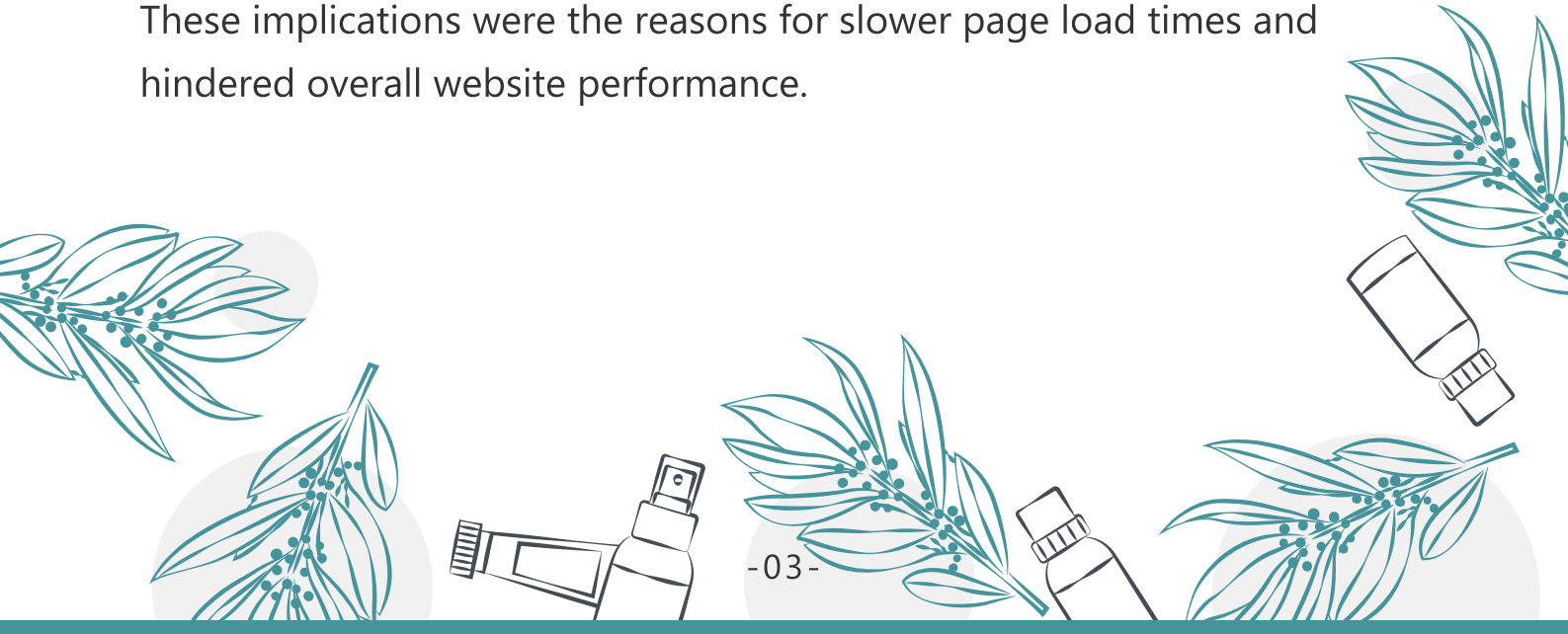
During our analysis of the website code to identify areas for improvement, we observed that the server responded quickly to requests from the front-end. Yet, pages were loading extremely slowly.

The online store's front-end code had already been extensively customized. All possible optimizations had already been implemented, such as improved **Time to First Byte***, applied advanced **JavaScript bundling**** and optimized CSS.

*Time to First Byte (TTFB) measures the duration between a browser's request to a server and the moment it receives the first byte of the server's response.

**JavaScript bundling is an optimization technique you can use to reduce the number of server requests for JavaScript files.

Additionally, we noticed that the excessive use of HTML tags in Magento had negative effects on website performance. This included larger page sizes, a higher number of network requests, and complex code structures. These implications were the reasons for slower page load times and hindered overall website performance.



PLANNING STAGE

Since we have already done everything for Brocard's website optimization, at this point we could only have two ways for resolving performance issues: create a new theme or develop a **progressive web app***.

*A progressive web app (PWA) is a type of web application that leverages modern web technologies to provide a user experience similar to that of native mobile apps. PWAs are designed to be reliable, fast, and engaging, allowing users to access them through a web browser on various devices, including desktops, smartphones, and tablets.

DEVELOPMENT STAGE

TEAM	**1** Project Manager	**3** Back-end developers
	3 Tech Leads	**3** QA Engineers
	4 Front-end developers	

Initially, we selected **Vue Storefront*** as our preferred PWA framework for the project. However, upon careful analysis, we realized that the **Elasticsearch**** module provided by Vue Storefront would conflict with Magento's Elasticsearch module. The REST API communication between Vue's Elasticsearch and Magento was not ideal for our situation due to the presence of significant custom code in the Brocard website. This would add unnecessary complexity to Vue Storefront.

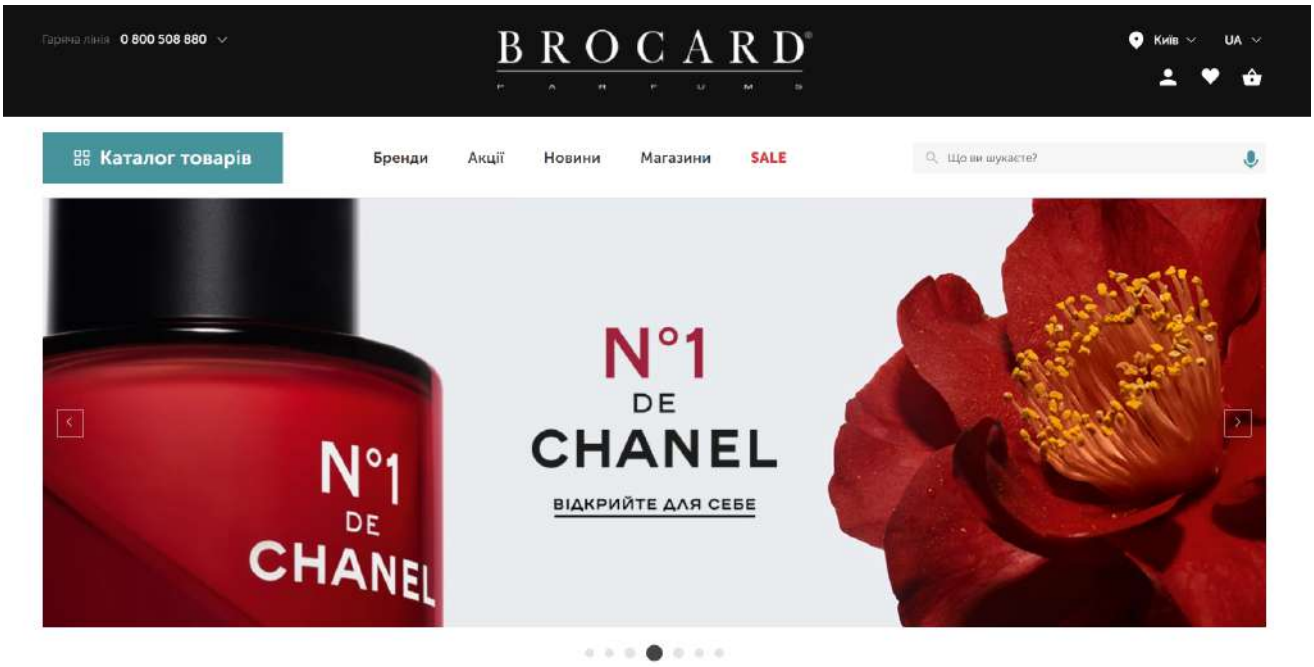
*Vue Storefront is an open-source PWA framework specifically designed for eCommerce applications. It is built using Vue.js, a popular JavaScript framework. Vue Storefront aims to provide a performant and customizable PWA frontend for eCommerce platforms.

**Elasticsearch is a powerful open-source search and analytics engine built on top of the Apache Lucene library. It provides a distributed, scalable, and real-time search solution with advanced search capabilities, including full-text search, faceted navigation, and filtering.

We then turned our attention to **GraphQL*** as a solution. We realized that this query language enables direct communication with Magento. Plus, it improves with every Magento update. GraphQL is framework-agnostic and can be utilized with different backend technologies. It empowers us to specify data requirements through a query syntax, and in response, the server delivers the requested data in a structured format.

> ***GraphQL** is an open-source query language and runtime for APIs (Application Programming Interfaces). It provides a more efficient and flexible alternative to traditional RESTful APIs for fetching and manipulating data.

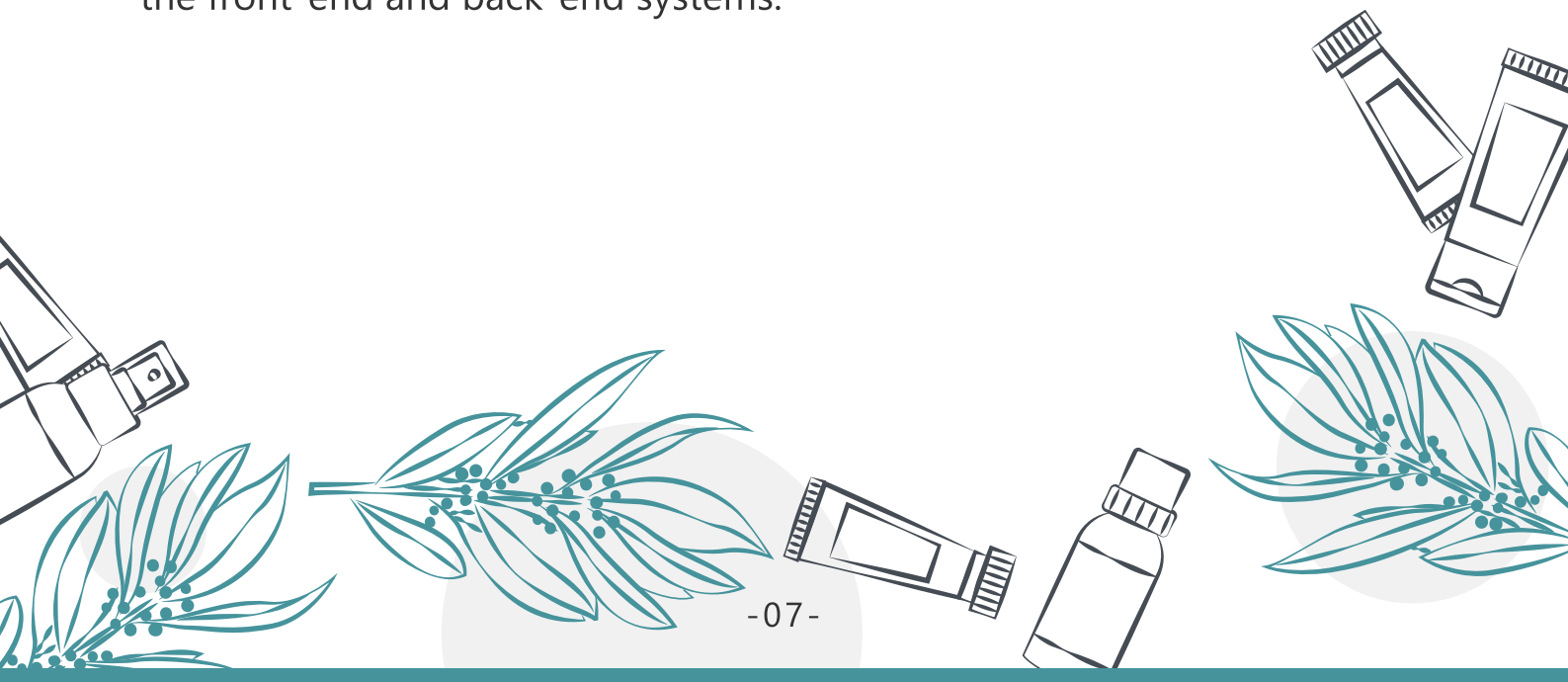
Additionally, we recognized that working with GraphQL would require significantly less time compared to Vue Storefront, as we would not need to write REST API code or handle Elasticsearch. Instead, we could simply write queries or mutations using GraphQL.



Plus, we needed **Server-side rendering (SSR)***. It allows making all HTML content visible to Google crawlers is essential for SEO. It allows search engines to properly index and understand your website's content, ensuring keyword relevance, improved visibility in search engine result pages, and enhanced user experience. By ensuring visibility, you enable search engines to extract valuable information, display rich snippets, and optimize for mobile indexing, ultimately increasing organic traffic and improving your website's SEO performance.

*Server-side rendering (SSR) is a technique used in web development to render web pages on the server and send the fully rendered HTML to the client's browser. In SSR, the server processes the request, fetches the required data, and generates the HTML content with the data included. The complete HTML is then sent to the client, where it can be displayed immediately.

We performed additional configurations to enable Magento and existing extensions to work seamlessly with GraphQL. We implemented resolvers, wrote mutations, and queries to establish direct communication between the front-end and back-end systems.



RESULTS

We received a fully customized PWA that significantly improved web page loading speed, as evidenced by optimal results in Google PageSpeed Insights. Additionally, we implemented custom HTML code to ensure all web page content is visible to Google, leading to a substantial enhancement in SEO. Moreover, this custom solution provides unlimited potential for enhancing the performance of Brocard's online store in the future.

LET'S HAVE A CHAT

Choose the most convenient way of communication for you — write an email or contact us in one of the messengers.

We'll discuss your project — provide individual calculations and offer our suggestions on how to upgrade your business.

Email:

sales@magecom.net

Phone:

+44 7491 43 5563

Office:

United Kingdom

102 Wornington Road

London W10 5QP

Messenger:

SKYPE

FB MESSENGER

WHATSAPP

TELEGRAM

magecom

Your Global Ecommerce Partner