MAGE ZINE

ISSUE 2/2019 Magento community journal www.magezine.co

PWA is (not) rocket science

Is that so? Check it out!

Who will win the PWA battle?

PWA experts & Magento developers:

Shane Osbourne

Ilja Lapkovskis

Jisse Reitsma

Miguel Balparda

Ignacio Riesco

Yevhen Sentiabov

Alex Paliarush

Oleksandr Lyzun

M1 end of life

How will Magento extensions fit into PWA?

PWA & Magento – The 3 key considerations for developers

Why are PWAs taking so long to enter the Magento mainstream


Magento®
An Adobe Company
www.magento.com

Man on the Moon

MAGE ZINE

On July 21, 1969 at 2:56 UTC Neil Armstrong set foot on the Moon as the first man ever. Millions of people watched this moment in front of television screens, believing that they were witnesses of a great breakthrough. At that brief time cosmos seemed to be at their fingertips. After the completion of the Apollo 11 mission, no other space exploration aroused so much interest, even though the human foot has touched the silver surface of the Moon more than once in the following years.

Łukasz Bajsarowicz

Magezine Evangelist
Magento 2 Backend Developer at Strix
Magento Open Source Maintainer

MAGEZINE STAFF

MAGEZINE EVANGELIST
Łukasz Bajsarowicz

CONTENT EDITORS
Katarzyna Gądek
Paulina Jóźwik

MANAGING EDITOR AND PRODUCER
Karolina Gajzler-Polak

DTP DESIGNER
Michał Hojnacki

COVER ILLUSTRATION
metaborg studio

PHOTOS AND ILLUSTRATIONS
stock.adobe.com

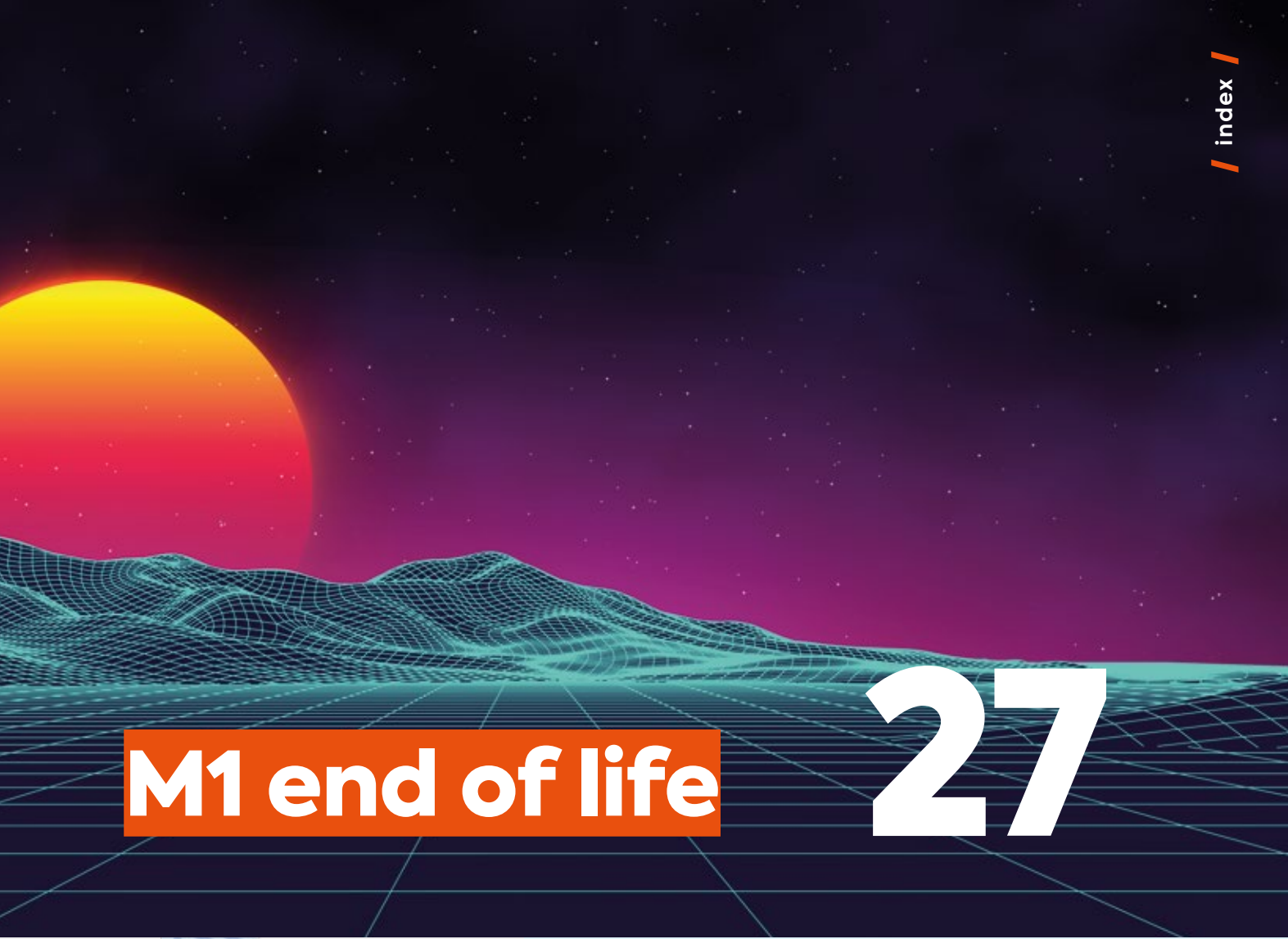
WWW.MAGEZINE.CO

Proudly made by Strix

50 years have passed since that day and the world looks completely different now. Today's smartphones have incomparably more computing power than computers of that time. Technology is evolving at a pace that can make even people born in digital times dizzy. We can say that we are a society of change, and like no generation before, we are able to adapt to it.

Of course, these changes do not bypass e-commerce (how could they?). Changes that were announced as turning points not long ago are now commonplace (omnichannel, mobile shopping, AI in commerce – you name it). The question is: what will revolutionize the market today? Will it be PWA? Let's wait and see. But we know for sure that more and more stores are taking the opportunities offered by Progressive Web Apps and providing their customers with an even better shopping experience. That's why we decided to make PWA our cover story.

We've prepared an article that introduces you to this topic, we also have an overview of the most popular e-commerce solutions. But the best part is that we've gathered authors who know everything about PWA! Ilja Lapkovskis, CTO of Scandiweb writes an article *Why are PWAs taking so long to enter the Magento mainstream?*, Jisse Reitsma about *How Will Magento Extensions Fit into PWA*, Shane Osbourne about *PWA & Magento: The 3 Key Considerations for developers*. Of course, these are not all the texts you will find in this issue. Anyway, check it out for yourself!



MERCHANTS

The latest version of Magento Commerce and Magento Open Source 2.3.2 are now available

The latest release of Magento software includes several security enhancements along with substantial performance improvements. Highlights of this release include:

- **Security:** Multiple enhancements were identified by leveraging Adobe's sophisticated security tools and the large reach of the Adobe Hacker One bug bounty program.
- **Performance:** Significant performance upgrades include 20% improvement to storefront page-load times, product images loading concurrently with other page content, and up to 90% improvement in category browsing for merchants with large catalogs.
- **Productivity:** Several actions are now performed as asynchronous background processes, allowing administrators to continue working while tasks are being processed in the background.
- **Quality:** Over 130 product quality enhancements across many critical areas of the platform.

Source: https://community.magento.com/t5/News-Announcements/Now-Available-Security-and-Performance-Updates-in-The-New/m-p/135197#M238

MERCHANTS

Adobe (Magento) is named a leader in Gartner's 2019 Magic Quadrant for Digital Commerce

Adobe (Magento) has been evaluated as a leader in the report following its acquisition of the Magento platform in June 2018. This is the third consecutive year that Magento Commerce has been named a Leader. The report evaluated 13 vendors based on their ability to execute and completeness of vision.

Source: https://magento.com/news-room/press-releases/adobe-magento-named-leader-2019-gartner-magic-quadrant-digital-commerce

DEVELOPERS & MERCHANTS

Magento Association Membership signups are open

The Magento Association is dedicated to fostering and supporting technology projects, community events, training and education, and online collaboration. The Magento Association has launched membership signups. Registration takes place on their website (link below). Membership fees are waived until 1.1. 2020 and you are not obligated to renew your membership after this period. Membership fees after 1.1.2020 are still to be determined and will be communicated well in advance.

More info: https://www.magentoassociation.org/membership

MERCHANTS

Enterprise and Mid-market editions of Paradigm B2B Combine

The company was awarded the most medals (17) across both reports. Among the strengths highlighted in the report were:

- **an impressive partner ecosystem**
- **rich extension marketplace**
- **a sizeable customer base you can reach out to**
- **a "rightsized" pricing model and an easy ability to customize the solution via partners**

Source: https://magento.com/blog/magento-news/magento-commerce-leads-enterprise-and-mid-market-editions-paradigm-b2b-combine

News and Announcements from Magento Architectural Team

The most prominent updates and highlights happened since the first release of Magezine published during Magento Imagine this year.

DEVELOPERS

JavaScript Bundling

Andrew Levine from Architecture team recently documented some non-optimal aspects of JavaScript loading in Magento 2 themes (https://tinyurl.com/Magento2JSBundling).

In a joint effort, both Magento Performance and Architecture teams have been working to complete the development of a new JavaScript bundler for Magento 2 storefronts, which is designed to address these shortcomings.

The bundler (tentatively named "baler") will run during the static content phase of a store's deployment and will require little to no configuration. Baler works by analyzing the various ways that JavaScript dependencies are included in a store (mage-init directives, mixins, require, define, etc), and then either bundles or preloads assets to deliver them as efficiently as possible.

Our early tests (with only partial optimizations) have shown a 7-8 points increase in the Lighthouse score for Luma with sample data, along with a substantially improved time to First Meaningful Paint (FMP).

You can follow the development of the project on Github (https://github.com/DrewML/baler/). While these are still early alpha builds, it can work for many stores today.

DEVELOPERS

Performance Efficient Storefront API to speedup GraphQL API

Introducing a layer of GraphQL endpoints as a communication protocol with PWA based storefront since Magento 2.3, we did not introduce performance efficient way to process these API calls under the hood in Magento. So, most of GraphQL resolvers currently rely on the same Magento Service Contracts as REST APIs do. This means that existing GraphQL endpoints use the same coarse-grained Magento services which do not support partial data retrieval and persistence, this neglects most of the benefits brought by GraphQL having performance for GraphQL pretty much the same to what we have for our REST APIs.

This is the issue we started to address introducing new Storefront API layer – ***https://github.com/magento-performance/architecture/blob/graphql/design-documents/graph-ql/storefront-api.md*** as a layer of lightweight APIs which would help us to benefit from all advantages brought by GraphQL.

Magento Performance team will highlight fresh performance measurements for main Catalog scenarios after finishing work on Storefront API for Catalog soon.

DEVELOPERS

Code Ownership

As you know, Magento Architecture team has assigned Components Ownership (***https://github.com/magento/architecture/wiki/Component-Assignments***), where everyone can find precise owner of each and any Magento component. Based on this list, we're adding CODEOWNERS (***https://help.github.com/en/articles/about-code-owners***) file to magento2ce repo. If everything goes well, we'll update other repositories as well soon. PR – ***https://github.com/magento/magento2ce/pull/4668***. CODEOWNERS file used to define individuals or teams that are responsible for code in a repository. The main feature which CODEOWNERS provides is individuals from that file are automatically requested for review when someone opens a pull request that modifies code that they own.

In Magento we introduced the Code Owners mechanism for informational purposes. Nothing is changing in the processes. Dev teams review and merge PRs the same way as they do before. The only difference you will see is that corresponding architects will be automatically assigned as reviewers for PRs. You don't need to wait for review completion. Please continue including architects in discussions related to their components, PR notifications can't replace a good discussion.

The things are different with Community Contributions when many PRs created by full initiative of external contributors w/o preliminary grooming with people from Magento, so that the way how a fix/feature is implemented may not be in sync with our desirable vision. Adding a code owner (architect) assignment automatically based on introduced changes in the scope of PR may facilitate the process of providing feedback and starting the conversation between the contributor and component's architect. Which may decrease time for initial feedback and facilitate the process of arch review for community created PRs, and in general will increase awareness of component owners to the changes happening in their components.

DEVELOPERS

MySQL 8.0, MariaDB 10.3 and 10.4 support and compatibility investigation

Quite a lot of merchants started to switch to MySQL 8.0 recently because of new capabilities and improved performance and scalability provided out of the box. While in Magento we can't start using MySQL 5.7+ specific features (like, JSON index generated column or UUID native support and conversion) for Magento 2.3.* because that would be Backward Incompatible for those merchants who still use MySQL 5.6, we started to investigate a possibility of introducing compatibility with MySQL 8.0 for Magento 2.3, which means that it would be possible to install Magento 2.3.* with MySQL 8.0* relying on existing set of functionality and start running the whole suite of automated tests with the latest version of MySQL. And later in Magento 2.4.* we going to drop support of MySQL 5.6 in favor of supporting 5.7 and 8.0.

There are some issues have been identified to this moment, and one of those issues affects caching of AUTO_INCREMENT value in MySQL 8.0 you can read more about this in this Architectrural proposal – ***https://github.com/magento/architecture/pull/233***

DEVELOPERS

Run Only Impacted Tests for CI/CD

Magento Codebase is constantly growing as well as Magento test coverage, and this is inevitable process. As a side-effect of this process we need to run more and more automated tests for each Pull Request being delivered to Magento, which require more resources to be allocated to prevent increasing the time for running CI/CD builds and waiting in queue. Currently, Magento runs all available tests fully on every PR delivered: Unit, Integration, Magento Functional, Integrity, Static, Web API (REST, SOAP, GraphQL), Performance, Sample Data, etc. This require a lot of resources and time to run all these tests, and this is especially non efficient when changes being delivered are local and affect just a few lines of code. As an alternative we started to consider an option where we will run only impacted tests on created PRs. This approach is not new and applied by big companies with large codebases. Here you may find some more theory on this topic by Martic Fowler – ***https://martinfowler.com/articles/rise-test-impact-analysis.html.*** The main idea is quite straightforward – we need to run all test suite and gather code coverage for each and every test being executed. That will provide us a map of relationships: TEST –> PHP code (set of classes/methods) executed while running this TEST.

Then, accepting a Pull Request which contains some code changes, we need to get a reverse relationship, for each modified class/method we may extract a set of tests need to be launched, and run only those tests but not the whole suite. We think that would be especially useful for community contributions which usually aiming to fix particular bug introducing a few lines of code changes.

It may be still tricky to determine which tests need to be run if changes occurred in XML declaration (like layout.xml or di.xml), so that in this case we may run the whole test suite to make sure that we don't miss anything. And of course, we need to periodically re-build our map containing all the relationships for better heuristics. We already investigated that possibility a couple of years ago, but the major problem for us that time was the performance of xDebug while gathering code coverage statistic, which sometimes maybe a dozen time slower than running code non gathering these data.

But there were several tools appeared recently which show pretty promising results where the performance while gathering code coverage as almost the same as running a pure code.

More to read here – ***https://medium.com/@nicocabot/speed-up-phpunit-code-coverage-analysis-4e35345b3dad***

DEVELOPERS

Deeper Integration with Amazon technologies

Taking into account that Magento is moving towards Service Isolated architecture where components would be deployed on a distributed bases, we need to fill the existing gaps and increase support of Message Queue adapters, as currently Magento supports only RabbitMQ. This is especially important taking into account that another Magento offering – Magento Order Management (MOM) started to support Amazon SQS as a transport for async API calls. Amazon Simple Queue Service, or simply SQS is a simple but powerful service for generating and consuming messages. It can help decouple systems for improved scalability and robustness.

The possible benefits of supporting SQS in Magento are:

- Eliminate administrative overhead, as this is SaaS service provided by Amazon and merchant is not responsible for hosting it. AWS manages all ongoing operations and underlying infrastructure needed to provide a highly available and scalable message queuing service.
- Reliably deliver messages. Amazon SQS transmits any volume of data, at any level of throughput, without losing messages or requiring other services to be available.
- Keep sensitive data secure. Amazon SQS supports server-side encryption (SSE) to encrypt each message body for exchange sensitive data between applications. AWS Key Management Service (KMS) logs every use of the encryption keys to AWS CloudTrail to help meet possible regulatory and compliance needs.
- Scale elastically and cost-effectively. Amazon SQS leverages the AWS cloud to dynamically scale based on demand.

Another topic we got back to the investigation of is the support of Amazon S3 / CloudFront. Magento 2 in its current state provides Pull CDN support (HTTP Caching Reverse Proxy), i.e. supports an ability to generate URL for static files (CSS, JavaScript, Images) pointing to another domain address (not the one where Magento site hosted), and client visiting magento-based website makes the HTTP request for media content to the reverse-proxy service, which retrieves it from Magento and then keeps in own cache. This way we support integration with Fastly for example. In this integration Magento is not aware about existence of CDN at all and that media content is being cached. Contrary to this we would like to provide a support of Push CDN, when Magento would deliver media content directly to a CDN system and retrieve it out there if needed. That would help us to improve scalability of file system operations.

Currently we consider that activity as a way to improve our File System APIs, where Push CDN would be introduced along with File System Adapter, and ideally the business logic which work via File System API should be agnostic whether it relies on File System or on Amazon S3 CDN.

DEVELOPERS

Service Isolation

Storefront components	Store Management / Backend components
Catalog	PIM (Product Information Management)
Checkout	CPQ (Configure Price Quote)
Customer	CRM (Customer Relationship Management)
Orders	Order Management
SEO	Marketing
Store Information	Store Management
	Inventory Management

In the scope of Service Isolation track we introduces a vision of separation Magento application onto two independent applications, each one consisting of a set of independent services. These applications are: Storefront and Store Management. So that, just imagine that Magento as you know it today is split into two independently deployable and scalable applications. Which may evolve and change independently out of each other. The main user of the first application is customer, while for the second one it is a merchant.

One of the major requirements for Magento is to provide a possibility of frictionless upgrades to the newly released versions of the platform, so that prevent merchants to stuck with an outdated version because of upgrade costs or incompatibilities with 3rd party extensions being used and not supporting changes released with the latest update.

Also, some of the changes which either require of modernizing technological stack (especially front-end technologies) or changing the architectural paradigm or business flow may lead to drastic incompatibilities being added to the platform which would affect all the existing consumers, so it's impossible to deliver these kind of changes in patch releases.

To provide an ability of easy upgrades and along with that evolve the platform to address business requirements of big enterprise merchants who have highly customizable integrations with 3rd party systems, it was decided to split the platform on two independently deployable and scalable applications: Storefront and Store Management. Both of applications supposed to have own set of APIs and data storages which may evolve independently out of each other. While the existing monolithic application of Magento would play a role of Store Management application, the new one – Storefront application is started as a greenfield development.

Such approach provides us an ability to design and develop a new version of the platform in parallel while supporting existing approaches and keeping already released APIs forward compatible.

The Storefront application is not a monolithic by its nature but rather represents an isolated set of components, each one representing particular business domain. When talking about components we run into the difficult definition of what makes a component. Magento defines that a component is a unit of software that is independently replaceable and upgradeable/releasable, which implies we look for points where we can imagine rewriting a component without affecting its collaborators. Indeed, many component groups take this further by expecting services to be scrapped rather than evolved in the longer term, for example, a merchant supposed to easily substitute Magento's out of the box store management capabilities integrating 3rd party ERP or PIM system.

IN THIS ARTICLE YOU'LL FIND OUT:

- What are Progressive Web Apps?
- What are the advantages of PWA for business?
- How much does it cost to implement a PWA?
- What are the results after PWA implementation in e-commerce?

By Magezine Team

WHY IS THE PWA TECHNOLOGY SUCH A BIG DEAL IN 2019?

Progressive Web App is the future of web design. Thanks to PWA, users can easily install the store's website on their smartphones. It is a big change for e-commerce. A change that responds to the need to provide customers with the best possible experience in mobile shopping. Is now the right time for online stores to invest in PWA?

PWA SHORTCUT

What is PWA? During Imagine 2019 in Las Vegas, James Zetlen described PWA as a very successful website. This was possible due to the fact that PWA utilizes modern-day technologies, providing its users with an experience comparable to that of native mobile applications. Theoretically, the explanations could finish here. In practice, it is not that simple.

Is PWA technology completely new? In fact, Steve Jobs already mentioned it in 2007 when he introduced his first iPhone to the world. Jobs wanted developers to build iPhone apps using standard web technologies. The apps were supposed to be used by the Safari web browser. Unfortunately, it did not work out then, and soon a real mania for native applications started.

The term "Progressive Web App" was introduced in 2015 by Frances Berriman and Google Chrome engineer Alex Russell. It was first meant to describe web applications that use new features supported by modern browsers, such as service workers and web app manifests.

However, PWA is such a new solution in e-commerce that only a few online stores can boast the implementation of this technology.

What's behind the PWA abbreviation?

- **Progressive** – works for every user, regardless of browser choice, using progressive enhancement principles.
- **Web** – PWA websites are written in languages typical for the Internet e.g. HTML, CSS, JavaScript
- **App** – working just like a native mobile (or desktop) application, a consistent interface, user experience and functionality.

What does PWA give its users? First of all, it provides an improved usability, thanks to the combination of what is best in the Internet use when accessed with a browser and with an application.
PWA provides:

- **Easier access to the site** directly from the search engine – the user does not have to download the application from Google Play or the App Store. Just one click and the page will be saved on his phone (as an icon). Importantly, the saved page will take up much less space than a native application. For example, the native Twitter app is 62.49 MB, while the PWA version of Twitter is only 160 KB.
- **More convenient browsing**, searching for information or buying on a mobile device – all of this happens without installing the application.
- **Faster loading** – compared to websites that do not use this technology. PWA assumptions require it, for this purpose can be used the possibilities offered by browsers e.g. content caching in the browser or lighter media format (webp format instead of jpg/png/gif).
- **Offline access** – thanks to the service workers (JavaScript file), if there are problems with the network, the application still works. Some of the application resources are saved in the device cache so that they run faster and are always available.

PWA sounds like a completely new platform, but it's not; it's a convenient term for a handful of new browser features and design patterns. So it's an evolution of Web development, much like "Responsive" has been.

Jemes Zetlen
PWA Studio Magento contributor

PWA BENEFITS FOR BUSINESS

Speed

This is one of the most exciting functions of PWA. Mobile sites that load in 2 seconds or less have a 15% higher conversion rate than an average mobile site. This means that users leave the site when it is loading for more than 2 seconds. Therefore, changing from an ordinary mobile website to a PWA website will bring significant benefits for businesses. Pages using this technology charge even 1 second. There are several examples in the e-commerce market that confirm this, e.g. the online store Jumia (Africa), which – thanks to a PWA – reduced the bounce rate on the website by 50%. The cosmetic giant Lancôme, on the other hand, implemented PWA in 2017 and recorded an 84% decrease in the page loading time.

Page loading speed is important to increase conversion but it also crucial from the SEO perspective. In January 2018, Google officially announced that the mobile version of its website loading speed will be the key factor when considering the visibility of websites in organic search results.

Engagement

Today's smartphone user has an average of more than 80 apps, but apparently makes use of merely 9 applications per day. Most often these are social media,

games, music or video apps. In fact, it is very difficult to engage a user to re-enter a native shopping application. Therefore, from the perspective of e-commerce, PWA can be very useful, thanks to the possibility of using push notifications. With their help, the shop will provide users with current information of various types, such as order status, delivery date or promotions. In this way, shops will have an additional opportunity to engage their users and encourage them to make further visits.

This is confirmed by the case studies. For example, thanks to PWA and push notifications, the eXtra online store from Saudi Arabia has increased user engagement 4 times. After installing a PWA, users who agreed to notifications spent twice as much time on the site. As a result, the brand recorded a 100% increase in sales from their web push notifications.

Conversion

There are already implementations of PWAs, after which websites have significantly increased conversion. A good example here is the search engine of Trivago hotels. Approximately half a million people have already added Trivago PWA to their home screen and, as a result, the site has managed to increase its engagement whereas the conversion rate has gone up by 97%. What about e-commerce? One of the most frequently mentioned case studies in the context of PWA and conversion is the one of AliExpress. After the implementation of a PWA, this online store recorded an increase in conversion of 82% to iOS.

IS PWA A GOLDEN SOLUTION?

First of all, PWA is a relatively new solution in itself, especially in the case of Magento. This can result in high implementation costs. Prices are rising, especially when creating dedicated solutions. However, Magento has met the demand for this technology and created a set of tools named PWA Studio which allow building online stores in PWA way. This tool helps developers learn PWA techniques, build interfaces, create components and extensions for reuse or sale on the Magento Marketplace.

The implementation of a PWA at the moment is primarily an investment that requires time and the creation of a whole foundation for a modern solution, as well as its further development.

PWA flaws:

- **less access to some functions** – PWA can connect to the camera, microphone etc., but in the native applications, the access to the features is wider. Despite this in typical implementations of online stores, what PWA offers will be enough.
- **Limited functionality on iOS devices** – unfortunately, the PWA technology is not yet fully implemented on iOS devices.
- **Unable to collect user data** – unlike native mobile applications, PWA pages do not have access to user personal data, such as contact lists, connections and messages.

Native Apps vs PWA - basic comparison

Native Apps	Progresive Web App
A native app is built specially for one platform	PWA in an app taht runs in a browser and behaves pretty much like native app
The separte code base for each platform such as Android, iOS	Dont't need separate code base, don't need to install from Goolfe Play or App Store
Unbeatable user-experience due to native hardware acces	Superior user experience through modern web standars
Requires higher budget to write platform-specific code	Relatively cheaper than native app as runs on multiple platforms with a single code
Need to download from the app stroe	"Add to Home Screen" prompts and runs directly in a browser
Requires more space	Very little space is used
Needs higher data consumption and network	Works well in slower network and offline

Source: https://medium.com/quick-code/progressive-web-app-heres-everything-you-need-to-know-about-pwa-cf2fefcbf24e

HOW MUCH DOES IT COST TO IMPLEMENT A PWA?

Building a website based on a PWA is much cheaper than creating a native application, which makes it possible to optimize costs. A company that needs a native application will need to create one for both Android and iOS devices. This can be very expensive because each of these systems will require a separate team of programmers for each browser and mobile application. One team and one technology are enough to create a PWA.

Building a PWA from scratch costs $400k- $1m. The implementation of existing solutions such as PWA Studio, Vue Storefront, Deity, etc. is a closing cost of 300–600 WH (working hours), so the estimated total cost of such project would start from $15k.

WHAT TECHNOLOGIES DOES PWA USE?

The following are used to create a PWA:

- frameworks and libraries of JavaScript such as React.js, Vue.js and Angular,
- web app manifest – manifest.json (defines accents of mobile and desktop browsers, such as the color of the top bar and the page icon)
- Service Workers – is a script that runs in the background, separate from the web page (allows browsing content without access to the web).
- Verification of compliance with the PWA is done by checking the Progressive Web
- App Checklist (published by Google) and passing the Google Lighthouse benchmark.

Many PWA solutions for e-commerce appear on the market. The important information for merchants is that not all of them are the same. What is the difference? Approach to building PWA. Developers can create custom PWA from scratch or they can use already existing framework to do that. Example? In the first case, they can use PWA Studio, which is a collection of tools that lets developers build complex Progressive Web Applications. While in the second case, they can lunch ready-to-use framework like Vue Storefront or Deity and connect with Magento backend through the API.

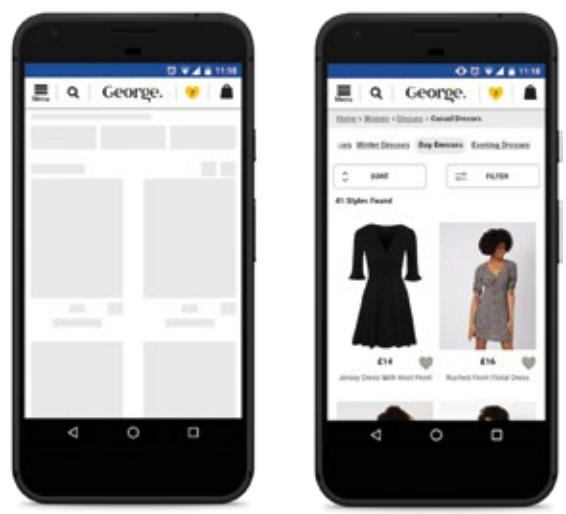
PWA SUCCESS STORY IN E-COMMERCE

AliExpress

Before AliExpress invested in the PWA, the brand tried to gain conversions through a mobile website. However, it did not bring satisfactory results. The website did not provide users with adequate experience on mobile devices. After the implementation of the Progressive Web App, AliExpress increased the conversion rates for new users by 104%. This investment has also resulted in an 82% increase in Safari's conversion rates. What's more, users now visit twice as many pages per session, and the session time has increased by an average of 74% across all browsers.

George

George is one of the leading fashion brands in the UK. The company wanted to provide a better quality of shopping through mobile devices, which was achieved after the implementation of a PWA. In accordance with the best practices, the PWA managed to significantly reduce the page loading time. The company also recorded a 31% increase in conversion and a 20% increase in page views per visit. In addition, the brand has introduced "Add to home screen" notifications that appear to the users when they enter the site. The use of notifications has increased the time of customer service by 28%.



Skeleton screen during page transition

Page fully rendered after page transition

Source: https://developers.google.com/web/showcase/2018/asda-george

LuxMart

LuxMart, a Canadian manufacturer of luxury accessories for mobile phones and laptops, was struggling with low conversion rates on mobile devices, with 85% of the traffic on the brand's website coming from mobile users. To improve the brand's mobile customer experience, they had to come up with a solution that would be both features-rich and fast enough to generate conversions. For this reason, the brand decided to use a PWA. The effect? Thanks to the PWA, the conversion rate on mobile devices doubled from 0.6% to 1.24%, and on some days of the month, after the implementation of the PWA, it increased up to 3.1%.

Results after PWA implementation in e-commerce

George.com	▪ **3.8x** faster average page load time ▪ **2x** lower bounce rate ▪ **31%** increase in conversion rate ▪ **20%** more page views per visit ▪ **28%** longer average time on site for visits from Home screen
AliExpress	▪ **104%** for new users across all browsers; **82%** increase in iOS conversion rate ▪ **2X** more pages visited per session per user across all browsers ▪ **74%** increase in the time spent per session across all browsers
Jumia	▪ **38%** Open rate 9X more conversion on previously abandoned carts from web push users ▪ **7.85%** conversion rate on previously abandoned carts from web push users, vs. 4.5% for native app

JOIN THE PWA MOVEMENT

Should online shops invest in PWA? It is worth noting that native store applications are mainly used by the most loyal and engaged customers. It is hard to expect all users to download the application to their phones, e.g. during one-off and random purchases. In this case, PWA will prove to be the right choice, as it will provide users with an experience similar to that of an application, which will contribute to an increase in conversion.

Secondly, the brand conversion on mobile devices is still low and most brands are not satisfied with it. The number of users using the Internet on a mobile basis is growing year by year. Unfortunately, this does not lead to an increase in conversion, which usually achieves 1/3 of the results of the desktop. The use of PWA technology can change this.

That is why, following the Forrester Research specialists, the message to e-commerce business owners would be: Join the Progressive Web App Movement. ●

By Magezine Team

Sources:

1. Case study: *AliExpress, Google Developers,* https://developers.google.com/web/showcase/2016/aliexpress
2. Case study: *Lancome, Google Developers,* https://developers.google.com/web/showcase/2017/lancome
3. Case study: *George, Google Developers,* https://developers.google.com/web/showcase/2018/asda-george
4. *51 Jaw Dropping App Usage Statistics & Trends,* 2019, https://techjury.net/stats-about/app-usage/
5. *Progressive web apps (PWAs) for SEO: Benefits, stats, examples,* https://searchenginewatch.com/2019/05/02/how-progressive-web-apps-pwas-for-seo-benefits-stats-examples/
6. Leif Bryan, 2019 — *The Year of Progressive Web Apps,* https://medium.com/datadriveninvestor/2019-the-year-of-progressive-web-apps-3027aea291f9
7. Robert Williams, *Luxmart doubles sales conversions with progressive web app,* https://www.mobilemarketer.com/news/luxmart-doubles-sales-conversions-with-progressive-web-app/539638/
8. https://www.pwastats.com

WHO WILL WIN THE PWA BATTLE?

selected solutions overview

IN THIS ARTICLE YOU'LL FIND OUT:

- What are the two main approaches to PWA?
- What PWA solutions are available on the market?
- What are the differences between the most popular solutions?

By Magezine Team

2019 is the year of mobile. Oh, wait – didn't we hear that last year? It was similar in 2017, wasn't it? That's right. However, now we can talk about the upcoming breakthrough in the mobile market, all due to the growing popularity of PWA applications. Will Progressive Web Apps turn out to be a revolutionary solution? We have to wait for the answer a little bit longer. But it is already known that the development of the PWA technology will certainly bridge the gap between websites and native mobile apps.

PWA has been a hell of a buzzword lately. We are almost certain that our readers know this term. If not, we recommend you to read the previous article in the Magezine, in which we introduce the topic of Progressive Web Apps and indicate the main benefits of using it in your business. In this article we will focus on presenting a few popular solutions for e-commerce on the market. But before we start describing them, one very important remark.

When considering PWA, at least two approaches have to be distinguished. The first one are the solutions which are a standalone PWA storefront for e-commerce that lets you connect with backend (e.g. Magento, Prestashop, Shopware) using an API. In this group we can point to Vue Storefront, Deity Falcon or FrontCommerce. In the second approach, the solutions offer a set of tools to build PWA – and in this group we will indicate for example PWA Studio and ScandiWeb.
Let's start with Magento solution.

PWA Studio

A set of tools that allows to develop, implement and maintain a storefront for Magento 2.

Launched in 2018, Magento PWA Studio is a new solution implemented in Magento 2 and it is designed to make conversion of e-commerce to PWA smoother. Magento PWA Studio uses modern tools and libraries to create a system and construction framework in accordance with Magento extensibility principle.

Magento PWA Studio provides the following tools:

- **PWA Module:** This tool provides modular help, server-side functions and is the basis for all topics created in Magento PWA Studio. It delivers Shell applications, manages RootComponent tasks, and integrates GraphQL workloads into server rendering for better optimization. Easy to install, this module is activated in the final phase and applies to all themes created in Magento PWA Studio.
- **Pwa-buildpack:** A set of necessary plugins and Webpack tools used to create Magento Studio PWA themes. It is also used to configure the local development environment of the Magento 2 platform.
- **Peregrine:** This is the React component library that simplifies the development of PWA Magento by providing basic functionality. Storybook JS, which provides documentation for developers, is also used.
- **Venia storefront:** Magento 2 site built with PWA Studio tools. Venia Storefront is a demo theme that can be used as a basic theme, built entirely with the above-mentioned tools.
- **UPWARD:** It is a server that could be considered as middle layer between PWA and APIs, it helps unify all APIs in one place, so there's no need of keeping track of multiple endpoints and how to access them.

Both developers and customers can benefit from the PWA Studio. Simply put, it allows to make online customers' purchase journey easier and faster. From a developer's perspective, the PWA Studio is a new front-end architecture which allows managing all channels via one code base, one deployment, and app. Of course, some weaknesses can be found as well.

What are the biggest pros and cons of using Magento PWA Studio?

PROS

- **Providing expansion tools** – you can use them in your local environment to create PWA applications.
- **GraphQL usage** – GraphQL includes declarative retrieval of data from queries without excessive retrieval. With a single back-end, you can manage multiple users from different sources.
- **Easy configuration** – Magento PWA Studio has an easy backend configuration process; allows you to clone the repository and assign the URL of the Magento instance to a.env file, and allows you to run the application efficiently after the command is executed.
- **Strong community** – an advantage, but also a trademark of Magento, is an involved and large community. The PWA Studio also benefits from this, having a lot of contributors from all over the world.

CONS

- **Lack of features** – some of the functionalities are still in "to do" stage.
- **Errors to deal with** – at the moment there are still vulnerabilities that the Magento team is working on (e.g. how to validate the password when creating a new account).
- **API is not ready** – the GraphQL API prepared for PWA Studio is not fully ready for production implementation, and doesn't cover 100% of the functionalities available in Magento, such as language or currency change, etc.
- **Supports only Magento back-end**

Deity Falcon

Progressive Web App library for any type of website. Fully Open Source, Platform Agnostic and headless.

DEITY Falcon, a result of the work of developers from the Dutch company DEITY, is a stand-alone but modular library to easily build headless PWA websites. Deity Falcon started in October 2017 with their first PWA solution. Over a year later, in November 2018 they released the first open source version. The solution is built using ReactJS, NodeJS and GraphQL, and supports Magento 2 PWA storefront, Wordpress PWA and BigCommerce PWA Storefront.

During the development of Falcon, emphasis was placed on compliance with the principles of F.I.R.E., which means:

- **Flexible** – PWA can be used to build a website, blog, shop or even a portfolio.
- **Integrable** – easy to integrate.
- **Reliable** – it is able to handle even very high traffic on the website and is easily scalable.
- **Extensible** – easily expandable with new personalized features.

When it comes to indicating possibilities and strong sides of this solution, it is worth noting that in DEITY Falcon you can work independently on the front and back-end, and this significantly affects the optimization of development time. This is not the only advantage.

- **It only takes a few minutes to get started with CLI tool** create-falcon-app, which enables you to create an application based on DEITY Falcon with just one command.
- **Client rendering (SPA)** to increase page speed and reduce server load.
- **Built-in Server Side Rendering (SSR)** – you don't have to worry about SPA SEO complications.
- **Service worker** to provide application caching and PWA features such as offline capabilities and add to home-screen.
- Falcon's architecture allows it to be very **modular, lightweight and scalable.**

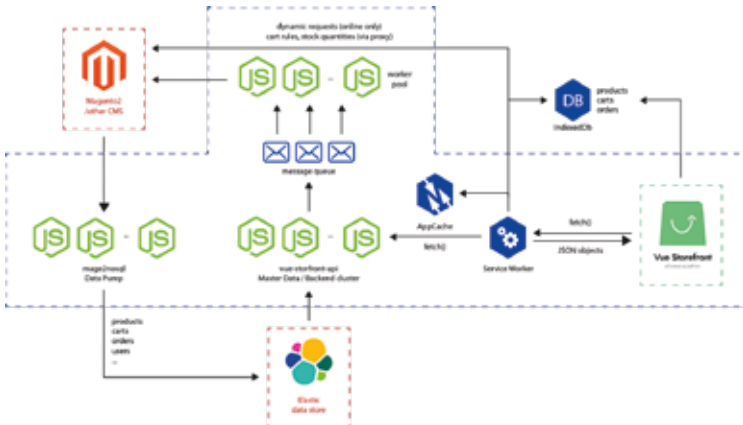
Jamie Maria Schouren | Co-founder at DEITY

DEITY Falcon is a launching platform for Progressive Web Apps. It allows merchants to create highly engaging user experiences and ultrafast frontends. With DEITY Falcon you can create a top performing Progressive Web Application based on standardised web technologies, fully enhanced with the newest JavaScript features, making them feel and function like true native Android and iOS apps. Working closely with top players such as Google and BigCommerce, DEITY Falcon is setting the new standard for modern web experiences. Combining both technological stability and full feature-rich web applications, DEITY Falcon is ready to revolutionize any webshop. It is built with the newest web technologies including NodeJS, ReactJS and GraphQL. Using these technologies, the Falcon PWA becomes highly scalable, reliable and extensible and immune to traffic overloads. DEITY Falcon has a unique architecture which allows for seamless integration with any existing platform, and unlimited scalable and extensible possibilities. DEITY Falcon can integrate with any data source, no matter the size, while handling millions of traffic hits at the same time. Blending mobile and web is a game-changer in the industry that will lead to astonishing business results—making this a can't-miss technology for retailers.

Vue Storefront

Standalone PWA storefront for e-commerce, possible to connect with any e-commerce backend through the API. Free and open source with built-in support for SSR, PWA and SPA.

Vue Storefront architecture



Source: https://github.com/DivanteLtd/vue-storefront

Vue Storefront is a PWA solution tailored to the needs of e-commerce and integrated with the largest platforms - e.g. Magento, Shopware and Prestashop. It is so far the most popular solution on the market.

Vue Storefront is built using vue.js. As you will see in the table at the end of this article, most of the solutions are based on React.

Divante - Vue Storefront's provider decided to use vue.js because, as they claimed, it is a technology that can be mastered in two weeks, so it is possible to quickly qualify employees who are able to work on PWA stack. What is worth mentioning - developers can expand JavaScript code both on the client and server side - the data is automatically synchronized. What are the other advantages of Vue Storefront?

- **Platform-agnostic:** Vue Storefront abstract the common data by converting the platform-specific API calls with the help of its layer.
- **Frequent updates:** new updates every week.
- **Effective support**
- **Server-Side Rendering:** SSR renders when the search engine crawls requests, making it possible to generate HTML for pages that are more dynamic and whose content is unknown at the time.
- **Offline browser storage:** It stores data in IndexedDB and LocalStorage what enables native caching (Cache API).

Of course, there are some critical voices in terms of this solution. One of the cons is e.g. the lack of complete documentation to be found in Vue Storefront. What else?

- **Uses REST API:** It can't make asynchronous calls. As compared to GraphQL, it falters with request response.
- **Lack of compatibility with several Magento functions:** At this moment Vue Storefront is missing a few Magento features.
- **Payment methods:** incompatible with some payments providers.
- **iOS:** Some features are not compatible with iOS devices.

Sander Mangel — Technology Leader at Vue Storefront

PWA offers exciting new possibilities for e-commerce, improving the users shopping experience. Vue Storefront is there to make building a PWA a great developer experience as well. It provides scaffolding, boilerplate for features and takes care of edge cases for the developer. And not just for the frontend, Vue Storefront offers the right tools to quickly implement a headless architecture. While giving the merchant a robust solution, with an extensive and thriving community and a list of skilled solution providers to choose from. Vue Storefront offers an enterprise level solution on the cutting edge of PWA tech.

Scandi PWA

The solution provides a PWA for Magento-based stores. The First Open Source PWA Theme for Magento.

Developed by Scandiweb agency in 2018, uses React as front-end app framework. The latest technological stack (included Redux and GraphQL as well) provides users with the best experience. In terms of technologies, it is worth to mention: ready development environment on Docker, Varnish & Persisted queries for data sync, caching layer for 0.020 seconds response time, and no extra DB or middleware. ScandiPWA uses technologies such as GraphQL, Varnish and Redis to improve the performance of the website. These elements are required for the project to enable ScandiPWA to work with the shop.

Since the ScandiPWA is closely integrated with Magento backend, and designed to be as simple and familiar as possible, ScandiPWA theme can be implemented on any Magento 2.3 version project without any changes to its infrastructure. Find out more about ScandiWeb in this issue of Magezine.

ROADMAP

DONE
- Full PWA Theme for Magento v 1.0
- Caching layer & Persisted queries
- Google recommended SSR for SEO of PWA
- 301 redirects
- Layered navigation
- Android app deploy from PWA
- Mobile-first theme 2.0
- Cart price rules
- iOS app deploy from PWA
- Reviews
- Wishlist

IN PROGRESS
- Product Compare

TO-DO (due summer 2019)
- Advanced GTM data layer
- Payment methods
- Shipping methods
- Multi-store, currency, translations
- Advanced SEO module

Source: https://scandipwa.com/#about

Alfreds Genkins — ScandiPWA Front-end Tech Lead

Our dream is to make the Magento ecosystem leaders in PWA adoption. The key to bringing our dream to life is to make PWAs accessible, transparent, and affordable. And those are the pillars ScandiPWA is built upon - an open-source plug-&-play Magento-first PWA theme that can be implemented by any Magento developer in a day's time.

Front-commerce

Front-Commerce is a solution that allows you to manage the front in PWA with the use of the latest technology in the architecture of microservices.

The project started in 2015. Today it is a ready-to-use technology provided by the French company Occitech. The solution was developed under Magento 2, but it is also compatible with Magento 1.

The operation of Front-Commerce is essentially based on a central interface responsible for communication with various APIs. It is therefore built with Elastic Search, Redis, GraphQL and Node JS technologies. Elastic Search enables the management of directory data with the possibility of extension, while Redis limits API calls, acting as a central cache. It then replaces various API data by a single GraphQL to a page theme that gets it from React.

This is not an open-source solution. Front-Commerce balances this fact, however, with a long list of features it supports.

Front-Commerce's components
- Magento 1 or 2 REST API additional endpoints
- NodeJS middleware (Express)
- UI components library following the "Atomic Design" pattern (comparable to "Peregrine" in PWA Studio)
- Blank React/GraphQL e-commerce theme as a starter (comparable to "blank" theme in Magento)
- Documentation

Tools for Quality
- JS Unit testing with Jest and jsverify
- Remote interactions tested with Pact
- Design System with Storybook
- Client and server hot reloading
- Skeleton project to bootstrap your project
- Debugging using DEBUG environment
- Docker configuration for local development

Products
- Promotions
- Upselling, cross-selling
- Tier prices
- Stock
- Multiple-images swatches
- Custom attributes
- Custom options
- Media gallery (images and videos)

User account
- Create an account
- Forgotten password flow
- Newsletter subscription management
- My Account
- information
- address book
- orders
- renew orders
- Wishlist (with Magento2 native feature)
- Impacts of group belonging

Payment
- Paypal (embedded & platform agnostic)
- Stripe (embedded & platform agnostic)
- LYRA / Payzen (embedded & platform agnostic)
- Ogone (embedded & platform agnostic)
- Gateway for Magento's payment method modules (requires a ~20 LoC adapter) such as Paypal, Adyen or PayZen

Search and Product attributes
- ElasticSearch search engine (with Smile ElasticSuite)
- Dynamic filtering and layered navigation (with Smile ElasticSuite)
- Reusable and extensible ElasticSearch core libraries to build custom queries easily
- Custom attributes
- All product types handled

General features
- Theme and components override mechanisms
- Extension mechanisms for GraphQL and node server features
- Connect other API and expose data as part of the GraphQL schema
- Use Magento 2 GraphQL endpoints
- Middleware caching layer (Redis based) with cache invalidation
- Local browser GraphQL cache

Cart
- Cart page
- Update quantity and options per line
- VAT and shipping estimate
- Coupon codes
- Related sales
- Magento guest cart merging on login

SEO
- Server Side Rendering (SSR) for search engines
- Magento's URL keys support
- Products's and breadcrumb Micro-data
- Sitemap generation
- Magento's redirections
- Media optimizing middleware

Content
- CMS Pages and Blocks
- Contact forms
- Categories Pages
- Product Pages with media gallery
- Widgets
- Lazy loading
- WordPress integration

Analytics
- Out of the box analytics support for more than 100 integrations
- Possibility of adding custom events

Multi-stores
- Multi websites
- Multi stores
- Multi views (and languages)
- Auto-select language from browser

Source: https://www.front-commerce.com/en/functional-coverage/

Karim Djebbar — Business Development & Partnership

Front-Commerce is a React e-commerce storefront started in 2015. It is ready-to-use and a pioneer on the PWA solutions market. The first releases targeted Magento2, but Front-Commerce is platform agnostic and also supports Magento1, making it very relevant for merchants worried about M1's EOL. Other platforms support is on the work. Since early 2018, several e-shops (Magento2 and Magento1 based) have been deployed in production with Front-Commerce. For example: "Chaîne Thermale du Soleil", "Terrang", "Autobertnard", and our first Magento 1 reference, "Collegien". Due to its anteriority on the market, Front-Commerce has the largest functional scope. Another advantage of the Front-Commerce solution over its competitors is the theme and components override mechanisms.

Front-Commerce aims at being PWA Studio friendly. Being React and GraphQL based means that developers working today with Front-Commerce could be efficient with PWA Studio when it will be feature-complete, and vice-versa. We will also look into synergies between our projects (UPWARD, hooks, design tokens, extension development...) as we progress. Its code is fully delivered to its customer and digital partner agencies. We provide a technical support and have a public documentation.

PWA solutions comparison

	PWA Studio	Scandi PWA	DEITY Falcon	Vue Storefront	Front Commerce
Started	June 2018	September 2018	October 2017	November 2017	2015
Frontend Base	React	React	React	Vue.js	React
Provider	Magento	Scandiweb	Deity	DivanteLtd	Occitech
OpenSource	Yes	Yes	Yes	Yes	No (open code)
Default Integrations	at least Magento 2.3	at least Magento 2.3	at least Magento 2.2, Wordpress	Magento 1 + 2, Shopware, Pimcore, CoreShop, Wordpress, EpiServer, SpreeCommerce, Odoo ERP, BigCommerce	Magento 1 + 2, Wordpress
Additional Tech-Stack	Redux, GraphQL, Webpack	Redux, GraphQL	NodeJS, GraphQL, Apollo, Koa, Webpack	NodeJs, Vuex, GraphQL, Webpack	NodeJS, GraphQL, Apollo, Express, Webpack
CMS	CMS API / PageBuilder	CMS API	Wordpress API	CMS API	CMS API, Wordpress API
Payments	Paypal (Braintree)	partial support	Paypal, Adyen (in progress)	Paypal, Stripe, Klarna, Mollie, Adyen	Paypal, Stripe, LYRA / Payzen, Ogone
Demo	veniapwa.com	demo.scandipwa.com	demo.deity.io	demo.vuestorefront.io	demo.front-commerce.com
Projects Live	creatuitypwa.site www.ukmeds.co.uk	www.hotme.ca		kubotastore.pl www.klebefieber.de soboredclub.com www.meubelplaats.nl	www.chainethermale.fr boutique.chainethermale.fr www.compagniedesspas.fr www.terrang.fr www.autobernard.com collegien-shop.fr

Source: Björn Meyer, PWA - E-Commerce - Compare List, tinyurl.com/y4wy6dol

CHOOSE THE BEST SOLUTIONS FOR YOUR E-COMMERCE

Comparing all of the solutions is quite a big challenge, especially in such a dynamically changing environment. We couldn't overview all PWA solutions available on the market. For a very simple reason – newspaper has its own law and this text had to meet the volume requirements. In order to organize all of the information gathered in this article and to give you a general overview of the situation on the PWA market, we have prepared a useful table.

AFTERWORD

As we mentioned, PWA is now a really hot topic (after all, we devoted the whole issue of Magezine to it!). New solutions are appearing, and the existing ones are developing their tools, adding new features all the time. The number of contributors is also increasing.

We are aware that at the time of publication, some of the information may turn out to be outdated. If you think that the second part of this comparison – with new solutions – might be helpful, let us know. We also wonder what the future holds, but we know one thing for sure – the PWA movement is unstoppable!

By Magezine Team

Sources:

1. Björn Meyer, *PWA E-Commerce solutions compared (open-source)* https://medium.com/@bjoern.meyer/pwa-ecommerce-solutions-compared-83bb498433e9
2. https://www.front-commerce.com/en/home/
3. https://www.vuestorefront.io/
4. https://github.com/deity-io/falcon
5. https://magento-research.github.io/pwa-studio/technologies/overview/
6. https://falcon.deity.io/docs/getting-started/intro
7. https://github.com/DivanteLtd/vue-storefront
8. https://magento-research.github.io/pwa-studio/technologies/tools-libraries/
9. https://github.com/scandipwa
10. *ScandiPWA Architecture and Design Insights using ReactJS + MobX*, https://blog.scandiweb.com/article/scandipwa-architecture-and-design-insights
11. Tom Karwatka, *Why should anyone use Vue Storefront if there us PWA Studio by Magento?*, https://divante.co/blog/vue-storefront-pwa-studio-magento/
12. Yogesh Suhagiya, *Magento PWA Studio Vs. Vue Storefront: A Comparison*, https://aureatelabs.com/pwa/magento-pwa-studio-vs-vue-storefront-a-comparison/

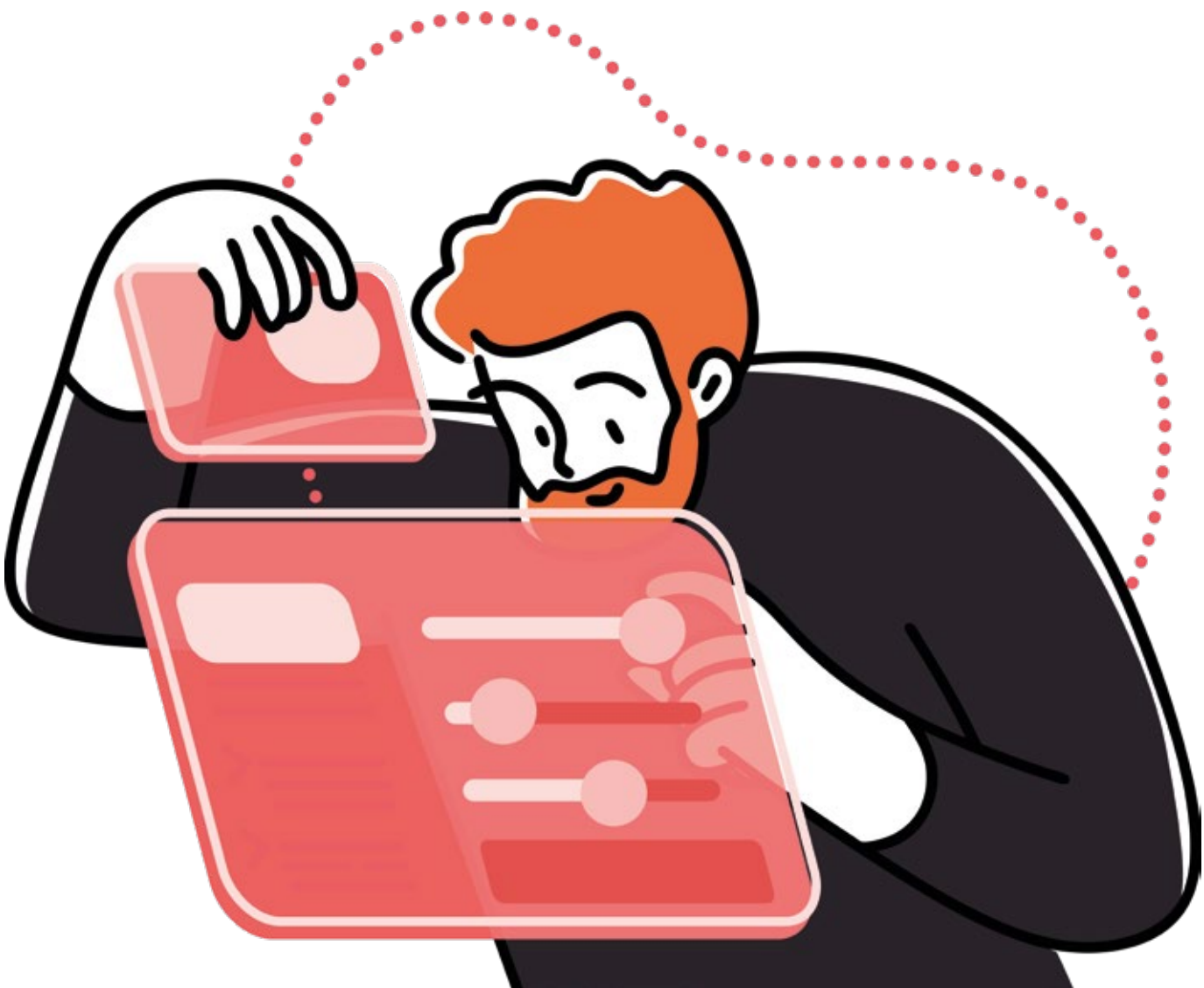
HOW WILL MAGENTO EXTENSIONS FIT INTO PWA?

IN THIS ARTICLE YOU'LL FIND OUT:

- Do Magento extensions work with PWA?
- What does the PWA market look like so far?
- Vue or React - which is a better option?

Jisse Reitsma

People love Magento. People hate Magento. But what I have learned about the current Magento 2 frontend is that everybody hates it. Its weird mix of Require, jQuery, Knockout and custom code is outdated and overly complex. I would even go as far as to say that Magento 2 projects have become more expensive simply because this frontend is so complex.



THE PROMISE OF A NEW FRONTEND

The PWA hype might be a healthy reaction to the dissatisfaction with this frontend. I have been training Magento developers on React now for some time and have learned a few things from it: Redux and Apollo might be a bit overwhelming at first. But once you get the hang of it, it is a breeze compared to the current frontend. To proof my point, I have been building a dummy shop myself (with React, Redux, Bootstrap, Apollo and GraphQL) and got routing, CMS parts, catalog, a cart and login working within 24 hours or less.

The promise of React (and likewise Vue) is that it becomes easier over time. The learning curve is much shorter than with the regular M2 frontend and at the end of that curve, it becomes fun. And it allows for faster development.

THE PWA LANDSCAPE AS OF YET

The Magento ecosystem is moving towards PWA. About three years ago, Magento was struggling with its releases of M2 and for the frontend, this opened up opportunities for other players. Dozens of companies ran Angular, Vue and React projects, many of them were dumped on GitHub. Only a few of those refactored their code for proper sharing: DEITY (React), FrontCommerce (React) and VueStoreFront (VueJS). Since then, other solutions have popped up like ScandiPWA (React) NeoStoreFront (React) and there might be more.

Now that Magento PWA Studio has stable releases (what?), some people seem to be waiting for a ready-made shop without the need for much customization. They are wrong. The real deal is with the technology.

You don't need to wait for Magento, you don't need to choose between DEITY or VueStoreFront. The technology is already out there and ready to use. All of those PWA providers are proof of that. Everybody is able to build a PWA now.

THE FIRST CATCH: EXTENSIONS

There is, of course, a catch: PWA providers might be building great solutions. But if none of those providers offer out-of-the-box support for Magento extensions, how can they even be compared with the current Magento frontend? How can these solutions be the future if they do not incorporate Magento extensions?

I can't offer a straight answer to this, no one can. But I believe the question itself has been put wrongly. It is similar to the assumption that Magento 1 extensions would work with Magento 2. The architecture is different, so are extensions. A better question would be how extensions will fit into this picture.

THE DEVIL IS IN THE DETAILS

So much for the generic PWA rant. The problem is that the devil is in the details. Without understanding the technology, it is impossible to make proper choices regarding PWA. For instance, one of the more popular PWA providers of the moment is VueStoreFront and they have chosen Vue. All of the other PWA providers mentioned above, including Magento, have opted for React. Does that make Vue a bad option? No, it's just a matter of taste.

The problem is that the devil is in the details. Without understanding the technology, it is impossible to make proper choices regarding PWA.

VUE OR REACT?

When I'm asked whether developers should opt for React or Vue, I usually say "whatever". There are differences in the ecosystem, the specificities of the language itself, religious wars have started because of it. In general, you could say that React is more complex than Vue at first, so Vue is easier to start with. However, Vue becomes also complex over time (and in e-commerce apps) and that's where the initial investment in React could pay off. Bottomline is that you either like the one or the other or both (or none). The initial gut feeling probably leads to the right choice. And if not, you simply switch to the other option after a few years.

The new frontend is supposed to be easy: Components are easily built with Vue or React. And I confirm this. I would say that for a junior frontender, the choice between Vue and React is just a personal choice. Start playing and you'll see. Once things get more complex – Redux middleware or Vuex plugins – it is much more about advanced JS patterns than it is about the syntactical sugar.

THE SECOND CATCH: EXTENSIONS

There is another catch in this PWA story. And that's again: extensions. The choice between Vue or React might be a personal one. But the reality is that a Magento shop nowadays often relies upon extensions. And extensions are mostly based upon the architecture that Magento picks. It's what happened with the current frontend (that sucks by the way). It might happen with the new frontend.

So, if Magento bases PWA Studio on React (and not Vue), this might cause extension vendors to consider only supporting React. Right? I think that would be wrong. If React is enforced by Magento and/or extension providers, then in five years, we will bitch about an outdated frontend again. And yes, React will be outdated in five years, simply because we humans are awesome in creating new and better things.

HEADLESS IS THE KEY

PWA is not the right focus here, headless is. We do not want Magento (or extension vendors) to dictate to us what kind of frontend technology we should use. Instead, we want to build our own frontend. And that reduces Magento to a headless system: The head is something we pick and make pretty using Vue, React, Angular 42 or whatever. The body – Magento – remains more or less the same: An API that does a lot of things in the background, while the frontend interacts with it. And the API of choice here is GraphQL.

GRAPHQL IS THE WAY TO GO

In my opinion, the real revolution is not about the actual frontend. It is about the new GraphQL API of Magento 2.3. And now that numerous endpoints have been added to Magento 2.3.2 (you guys rock), I would say that 90% of the needed functionality is there. GraphQL is ready.

And the other 10% that is not ready is easily built as well. Creating a new endpoint is nothing more than adding a etc/schema.graphqls file and a PHP class implementing \Magento\Framework\GraphQl\Query\ResolverInterface. Your schema.graphqls can extend upon existing schemas as well; for instance, to add new product attributes. And if you don't like existing endpoints, modify the responsible resolver class by intercepting its resolve()method using plugin methods like beforeResolve()and afterResolve(). And the API functionality can easily be guaranteed with simple integration tests and API functional tests for your endpoints that compare input with output.

REFACTORING AN EXTENSION TOWARDS GRAPHQL

To embrace GraphQL, lots of extensions will need to be refactored. An extension should not put a lot of logic in Blocks and templates. Blocks are meant for rendering, while templates are just there for output. Instead, logic should be moved to ViewModels. And the real logic shouldn't even be in ViewModels: A ViewModel is just collecting data (modelling) for the sake of the view. Instead, the real logic should be contained within repositories, data providers, services, use your imagination. If your module is properly structured, connecting your logic to a GraphQL endpoint is easy. And because it is so easy to do, a well-written PWA-ready extension should, at a minimum, offer a proper GraphQL API (if applicable).

Also, take a look at the modularity of your module: If you offer output in both the current frontend and GraphQL, make sure to split up your module in submodules: One FoobarStoreFront module for the current frontend, one FoobarGraphQl module for GraphQL. People building PWAs don't need your old-school Blocks. While you are at it, create separate modules for your core API, your backend, your CLI and so on. A bonus: Your extension will be ready for the new miauw-miauw Service Oriented Architecture of Magento 2.4+, at least in theory.

DIFFERENCES IN PWA PROVIDER ARCHITECTURE

Note that different PWA providers use different PWA architectures: DEITY, VueStoreFront and FrontCom-

merce connect from their browser-based PWA to a Node server that again connects to Magento and other services (ElasticSearch, Redis, CMS, PIM). On the other hand, others connect their PWA directly to Magento GraphQL.

This means that the PHP part of an extension might be the same. But the PWA part might connect to a flexible source. An initiative like UPWARD is meant to offer a generic solution for this, but unfortunately, few PWA providers support it. However, I believe that this difference in architecture shouldn't stop an extension provider from doing its job.

BRAINDUMPS ON REACT COMPONENTS

Extension providers should focus on the GraphQL API because that's the thing that glues frontend and backend together. Extensions could also supply frontend components, without needing to know the entire architecture of the PWA provider of choice.

For instance, a React component could be shipped together with the main extension as a proof-of-concept to fetch data from GraphQL using a simple fetch() or maybe an Apollo setup. Best practice is to split up that component into a container (the AJAX part) and a presentational component.

The system integrator or the PWA provider is then able to replace the dummy container with a solution-specific container (for instance based on Apollo Client or Relay). And the same applies for Vue.

EXTENSIBILITY IN A REACT OR VUE APP

While the GraphQL API offers a solid extension point for extension providers, a PWA does not. Perhaps this is how it is supposed to be: A React app, for instance, is composed out of React components that click together like Lego stones. Perhaps an extension provider should simply supply the Lego stones together with some instructions but let the system integrator do the clicking?

Theoretically, a PWA could offer for extensibility nonetheless; for instance, allowing for extension components to be automatically injected into container components (I promised not to mention extension.json). Or it could simply be in the form of extension hooks for things like routing, data retrieval, caching policies.

THERE IS MORE TO IT

And it's exactly extension hooks that DEITY and VueStoreFront have been working on. Most of their work seems to be put into their middleware (Falcon Server, VueStoreFront API), while their PWA uses the Lego stone standard. And extensions, therefore, become most useful in that middleware layer, not in the PWA. That's fine. But it makes me wonder if component standards are ever to become reality.

There are more challenges ahead: Commercial extensions can be offered via the Magento Marketplace or similar. I'm not sure if the Magento Marketplace is going to be able to support commercial NPM packages. Perhaps commercial extensions will need their valuable logic to the backend and simply offer the sample frontend components for free?

MY VISION OF THE FUTURE

While there is a lot more to uncover, the basic building blocks are there. Headless is made possible via a GraphQL API and to a minimum, extensions should hook into GraphQL to add their own query and mutation endpoints. This is the bare minimum. An added value would be to help a system integrator build a PWA by supplying real or sample components in either Vue or React. But that's also where the uncertainty lies for extension providers: Is that worth the investment?

And it has to be said:

With PWA, the extension market will not be the same anymore. Instead of buying a product-slider on the Magento Marketplace, you might simply build it yourself using free NPM packages.

I believe, extensions will still provide added value though, so that you don't need to build all Lego stones (and corresponding API endpoints) yourself.

How will Magento extensions fit into PWA? I don't know. I have a few clues. But to the very least, extension vendors should join this exciting movement and help where they can. Because otherwise, they will not be there in the future.

Jisse Reitsma

The founder of Yireo, extension developer and trainer. Since the arrival of PWA within the Magento ecosystem, he has been pushing and promoting the new technologies. Yireo organized a Reacticon conference twice in 2018. Jisse has also been working on various hybrid solutions, is building his own React shop proof-of-concept and has been training developers in the usage of React, Redux, Apollo, GraphQL and soon also Vue and VueStoreFront.

Marcell Kiss-Toth, Dave Littlechild

END OF LIFE

The discontinuation of Magento 1 comes with an array of issues for existing Magento merchants. As a result of this change your current e-commerce software will suffer a few obstructions.

TIME FOR CHANGE

Firstly, there will no longer be any development patches for security releases. Clients on M1 will be reliant upon third parties or general forums to fix and discover potential vulnerabilities. In addition, there will be no fixes for third-party plugins. Secondly, there will only be third-party libraries such as Encrypt used to encrypt data. The latest versions of these third-party libraries may not be compatible with M1 thus increasing the risk of being hacked. Additionally, M1 supports up to PHP 7.2x, which itself is EOF by December 2020. From that moment, the core code may need significant effort to enable it to work with 7.3, which has an end of life in December 2022. Ultimately, M2 will allow your business to run 20% faster, offer a streamlined checkout process, feature an improved administrator interface and will be mobile-friendly. The time to upgrade to Magento 2 is now, and we're here to make the migration as seamless as possible.

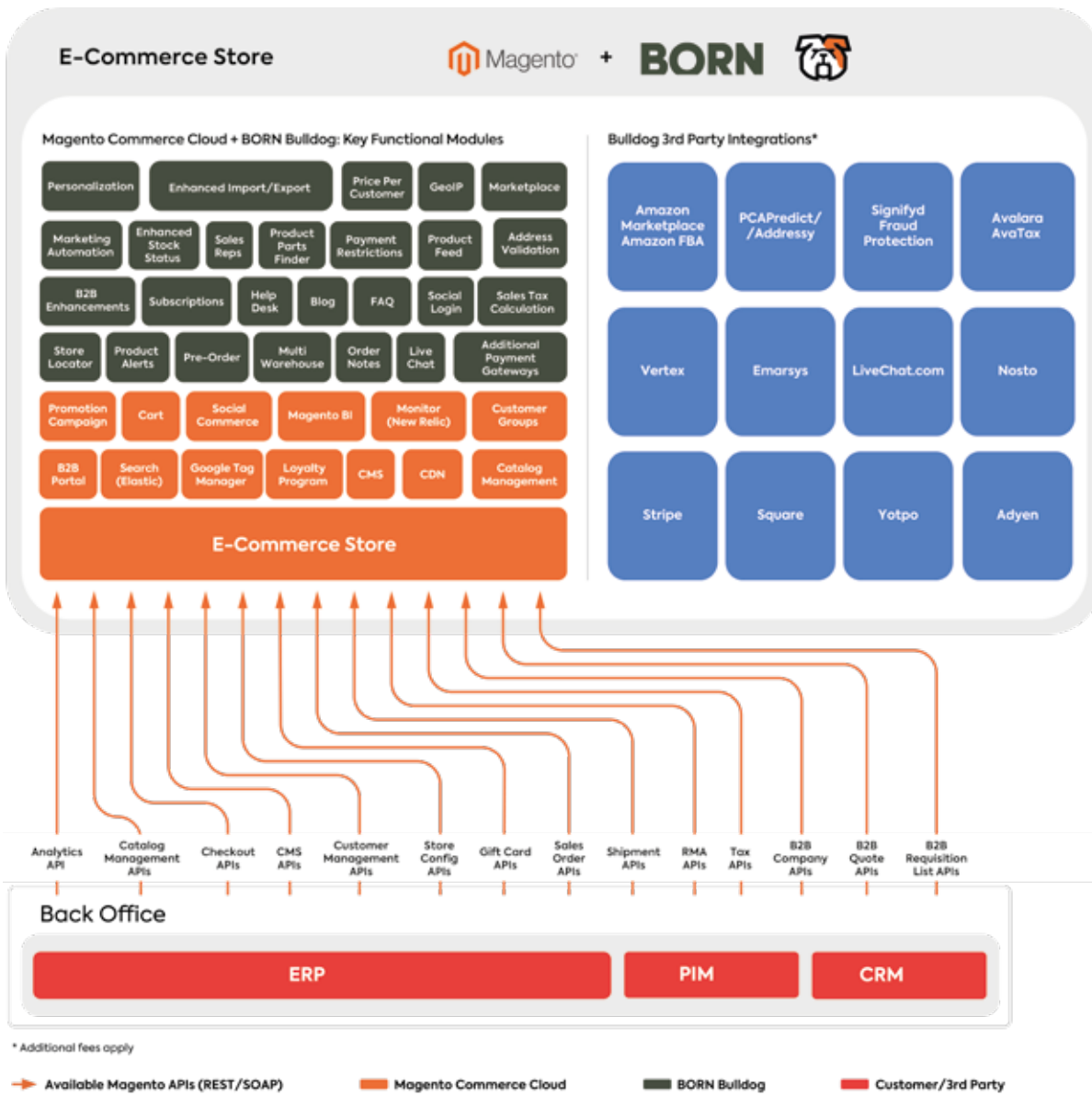
While this process may seem daunting, an accelerator is the key to making a quick, quality and seamless transition. Marcell Kiss-Toth, Global Magento Practice Head at BORN, talks about the advantages of utilizing an accelerator for merchants contemplating migration

The benefits of using an accelerator are twofold: the first would be to improve time to market, and secondly, it significantly reduces implementation costs.

The pre-built framework and connectors within an accelerator include necessary integrations for Magento 2 and have been tested and configured to a tee. It's important to note that all of the additional functionalities and services made available in an accelerator have all been sourced on behalf of the client. For example, instead of having to consider and speak to various email marketing providers, an accelerator will already include it along with a discounted rate. Data migration is another point of contention that would be streamlined with the use of an import/export extension made readily available through an accelerator. Additionally, universal features that are required for migration are already installed. Not only does an accelerator save time and costs, but it also reduces the overall risk to your project.

YOUR RELIABLE PARTNER

BULLDOG, BORN's Magento 2 certified accelerator, debuted in the Spring of 2018 at Magento Imagine. Since then, BORN has successfully completed



12 implementations for an array of enterprise clients across many verticals. With BULLDOG, you can launch your B2B or B2C ecommerce shop with rich features, a low budget and a short time frame, all without sacrificing quality. BULLDOG gives a completely new look and feel to the standard Magento accelerator as it includes pre-coded extensions, custom UX & UI features, and special enhancements to essential Magento 2 functionalities.

The image on the previous page depicts a partial architecture diagram of BULLDOG and is color-coded into four sections; the orange section is representative of Magento 2's out of the box functionalities. While everything in grey on top of Magento is a functional area of BULLDOG, this includes store locator, pre-order, subscriptions etc. In blue, are pre-integrated functionalities. These third-party integrations cover areas such as personalization, payment gateways, third-party logistics, and more. While these are its preferred partners here, BORN is always flexible in accommodating client preferences. Everything depicted in red, is the customer's internal or legacy system. This could entail an ERP, CRM, PIM, data warehouse, or anything of that nature. The arrows depicted above the legacy systems are out-of-the-box APIs that connect to the upper diagram. Ultimately, each component works together seamlessly eliminating the possible concern of conflict within the code base.

BORN understands that its clients have made customizations in M1 and aren't interested in starting over. BULLDOG doesn't force you to customize Magento because with the accelerator you automatically get the premium features that you had to customize in Magento 1. For example, store locator isn't included in Magento, but every single retailer (who wanted this feature) on this platform had to develop their own in-house store locator or they had to contract a third party. With Magento 2 and BULLDOG, you won't have to go through the same painful process as they compiled a carefully curated list of added value functionalities on top of core Magento, that merchants can utilize from day one, avoiding the customization process completely.

Emarsys is BULLDOG's preferred marketing automation integration. Emarsys is a leading global provider of marketing automation software and the first marketing cloud for retail and e-commerce. The Emarsys marketing platform enables true, one-to-one interactions between marketers and consumers across all channels, building loyalty, enriching the customer journey, and increasing revenue. Machine learning and data science fuels customer intelligence in an intuitive, cloud-based platform, enabling companies to scale marketing decisions and actions far beyond human capabilities.

The pre-installed Emarsys/Magento connector on BULLDOG connects your store and automatically syncs customers, events, products, and orders to Emarsys. This allows you to use Magento data in Emarsys for creating smart contact segments, personalizing your messages, building event-based programs, and automating retention marketing. The Web Extend functionality, part of the connector, allows you to deliver personalized product recommendations both on your website and in emails or to track revenue from your campaigns.

All-in-all, BULLDOG paired with the pre-installed Emarsys/Magento connector is your ticket to a seamless M1-to-M2 migration along with a lasting, flexible and powerful end result.

Marcell Kiss-Toth

BORN Global Magento Practice Head

As a seasoned professional, with 12 years' industry experience, Marcell joined BORN in 2017 as Global Magento Practice Head. He has extensive knowledge of agile software development and e-commerce technologies and has a proven track record across multiple verticals.

Dave Littlechild

Global Head of Partners and Alliances

With a 20 year career spanning London, New York and Sydney, Dave has been at the forefront of digital marketing and ecommerce globally for the past 15 years.

A proactive involvement over the years in the scope and development of marketing technology, Dave has also helped inspire countless marketers succeed in their goals, reduce costs and develop enormous growth in ROI.

He presently takes the reigns of the global partnership team for Emarsys, enabling marketers around the world to deliver truly personalized interactions.

PWA & MAGENTO:

Shane Osbourne

The 3 Key considerations for developers

Last year I gave a number of talks, all around Europe, and all around the same topic... PWA's!

Last year, I also promised I wouldn't be taking the stage again this year, giving the same talks around the same topics. But alas, a few months ago I broke this promise when I presented my talk at Meet Magento UK.

WHY?!

Because I'm still the first front-end developer to have led the successful deployment of a PWA on Magento PWA Studio, and I want this to change. It seems that Magento developers are currently in a flux which I like to refer to as; "The great Magento Delay". There seems to be much apprehension in the community surrounding PWA's, and a question that still remains: Has PWA now become the default choice for new projects? I asked the audience this question on the day, and around 80% of the developers in the room responded with the fact that they were still mostly working with Magento 1 & Magento 2.

So why is there a reluctance to push PWA's? Afterall, they are the future of eCommerce as we know it. Going from M1 to PWA is a huge paradigm shift. But it's not like we haven't made these huge leaps before. The dark days, as I like to refer to them, when migration began from Magento 1 to Magento 2, seemed at the time like a pivotal change. M2 was different in every way, there was uncertainty amongst the community then, and many developers seemed to be in the mindset that consisted of waiting for another developer to try it first! It feels to me that the same is happening with PWA Studio today. I for one believe this is the wrong mindset, and I want to help people get out of it!

If you're a developer reading this, you'll know that you learn the most when you solve a bug in production, or where you dig through someone's code base and you find out how the system works...that's when you do all your learning and it's worth a thousand blog posts!

As Magento developers, we should be striving to constantly innovate and deliver the best possible outcomes for our clients. As an early adopter of PWA Studio, I want to make the shift towards PWA's less daunting for other developers.

I've seen the common pitfalls and know the potential challenges you may face, so here are 3 key considerations you should be thinking about before embarking on a PWA project:

KNOW WHICH CLIENTS ARE SUITABLE

Do you have a project where you're migrating form M1 – M2? The M2 front-end is such a huge step anyway that you may as well go straight to PWA. So these types of projects are the ones where you should be considering whether a PWA is a viable opportunity.

HOW IS IT GOING TO AFFECT YOUR DEV TEAM

So, you've got your suitable client ready to sign on the line. Now you need to think about your dev team, because there's going to be some big changes!

The biggest question is: What is the role of a back-end dev in a PWA environment? Are they less or more important?

There's going to be heavy interaction between both teams, which might be quite different to how you currently work. The biggest shift in mindset that's needed is the processes – in a headless environment, back-end is driven by what the front-end need.

Back-end will still be looking into 3rd party modules and API's, data migrations, third party services etc, this won't change! But front-end developers will drive the development all in the name of performance and reducing HTTP requests.

The bigger change therefore comes for front-end teams, who are now directing what the back-end build, you may therefore need to consider expanding your front-end team.

LEARN FROM THE EARLY ADOPTERS

Implement browser-based UI testing.

Straight away, trust me on this! It's not the coolest thing to talk about, but you need to have ultimate confidence with the PWA studio framework.

If you're doing only mostly unit tests, you're doing it wrong! Most of your tests should include your browser clicking around a site. You need to be implementing this from the beginning.

SO, WHO IS THE IDEAL CLIENT?

I've categorised clients based on what I've personally experienced and assigned a tick or a cross as to whether they're ideal clients for PWA:

× **Admin XML:** They want to go through every minute detail and look at what they can tweak and want ultimate control. Avoid them like the plague!

✓ **Content-heavy websites:** Heavy use of CMS pages, CMS blocks, integrated blog...there's no problem with this!

✓ **Giant catalogues:** Tons of searchable products, Needn't be an issue! Infact, PWA is the perfect solution. The worst case scenario is you have a M2 extension that's making your database lock up because Google bot has come through and hit your category layer and nav which happens in production a lot. If you move to a PWA architecture, you're not going to have those issues.

? **Third party modules:** Those clients where the site feels like a lego block of modules! Using search, mail, reviews and everything else. This is the tricky one. It's a big MAYBE! If the module works in the traditional M2 extension way where it likes to modify and take over a piece of the page completely. Does it want to add a little JavaScript to the page, listening to and altering functionality... forget about it! If it's just back-end functionality that the modules are affecting, such as; email, SMS, shipping, PSP...no problem! They key point here is that it's not simple to just say or no to third party modules, investigate what your clients requirements are.

× **Precious about hosting:** This is most probably a no! If this client wants the cheapest hosting provider and they're not willing to budge on adding layers of complexity to the stack. Hosting a PWA is not rocket science, but it is a few more layers than what a client is used to compared to a M2 store.

Decide on server side rendering (SSR) early.

You do not want to have to retrofit this on to an existing app if you need it, so understand if you need this early on as it's going to affect performance and your front-end routine. If your website has content which you want to make available immediately, or you have time sensitive content, you probably will need to so some kind of SSR. And you'll need user testing for it. I'm not too proud to say I once had a checkout go down for 2 hours because I hadn't tested the rendering on the page!

Develop a strong versioning strategy.

This is the key to deployment. Imagine you're navigating through a site and somebody pushes out a critical bug fix. Since a PWA behaves more like a native application it means users will not automatically receive the new HTML that contains updated links to resources like Javascript files as they would in a traditional M2 website. The solution then is to version the entire PWA along with the service worker in a way that allows a hand-shake to occur at various times to ensure users are always running the most up-to-date version.

Don't invalidate the entire cache for every deploy.

On first use, a PWA may use a Service Worker to pre-load all of the assets needed to power the application – very similar to what happens when you download an app from the App Store or Play Store. It's a great way to improve performance and helps to ensure that simple connectivity problems don't prevent subsequent pages from loading (as the code is already on the device). It can result in a large number of files being cached on each user's device though, so you'll need to check that your build process and

your Service Worker configuration allows for incremental updates. Otherwise when you make changes to 1 area of the PWA, you could end up throwing away all the un-affected files as well which would result in the user having to effectively re-download the entire PWA all over again. You don't want to re-fetch all of the code for the Checkout just because you changed a feature on the Homepage, for example.

Load features on demand.

Luckily the old days of generating a single bundle of code that's capable of powering an entire website are long gone. Loading just the code relevant to the area that the user is currently accessing is now an accepted best-practice. This means as a user navigates through an e-commerce website, the code for the Product detail page is never loaded or executed until they actually get to that page. We consider this to be the absolute baseline for a PWA – lazily adding JavaScript, CSS, images & data is not even up for consideration – it's the foundation of a performant application on the Web. We can take this one step further though – by also deferring in-page features until the user requests them. The best example being a high-quality image zoom feature on a Product page. The hi-res images would naturally not be loaded until the user clicks the zoom icon – but what about the Javascript & CSS that powers the actual zoom functionality? This is what we mean when we say 'Load features on demand' – there's no need to even consider the Javascript or CSS for that zoomer until requested – especially since many people may never actually click the zoom icon anyway.

Other examples would be pieces of UI that are displayed in modals or that only appear after a user interaction – by excluding this code from the initial payload, you have the opportunity to further improve performance of all pages!

Shane Osbourne

The Lead Front-end Developer at JH. He is also the creator of the popular & open-source project Browsersync, an instructor on Egghead, and is very much focussed on improving the 2 areas of frontend development he finds most interesting - the overall performance of the Magento Platform and the industry's inevitable shift to PWA's

Tomek Strózik, Artur Krasiński

Story of creating a Progressive Web App prototype in 5 weeks

At Strix we have completed several large e-commerce projects based on Magento. We have carried them out in a standard model of web application implementation. Sometimes, however, there is an opportunity to work in unusual conditions. For example, when a client comes to us who is looking for a partner to implement a prototype of Progressive Web App... only in 5 weeks. Sounds extreme?

A CHALLENGE TAILORED TO STRIX

Our client was looking for a technological partner to build a prototype of Progressive Web App based on the latest frontend technologies. Everything that has been described until now doesn't go beyond the standard work that we do every day. After all, web applications and building e-commerce is our bread and butter. What is more, our developers have extensive experience in this area. Therefore, usually we don't have much fear of taking on a given assignment.

So what was special about this project? The answer is simple: TIME. In this case, the time spent on the implementation of the project was only 5 weeks and not a single day longer.

Every client wants the implementation of his project o be as short as possible, but creating a prototype of the Progressive Web App is a real feat. It became even more meaningful for us when we combined the short delivery time with the requirements that the application had to meet.

What was the creation of a PWA prototype like in practice? Below we describe everything in detail.

GENERAL ASSUMPTIONS OF THE PROJECT

The aim of the project was to build a prototype application that would show the typical customer journey in an e-commerce system. The key emphasis was put on the user experience and the technologies used. The client wanted the prototype to use the latest web technologies available on the market, and also presented the opportunities that modern browsers and mobile devices give us. The prepared application had to provide the user with feelings as close as possible to those of using native iOS/ Android applications.

A big challenge, and at the same time the biggest advantage of building a PWA prototype, was the freedom of the technologies used. We were able to prove ourselves in creating a product based on the latest web trends. This resulted in full motivation and job satisfaction. The results we achieved overnight were very good. With time, thanks to the used technological stack, the introduction of further functions was becoming easier and smoother. Each team member was able to find himself in the application's code and use his colleague's work. This translated into the effectiveness of work and the efficiency of implemented solutions.

Arkadiusz Przybylski
Front-end Developer at Strix

In addition, the application was suppose to use the native capabilities of mobile devices such as a camera and push notifications. All this to encourage system users to perform specific actions desired from a business point of view. High performance and speed of the prototype with partial possibility of using the content in offline mode were also important. At the same time, the application should maintain all the designed advertising and business functions and meet the requirements of search engine optimization (SEO).

SELECTION OF TOOLS AND TECHNOLOGIES

When we created the work schedule, we realized how ambitious our project was. The general assumptions, the list of tasks, and finally the time of implementation made us realize that the priority issue will be to select the right tools and technologies that will enable us to achieve the goal and create an MVP (minimum valuable product).

We decided to base the whole application on single file Vue components that contained JavaScript code, HTML and styles. The state of the user interface was managed using Flux architecture, which assumes data flow in only one direction (the central object supplies data to all components displaying the interface). This resulted in a high degree of component flexibility and a high degree of application modularization. Additionally, we used Nuxt, a toolkit (Webpack + babel, Vue router, Vue server renderer), whose main task was to support rendering of user interface modules on the server side and support for quick prototyping of functionality.

We wanted to create a prototype application that would run smoothly and quickly (like a native mobile application). At the same time, an important aspect was the flexibility and architecture of the written code. Our main goal was to create uncomplicated components realizing a single responsibility. Components created in this way were combined through "wrappers" into components realizing more complex business logic. This allowed us to make quick modifications when the initial assumptions were changed (we often redesigned the functions overnight to ensure the best possible experience while using the application).

THE TEAM AND THE ESSENCE OF SCRUM

The team that implemented the Progressive Web App prototype consisted of: four JavaScript developers, two UX designers and one project manager. In addition, we provided consultations with other development departments, such as backend developers.

The work of our implementation teams is based on agile methodologies (Scrum). Despite the extremely short implementation time, in this project we also decided to use the available Scrum techniques. This

approach provided a comfortable working environment for the development team because they worked in a familiar environment.

The short lead time of the prototype made it impossible for us to implement the textbook Scrum. All events such as planning, daily stand-up, retrospective or sprint review were shortened to a minimum so that the time spent on using Agile techniques was adequate to the prototype completion time. Additionally, thanks to this methodology, the client monitored the progress of the project on an ongoing basis, worked closely with the team and got to know the working culture of our company in practice.

KICKOFF MEETING

Kickoff meeting is an important event at the start of every project. With such a meeting we also started working on the PWA prototype. During the meeting, the team got acquainted with the business needs of the client, the purpose of the application and the main assumptions of the project.

Due to the short time of prototype realization (5 weeks!), after a general introduction and discussion of the project, the meeting turned into planning the whole implementation, the first sprint (which started at the same time) and the preparation of a framework work schedule, which the same day was to go to the product owner.

IMPLEMENTATION PROCESS

The sprint duration was set for a week. Thanks to this, we divided the implementation into several stages and monitored the progress of work on an ongoing basis. The first sprint for developers was used as a time to design an application architecture, search for appropriate plugins, configure development and production environments and plan the work. This allowed us to run both the UX team and the implementation team in parallel. The assumption in the designers' work was to prepare for the implementation of specific parts of the application and a description of functions from the user's point of view. Short lead time forced the development team to divide each stage into two additional parts:

1. **implementation of the function**, during which the programmers created the software architecture and programmed the appropriate functions available in the application without ready graphic designs,
2. **implementation of designed views**, during which we introduced styles and graphics into previously prepared functional components.

The foundations for the work of programmers in the first part of each stage were mock-ups provided by designers at the beginning of the sprint, which contained the outline and conventional elements of individual functions. This allowed the user interface to be divided into appropriate modules and components.

DESIGNING UX

Step one: Personas

The tight schedule of work meant that we were not able to conduct interviews with users. It is on their basis that we usually build persona, i.e. a reflection of typical customers and users. Therefore, we had to develop Proto-Personas based on the data available in the network and the experience of the team of designers. It is worth mentioning that in the past we implemented projects in the industry, which is concerned with the prototype of the application being created.

RIGHT TOOLS

Based on our experience and analysis of the tools available on the market, **we chose the Vue framework** and the tools related to its ecosystem. Why? Framework Vue is characterized by several important things, these are:

- **High performance and component rendering.**
- **Extensive and solidly prepared technical documentation**, which allows you to quickly assimilate knowledge and efficiently solve technical problems encountered during the implementation of the project.
- **Intuitive syntax with low technology input** and high code writing performance.
- **Small baseline size of the framework (~60 kb).**
- **Active and vibrant community centered around Vue.**

Step two: information architecture

We have analyzed the existing assortment in the current system and the customer's product catalogues to better understand the goods sold. Then, based on personas and product analysis, we made a list of functions that will be necessary for the customers of our customer's shop. On the basis of the collected data, we started designing the information architecture.

Time played a key role in this project. We were not able to work out a service information architecture together with the customer's users, nor to evaluate the results of our work. We had to rely on the experience of the design team and proposed the best, according to the team, information architecture and product categorization. We suggested to the client that in the near future we should conduct a user survey for the classification of information on the website using the method of card sorting and subsequent clusters analysis of collected data. The aim of the research would be to verify the solution prepared by us and to introduce possible modifications.

Eryk Ulewicz – ***Head of UX at Strix***

Step three: designing the application

After all the above mentioned preparations, we could finally start designing the application. Each of the designers prepared individual sketches of the shopping path, on the basis of which the team together with the developers worked out a common solution. Then we started to design low and high detail mock-ups, corridor tests to create a graphical prototype of the application.

The next iterations included further functions of the application. In a very short time we worked out many more screens and functions than we had previously assumed in POC (Proof of Concept). Our work resulted in 137 paper sketches, 69 application screens and a graphic prototype.

PROPOSED SOLUTIONS

The prototype of the application, which we have prepared, showed the possibilities (at the moment) offered by PWA. They are very similar to native iOS/Android applications (about 95% coverage). The technologies, tools and designed architecture used by us give a number of benefits in the form:

- **flexible solution** – one team of developers creates an application for all devices (desktop/mobile),
- **easy development of applications** – architecture based on micro-services and micro-components,
- **cloud ready** – the tools used support solutions for continuous integration and delivery in the cloud (Continuous Integration and Delivery in the Cloud),
- **access to applications and offline content,**
- **SEO optimisation.**

COMMUNICATION AS A WAY TO SUCCEED

What made us successful? First of all, communication was important. We consulted all the application elements between the development team and the design team. The developed solution was presented to Product Owner for evaluation and acceptance. Each sprint ended with a clear division of responsibilities and an estimated deadline for implementation. Thanks to this, each person in the team knew what element they were responsible for and how long they had to complete their tasks. Efficient communication enabled a perfect flow of knowledge between the team members, which in turn allowed for a quick reaction and possible corrections of technical and design decisions.

There are cases in which experience, commitment, passion and only 5 weeks are enough to complete a project.

Secondly, the cooperation and openness between the team members was an important aspect. We discussed all encountered problems (technical and design) and solved them at the daily meetings of the team. This way we could quickly eliminate obstacles.

All these things allowed us to achieve the project objectives and meet the deadline.

VALUES FOR THE CUSTOMER

What did the client achieve by working with us? In addition to receiving the Progressive Web App prototype, he could see exactly how our implementation teams work, what we put emphasis on, what our technical and design skills are, and how we respond to changes and feedback during the project.

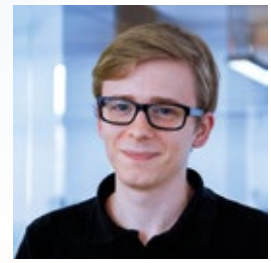
The target product will be much more complicated because due to the short delivery time and the nature of the assignment, the following could not be realized: unit tests of the application, connection of the created application with the proper API, preparation of the proper production environment, issues of content regionalisation and placing complex marketing content (integration with an external content management system).

However, it should be noted that the client's approach, i.e. ordering a prototype of the application he wants to implement in his systems, allowed him to check how we actually design applications and how we implement them. Such empirical data are much more valuable than references, a review of the portfolio of completed implementations, or a conversation with the sales department.

WAS IT WORTH TAKING UP THE CHALLENGE?

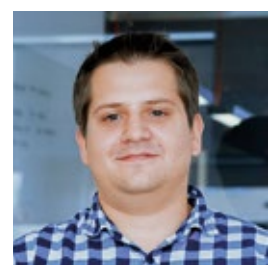
Definitely yes. Preparation of the Progressive Web App prototype brought us a lot of interesting experiences. Within 5 weeks our team had to solve many problems quickly and efficiently. In this way, we gained valuable experience related to teamwork, working with the client and the implementation of projects in a different mode than usual.

We also found out that testing a specific solution, tool or idea does not require creating complicated models combined with time-consuming planning. There are cases in which experience, commitment, passion and only 5 weeks are enough to complete a project.



Tomek Strózik
Front-end Developer at Strix

He has 5 years of front-end development experience. He is a certified Magento Front-end Developer who strongly believes, that sharing knowledge is the best way of learning. In Strix, he took part in several projects including e-commerce storefronts, PWA and internal products built on top of Magento, Vue.js and Nuxt.js. Currently, he works on improving storefronts using modern JavaScript, GraphQL and Flux architecture.

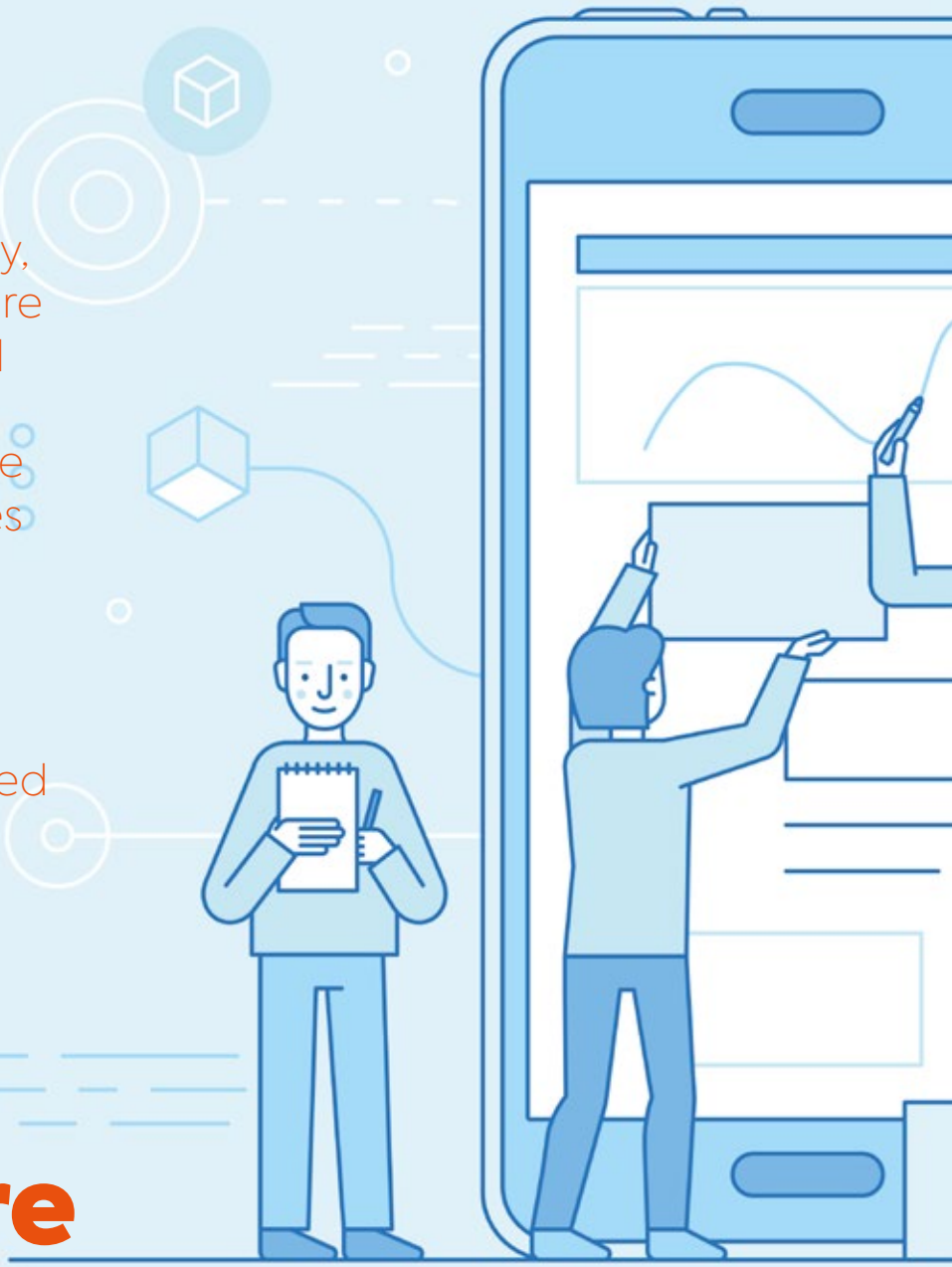


Artur Krasiński
Head of Front-end development at Strix

For over 4 years he has been working in one of the largest companies offering e-commerce implementations in Poland. He is a certified Magento Developer. He supervises the work of several developers in the implementation team.

Ilja Lapkovskis

The buzz surrounding Progressive Web Applications is growing by the day, with more and more merchants around the world seeking to capitalize on the novel opportunities offered by a new age of web development. Magento Imagine 2019 was dominated by talk of PWAs and it looks like e-commerce is about to enter a new era. But...



Why are PWAs taking so long to enter the Magento mainstream?

But why is it taking PWAs so long to enter the Magento mainstream? What is happening behind the scenes? Challenges. Challenges on every step of the way. Some of these challenges are faced by every emerging technology; others are specific for PWAs. In this article, we invite you to take a candid look behind the curtain to understand at least some of the reasons why PWA adoption is disproportionate to the hype.

WHERE ARE ALL THE MAGENTO PWA STORES?

The hype surrounding Progressive Web Applications truly hit the Magento community around the time of the Magento PWA Studio release in January 2019. And yet, half-a-year later, there are relatively very few live projects, even though the hype hasn't died down. Why? What is stopping merchants from taking advantage of PWA storefronts? To understand this, take a moment to appreciate the complexity of PWAs. It's monumental. Being a compilation of a wide variety of technologies, with some of them continuously evolving further, PWAs deliver a level of complexity rarely encountered by Magento developers.

Developers must learn a lot of new things, which also are constantly changing, along with the nascent PWA best practices, resulting in a uniquely demanding environment.

Magento knows this full-well. Their PWA initiative is developer-first, with PWA Studio being a suite of tools for arming developers with the necessary instruments and knowledge for creating PWA solutions. However, even Magento's solution is in a shifting state, due to perpetual updates and changes, which are to be expected given the novelty of it.

These things – the huge complexity of PWAs, complemented by the volatility of present solutions is extremely discouraging, despite the potential game-changing e-commerce benefits, and it's no wonder adoption is cautious. It's likely to continue in the same manner, unless a stable solution is found around which merchants and developers can rally.

Our ScandiPWA team sought to create something far more accessible to merchants and developers alike – a PWA theme. Install it, edit it, and you're good to go! Or, at least, that's the idea. A single Magento developer should be able to set it up in a day, and it's open-source, which means it's extendable to your heart's desire. Being an exclusively Magento theme, we chose the same tech stack Magento uses to be on the same page as the community.

However, as a solution tightly coupled with Magento, ScandiPWA doesn't escape the rollercoaster of updates. While having a theme-like structure might make it easier for developers to get started, it doesn't necessarily solve the second part of the problem – the volatility. In the next section, let's take a look at a concrete example.

WHEN BEING AHEAD DOESN'T PAY OFF...

ScandiPWA's development began half-a-year before the release of Magento PWA Studio. We had a bonkers fast demo store up and running by the time of the January release. Creating our own PWA theme meant developing a plethora of solutions absolutely from scratch and building ScandiPWA from the ground up. This, in turn, meant that we tended to develop things before Magento introduces them, which turned into a backward challenge of keeping up with the latest changes.

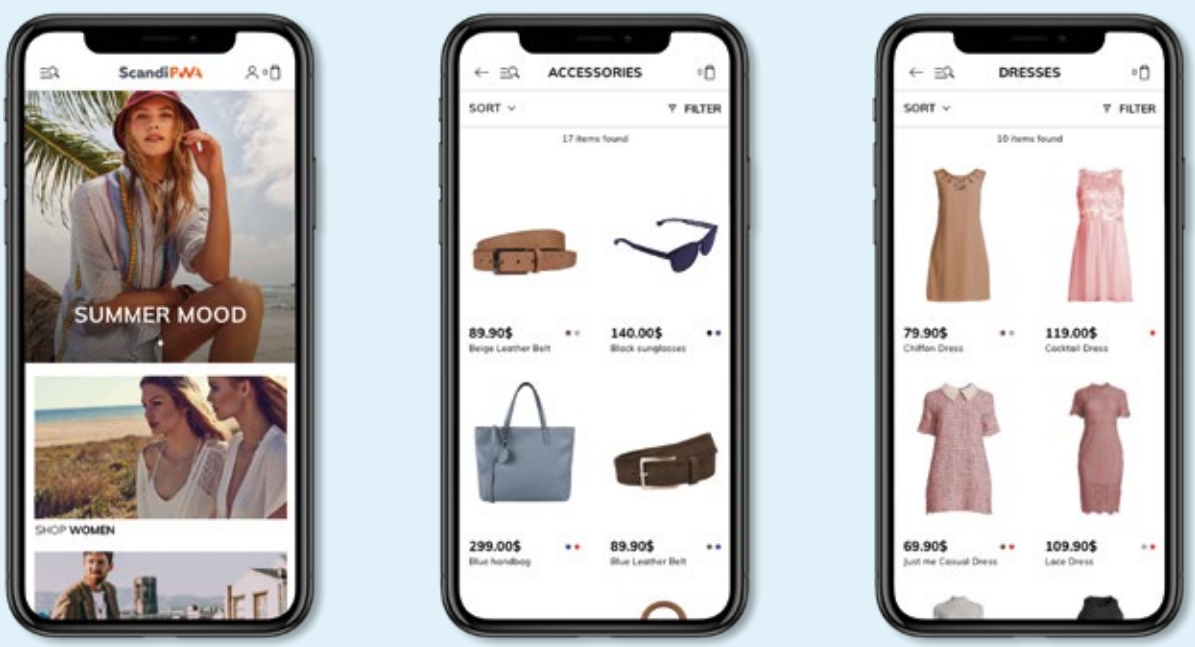
Solutions and decisions introduced by Magento often conflict with already developed ones by ScandiPWA. In the spirit of keeping true to Magento's way of doing things and ensure stellar performance, ScandiPWA often needs to return to already developed solutions and rework them.

For instance, along with version 2.3.2, Magento released their caching layer, which, while similar, had some crucial differences from ScandiPWA's solution, that took 2 whole weeks to adapt. Naturally, some of the introduced changes were welcome, whereas others were more limiting, such as the GET request size.

Being strictly based on Magento, it's important to adapt from a design and structure perspective. As a tightly integrated theme, it's important to avoid heavy customization and to make the most from the resulting benefits in terms of performance and ease of use, despite the high price paid for being such.

IN A RAPIDLY CHANGING ENVIRONMENT, COMMUNICATION IS THE KEY

In a time where PWA development is speeding full steam ahead, it's not enough to create cool things. It's important to share them with the community, and this importance is aggrandized when you're an open-source project like ScandiPWA. Someone always needs your



Scandi PWA Demo, source: https://demo.scandipwa.com

PWAs are cutting-edge technology and have issues with robots, SEO, analytics, indexing, and everything else. ***That's not true.***

help and someone always misses the latest updates even when crucial information is broadcasted through every communication channel available.

The difference-maker is that people still love to communicate with people, instead of machines.

Talking with someone who faced similar issues as you seems to generally be preferred over digging through mountains of documentation. And this is fantastic for any open-source project. Why? Because it's the most valuable kind of feedback on your project from a developer perspective – understanding where the issues lie. Nevertheless, some feedback is more valuable, some – less. Answering questions about the same issues over and over gets old fast, and when you've exhausted all your communication channels and the same questions still arise, a solution needs to be found.

In our case, the best solution we've discovered are short-format videos. Explaining a concept or an issue and how to resolve it in 10–15 minutes, doing it clearly and empathetically, is proving itself to be a powerful avenue of communication. It capitalizes on the human touch sought after by the community while alleviating the issue of repetitive explanation on the ScandiPWA developer side. What's more, it's as valued externally, as it's embraced internally.

HYPE LEADS TO MYTHS, MYTHS LEAD TO MISUNDERSTANDINGS

Given the excitement surrounding PWAs, it's surprising how misunderstood they are. Whether we're putting that on PWA marketers, or a lack of accessible information, in either case, the result is the same – buzzwords, myths, and misconceptions. Here are some encountered on a common basis.

New technology = new issues

PWAs are cutting-edge technology and have issues with robots, SEO, analytics, indexing, and everything else. That's not true. PWAs are sometimes informally referred to as SPAs (Single Page Applications) on steroids. SPAs came on the scene over half a decade ago and were marred by the same technical and indexing worries people have about PWAs. The architecture between the two is very, very similar, and a lot of issues people bring up were already solved a long time ago with SPAs and the solutions apply to PWAs. The wheel need not be reinvented.

Ilja Lapkovskis
CTO, Scandiweb

An infrastructure engineer, Magento back-end guru, and public speaker, Ilja's refined analytical skills, holistic vision, and extensive technical knowledge are continuously applied to create future-oriented eCommerce experiences. Ilja is the chief driving force behind the ScandiPWA project, seeking to make PWA accessible for Magento merchants worldwide.

100x faster!

What's more, people often cite performance improvements as being a technical marvel achieved by PWAs, throwing numbers around saying PWAs will increase your load speeds 5x, 20x, 100x. As mentioned earlier, PWAs are an amalgamation of modern technologies, a lot of which, when optimized, do produce improvements in performance, through new data-loading and data-caching techniques. Technically speaking, the chief advantage PWAs bring in terms of performance improvements is the control we, as developers, have over the data sets requested. We don't have to wait for pages to be loaded by the server; we can get a bunch of data and display it immediately, get another bunch – display, another bunch – display.

The real power of PWA performance improvements comes in terms of visual information delivery. The focus is on how to load things to ensure the best subjective experience, so it's more about UX and UI than purely technology. From smart data delivery to leveraging placeholders for seamless image loads, the feeling of speed can bring far more benefits in terms of performance. Of course, there are measurable technical improvements, but be wary when you hear "100x faster" and other buzzwords thrown around.

I can browse the internet offline!

Offline mode is another commonly misunderstood buzzword. No, PWAs won't deliver a full offline experience, as soon as you visit the website and turn off your internet. It is supposed to, but with a bunch of limitations. "Offline mode" uses smart caching to deliver uninterrupted user experiences even when the internet goes down, or the network coverage is spotty, but it's not an alternative to a full-blown native app with a pre-downloaded catalog.

When it comes to offline e-commerce, the controversial thing is the checkout. We're in love with online payments, and while, in theory, it could be possible to permit request queues and filling out the checkout form, there are a deluge of issues with this, starting from people adding items that might be now out of stock, to platform limitations with, for example, iOS is still limiting developers in terms of background or offline features. Perhaps we should simply redefine "online" and "offline", or just switch from "offline-first" to a less confusing term.

FINAL REMARKS

It's not all doom & gloom. While adoption might sometimes feel slow, and challenges – plentiful, we, as a Magento PWA community, are surely moving in the right direction. It's important to keep foundational issues in mind, not spread misinformation, and work together, because PWAs most certainly will and already are redefining what e-commerce is, can be, and will be.

PWAs are here to stay. Challenges bring solutions. Progress is measured by the day. Despite the outlined volatility, the issues, the misconceptions, PWA projects for Magento are being developed and deployed as we speak – adoption IS happening and it's picking up speed. In fact, agencies are already finding themselves strained for resources with the influx of projects. And it's the early adopters that have the advantage. While you'll be implementing your first PWA solution, they will have optimized theirs.

As with all new projects, problems, conflicts, bugs are inevitable, and the single solution merchants, extension developers, and others can rally around is still a ways off. However, if you're willing to patiently wait for this solution, expect to be playing catch-up with your more proactive competition when it finally rolls around.

DON'T MISS THE BOAT

Magento – as all companies and projects – has a history full of ups and downs. Currently we are on a new path creating a more stable push towards more and more "ups" for everyone. Empowering the global community and commerce ecosystem through open collaboration and education has always been our primary goal which is now formalized through the association. Now we need members to help us further this vision.

So, you've actually spent some money to buy (Meet) Magento event tickets; you've spent some money to travel to those events, maybe even to book accommodation, and you've spent a full working day listening to all of the wonderful things the Magento Community has to offer you and your business.

And now you're reading this second edition of the Magezine, because you want to keep yourself up-to-date with the latest developments that may have an impact on your career or your business.

Just the fact that you've attended Magento conference and/or reading this column means that you have a stake in the success of the Magento ecosystem. For most of you, Magento is the core – or at least a vital part – of your (or your clients') commerce system. So maybe you don't know it yet, but this means that the newly formed Magento Association is going to be very important for you.

As with many other IT projects, things went up and down in Magento's past. Magento had an amazing kickstart by the founders Roy and Yoav, it mainly went downhill with eBay and euh... X.commerce... and the sky turned blue again when Permira and Mark Lavelle took the helm. And now Adobe is taking the lead in Magento's second decade. The fact is that many of your businesses and maybe even your professional career to a large extent relies on the success of both the software, the brand, and the ecosystem that is Magento.

Guido Jansen

Psychologists and Usability Expert focused on e-commerce and working for Vaimo. He helps companies to build international customer journey optimization teams and to build a culture around experimentation and validation. Since 2008 his work included many Magento projects, he is a Magento Master and is part of the Board of Directors for the international Magento Association. He is often invited to speak at international events about persuasion, e-commerce, conversion optimization and experimentation cultures.

Overall, things right now are pretty ok but that doesn't mean that we don't have our challenges. In both the Magento core and the ecosystem layers around it, a lot of things can be improved, and – if left unchecked – might unnecessarily impact your bottom line in a negative way. Just as a single example: think about all of the merchants that are still on Magento 1. We not only have a responsibility to take care of them, but moving them to Magento 2 or finding a way to support them after Magento 1's End of Life might simply be very good for the business.

We've started a committee around this, and together we want to figure out how we can turn all those Magento 1 merchants into a strength for our community. And besides internal factors, we are also in an industry where competitors and consumers are rapidly developing and changing. The market right now is very different from 2008 when I started working with Magento and will be very different in another 10 years.

Now, don't get me wrong: this is not a doom and gloom co-lumn. We got many things right to get where we are today. Most software doesn't last over a decade, so that's already a proof that a lot of things went very right.

Magento is no longer a company but is now a product in the Adobe ecosystem. And I think that especially with Adobe very exciting things can and will happen. But as an open source community, we can't keep looking at the big almighty company to have things fixed for us. As a collective, we all have a responsibility to keep our ecosystem thriving.

With the Magento Association, we have laid the foundation for these community initiatives. Together, with input from a lot of community members, we have set a vision for our ecosystem that I think will resonate with many of you. We believe in an open, healthy and powerful Magento ecosystem, in which we empower each other and work towards continued success of future generations. This means developers, merchants, consultants, users, agencies, partners and maybe even long-lost psychologists like me ;)

We will advance and empower you through open collaboration, education and thought leadership. We will work on strengthening and raising our shared foundation so that we can take on the rest of the world.

As a Magento business and a Magento professional, you simply can't afford to miss the boat on this one. We need to take care of Magento, our shared core of commerce. You need to be part of this from the get-go, use your vote as a member and you will be able to influence where the association spends its time and money. And as a bonus, you will be able to start or join a committee on a topic important to you, have extra visibility towards Adobe and get discounts on events, training and certifications. So, what are you waiting for? Get your phones out and register!

We opened up membership last April at Imagine and we have now seated the first committees and announced our first title partners. Individuals can join through magentoassociation.org/join, and you can contact us if you are interested in business partnerships. The whole board is available for any of your questions and we'll be very happy to tell you all about how the association can facilitate you. Your professional success depends on the success of the whole Magento ecosystem. In the open source community, it's our collective responsibility to be better and I have no doubt that we can do that, and we believe the Magento Association is the best vehicle to carry us forward.

Join us.

We are extremely excited for what's to come and we look forward to all your ideas and contributions.

WHAT MAGENTO MEANS TO YOU

Ignacio Riesco

I didn't have the chance to attend the last Imagine, but I watched all of the online sessions from the event and kept refreshing twitter feed almost constantly. Let me be honest – I was a little bit worry about avoiding the word Magento during the event. Maybe it's a coincidence, maybe it's an announcement of something bigger – erasing the word Magento and replacing it with another, e.g. Adobe Commerce. It's just a word, don't make a big deal out of it – you can think. Yes, it's true – it's just a word. But for me, it means a lot.

I devoted ten years of my professional career to Magento and this name is of great value to me, personally. Then I thought to myself that there were more people like me. That's how I came up with the idea of a series of articles with different people in which they share their experiences with Magento and what that word means to them.

And then Magezine appeared on the market, which gave me the space to express my thoughts.

So, here we are! I hope you'll enjoy this text! Who knows, maybe it'll inspire you to think about the whole issue. I really want you to share with the readers and whole community what Magento means to you – in your personal and professional live, and to indicate the moments in your careers that made Magento special.

LET'S BEGIN WITH A LITTLE BIT OF HISTORY AND CONTEXT

I may sound strange, but I fell in love with Magento, even though it wasn't love at first sight. And it started with this:

With my company, we were at Magento Developers Paradise in Ibiza (2011). This event opened my eyes to the importance of Open Source and the developing community behind Magento. All those brilliant minds gathered in one place, focused on a common goal – how to improve Magento and make it even better.

We participated in Xcomerce Conference in San Francisco (2011) and I adored its spirit since day one. But we missed the first Imagine in LA. When I followed it from Twitter I said to myself: this is it! I made the decision of attending next edition, and I kept my word! So, we went to Imagine and we met a lot of brilliant people there.

But that moment, which I might call a breakthrough, was the first edition of Meet Magento Spain (March 2014). It was like a dream come true — a Magento conference, organized by us and full of Magento enthusiasts from around the world. It was an excellent event with international line-up and awesome conversations afterwards. I think that many things that are happening now on the e-commerce market had their beginning on this event.

MAGENTO MEANS...

If we wanted to stick to the book definition, we could say that Magento is an open-source product, it's the name of company that owns the product (now Adobe), and it's the community (in my opinion the strongest and the most brilliant on the planet).

But this definition is devoid of emotions. And yet for me and for many of us, this word means much, much more. That's why the feeling that this word is going to be erased scares me so much.

LOTS OF DIFFERENT (PERSONAL) MEANINGS

- **Magento means growth.** From 2 employees to more than 50 people working in the company in three different locations. More than 150 people have worked in interactiv4 since the beginning, and most of them are still associated with Magento. It's not even just about how the company's size increases, it's about the space for professional and personal development that everyone here gets. We even took the first tagline of Magento as a statement: A platform for growth.

- **Magento means improvement.** I am pleased to see that my colleagues are improving their quality of life. I believe that Magento gives them this opportunity. Working with so many creative people is not only a chance to learn from their knowledge and experience, but also a motivation to raise the standards in own professional live. Achieved successes also result in greater satisfaction with life in general – after all, these two worlds are not completely separate.

- **Magento means traveling the world.** Magento is a worldwide community. New York, Amsterdam or Krakow – you name it. I have had the amazing opportunity to travel to many places in the last ten years. They say that travel broadens the mind – it is one thing to get to know new countries. But the second thing, perhaps even more important, is the opportunity to talk to experts from completely different markets and exchange experiences. Meeting new people and then maintaining these relations through participation in events is an unquestionable perk of such travel.

- **Magento means evolving and overcoming difficulties.** It is about finding opportunities where others see problems. I remember how much trouble we had to face when we were working with M2 first releases. It was a long-announced and very unstable those

days. Such experiences teach us the most – we struggled so hard with it, but we've learned from our mistakes and drawn conclusions. It made us better and stronger in the end.

- **Magento means transformation.** The e-commerce market has evolved a lot since I started my adventure with Magento. Consumer behavior is changing, customers are changing, and merchants' expectations are changing. I am glad that despite this, our customers stay with us. It strikes me always when I observe how many stores have already trusted Magento and thanks to that they can grow and develop their business.

- **Magento means continuous learning.** Learning from good and bad practices, learning from own experiences and not making the same mistakes are very important lessons I took in Magento. I also love the fact that members of the community are always keen to explain how they solved specific problems and share their advice and tips.

- **Magento means friendship.** Obviously I have friends from Magento (I call them MageFriends) and they are an essential part of my life. They gave me the opportunity to learn from them and I'm so grateful for that. Most of them have a similar lifestyle to mine – we are connected by work, industry, but also by the perception of the world. In this way, they better understand the circumstances that may arise before us.

- **Magento means flexibility and complexity.** There comes a moment when you realize that you can build everything with Magento and you begin to think that it is more a framework than a platform. There is also a point where you can transform your business from merely a development company into a Magento consulting and development company, which is a completely different thing.

- **Magento means success.** Magento is in the middle of the chain of dependencies. It is stable and reliable, so it guarantees merchants higher sales. Thanks to this, the hosting companies can earn more from online hosting stores. Companies are able to develop more extensions to enrich your e-commerce, or e.g. improve your search results by using better search engines. All of this results in the fact that shipping and payment providers make transactions every second. Magento stands for the entire business ecosystem.

- **Magento also means celebration.** Because there's no success without celebrating, and we have plenty of reasons to celebrate what we have achieved in the last ten years. But remember – work hard, party harder!

- **Last but not least. Magento means you!** If there were no people involved, none of the above-mentioned examples would mean anything. The more we are, the stronger we become. So, let's be open to new faces, new energy, and fresh ideas.

CONCLUSION

Magento has become a huge and significant part of my life. Apart from the fact that thanks to it I can feed my kids, I can constantly broaden my knowledge about business, finances and even human resourcing. Magento is built and driven by smart and awesome people, and I'm so happy to be a part of it. Long-live Magento!

Ignacio Riesco

CEO and founding partner of Interactiv4, Magento Enterprise Solution Partner since 2010, and recently Awarded at Imagine 2018. Ignacio is Computer Engineer, and he has more than 20 years of experience in eCommerce. He is a very active member of the Magento Community since 2010. He is responsible for organizing and promote many Magento events such as MMES, MMNY, Madrid Magento MeetUp, PreImagine, Contribution Weekend Madrid and PreMagentoLive.

IN THIS ARTICLE YOU'LL FIND OUT:

- What are the differences between the APIs for checkout functionalities
- About dependencies between modules
- How an alternative checkout will look like in the scope of services decomposition

Checkout functionality is one of the most important parts of any e-commerce platform. At Magento, we want to provide our customers with the best possible shopping experience, so we are working to improve the whole process.

ALTERNATIVE CHECKOUT FLOW

Yevhen Sentiabov, Alex Paliarush

OVERVIEW

The Magento checkout has a lot of features like:

- One-page checkout with support for different shipping carriers (including UPS, USPS, FedEx, DHL)
- Multi-address checkout
- Instant Purchase with the ability to place an order from a product page
- Multiple payments integrations like PayPal, Braintree, Authorize.net, Cybersource, etc.
- Vault tokenization
- Different security features like 3D-Secure verification, anti-fraud protection

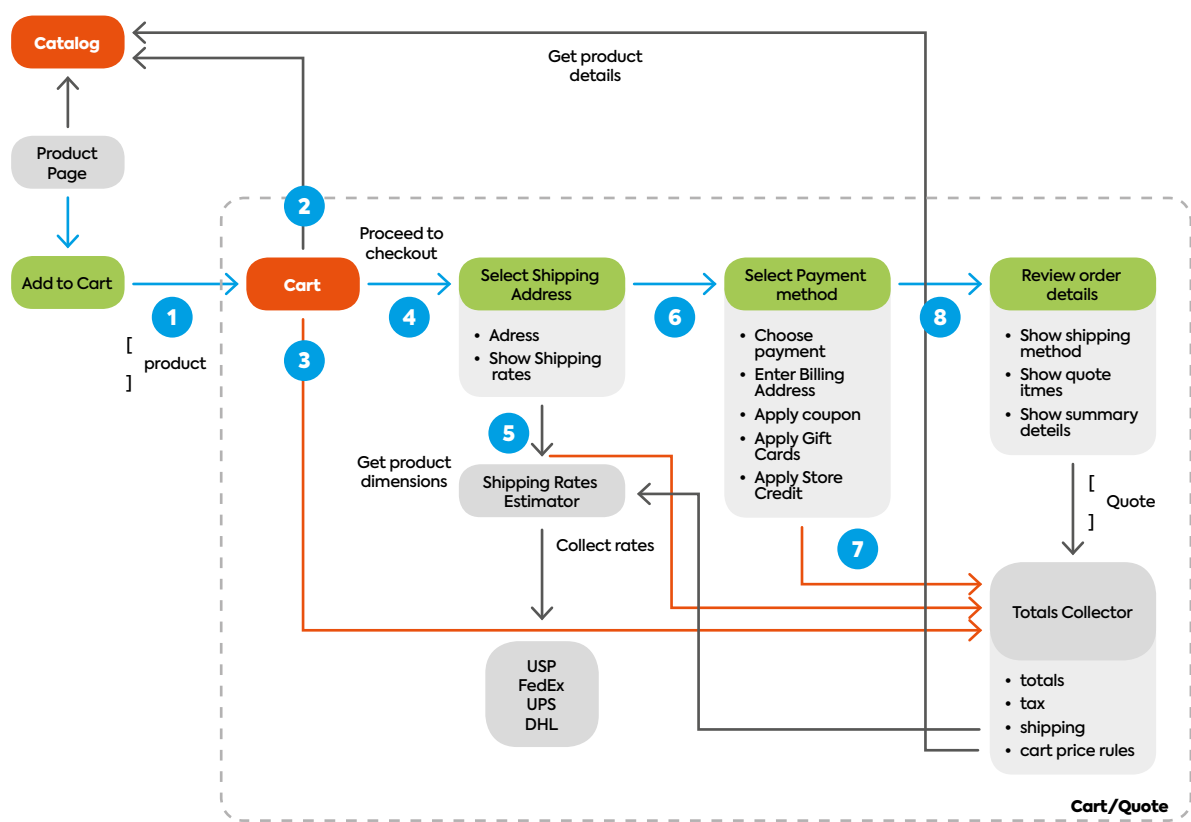
Currently, Magento provides two main APIs for checkout functionality: REST API and GraphQL. GraphQL has been under development since 2.3.0, and its capabilities have increased with each release. In fact, it's become the main API to support PWA functionality and other custom storefronts, although REST APIs can still be used. But in most cases, both REST and GraphQL APIs use the same backend service contracts. These service contracts do not always satisfy all requirements and sometimes perform suboptimally. It's not always possible to improve existing backend service contracts in a backward-compatible way.

In the scope of the Service Isolation project, the Magento team will be improving different areas, including checkout. We intend to introduce an alternative API for checkout in Magento 2.4.x that includes improvements for business scenarios, service contracts, performance, customizability, and expose it via GraphQL API while supporting the existing one.

CURRENT CHECKOUT

Let's review the current checkout flow and communication between components in Magento.

Curent checkout flow and communication between components in Magento



1. An item is added to the cart.
2. The cart calls the catalog product's repository to get product details.
3. The view cart action calls a totals collector to estimate totals on the quote.
4. "Proceed to Checkout" starts a one-page checkout flow, where a customer can specify shipping and billing addresses, choose a payment solution and place an order.
5. When the customer enters or selects the shipping address, shipping rates are estimated, which triggers whole totals calculation.
6. On the Select Payment Method step, the customer can specify a billing address, choose a payment method, and apply discounts and gift cards.
7. Actions like applying a customer balance, discounts, gift cards trigger the recalculation of the totals.
8. The payment review step also contains summary details like items, shipping method, totals.

The current Quote API has dependencies to the Cart implementation. In general, there is no clear separation between Quote and Cart for developers. Even ***\Magento\Quote\Model\Quote*** implements ***\Magento\Quote\Api\Data\CartInterface*** on the service contract level, so it's difficult to define how to use such service contracts. At the same time, ***\Magento\Checkout\Model\Cart\CartInterface*** has methods to operate with the quote entity. As you can see, there are circular dependencies between these two entities.

The Cart entity should contain only items but in the current implementation, it has addresses, payment details, discounts, customer balance, price adjustments, etc. It is mixed with the Quote entity. You cannot use Cart without Quote, and you cannot use Quote without Cart. There is no opportunity to substitute one of these with a 3rd-part integration or introduce a custom checkout flow.

The Add to Cart operation receives only a product ID and the quantity. All additional details, such as weight and product dimensions needed to calculate shipping rates, come from requests to the Catalog.

Most actions, like page reloading, retrieving shipping rates, applying discounts, gift cards, etc. trigger the recalculation of the whole cart/quote. The quote calculation is an expensive operation because it triggers all calculations for product prices, shipping rates (which are calculated for all available shipping carriers), cart prices rules, and tax calculations. The different quote's calculators, like Cart Price Rule calculator, change quote object and totals and behavior might be unpredictable as different calculators can operate with the same data.

Despite a lot of features and possibilities for customization and extension, the current checkout implementation has multiple areas which can be improved further.

PERFORMANCE AND PAGE LOADING

A number of JavaScript components slow down page loading, and this has a huge impact on checkout performance. The first load of the Checkout page (without compiled static files and generated cache) on a vanilla Magento installation contains 286 different HTTP requests.

Checkout has a well-defined project structure for JavaScript components separated into actions, models, and views. But at the same time, the number of components slows down the page loading and decreases checkout performance. Another issue, the current Storefront based on the combination of XML configuration with a complicated structure, knockout.js components, templates, and jQuery. It's not a trivial task to replace a storefront implementation by something more performant one like React or even use all the modern features of JavaScript.

MULTI-ADDRESS CHECKOUT FLOW IS SEPARATE FROM ONE-PAGE CHECKOUT

The difference between the multi-address checkout and one-page checkout is huge. The multi-address checkout does not support all features supported by one-page checkout. The UI is not so friendly, as it is based on phml views, which are hard to customize.

We added ***\Magento\Multishipping\Model\Checkout\Type\Multishipping\PlaceOrderInterface*** to support multiple orders (order for each address) with online payment methods, but each payment method should provide its own implementation for placing orders. A list of payment methods is limited, and payment integration should have Vault Tokenization (http://tiny.cc/w35mbz) support to be integrated with multi-address checkout.

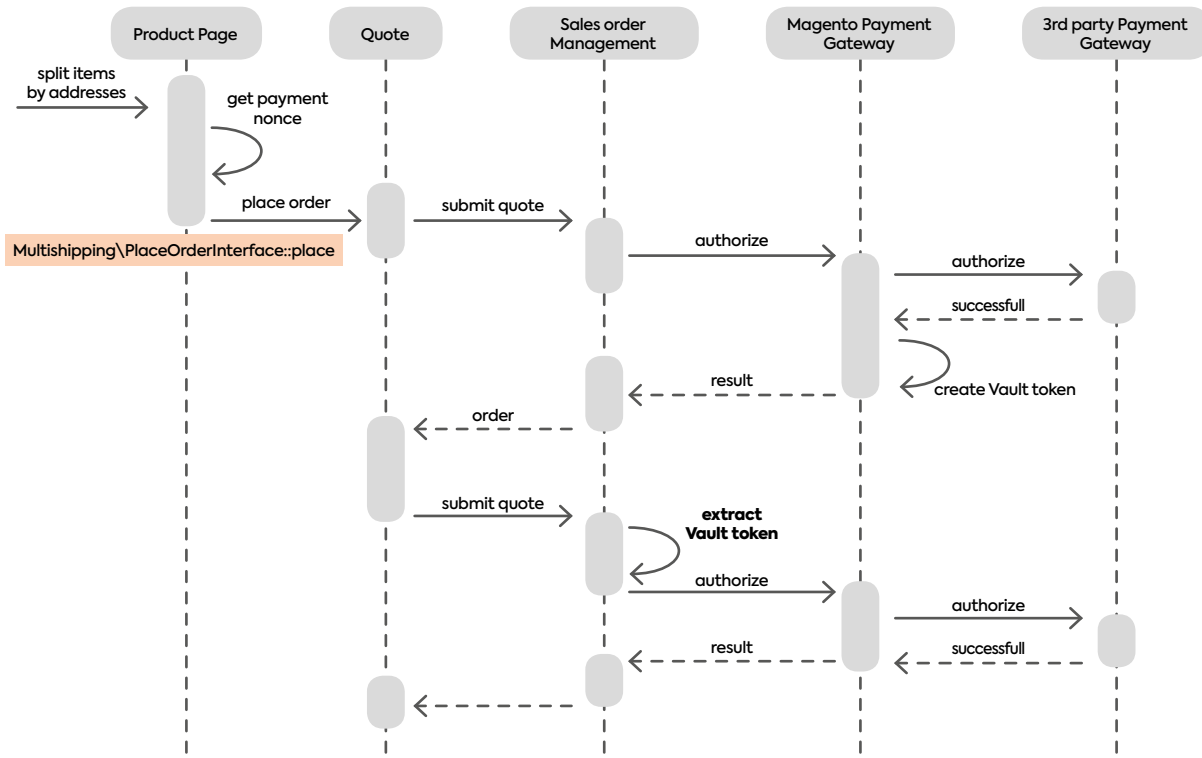
Another problem related to the quote entity itself: Magento does not support multiple quotes at the same time, and multi-address checkout is based on shipping assignments for the same quote. There are multiple usages of magic ***\Magento\Quote\Model\Quote:: getIsMultiShipping ()*** method to identify current checkout flow.

DEPENDENCIES BETWEEN MODULES

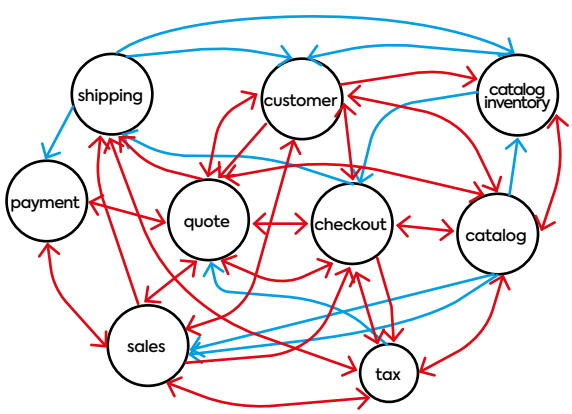
The current implementation of Magento operates on entities like Cart, Quote, Payments, etc. As previously mentioned, Cart and Quote, are tightly coupled together and depend on each other. Also, a customer always has an active quote even if he does not have any products in the cart.

Bi-directional dependencies between components, in most cases, require changes in both. Such dependencies make development, support and debugging more complicated, as shown in the following dependencies graph for Checkout components.

Multi-address checkout



Checkout Dependences Graph



The blue arrows show unidirectional dependencies, the red ones are circular dependencies between modules. We can see that most of the dependencies are circular. Also, this graph does not include implicit dependencies, which are hidden via modules communication. Different modules have redundant knowledge about other modules, and changes in the one module might affect the behavior of another module.

COMPLICATED CHECKOUT CUSTOMIZATION

The current checkout implementation has a lot of drawbacks for extensions and customizations. Developers need to go through great effort to implement a small customization for an existing checkout. The frontend logic is tightly coupled with backend logic.

Let's consider a small example. Suppose we want to customize the shipping rates output on the checkout page by adding the estimated delivery date. Currently, developers can use multiple approaches on how to build frontend customizations via REST API or GraphQL. GraphQL is easier to extend as you can add additional fields to the schema with custom resolvers. REST APIs require more modifications, because you can't change a service contract in runtime. Unfortunately, GraphQL does not solve all issues for our example because each GraphQL resolver uses existing service contracts. To extract additional shipping rate details to the presentation layer, we need to dive much deeper into the existing implementation and refactor it.

The *\Magento\Quote\Model\Quote\Address\Rate:: importShippingRate ()* method has a list of hardcoded fields that is not easy to extend:

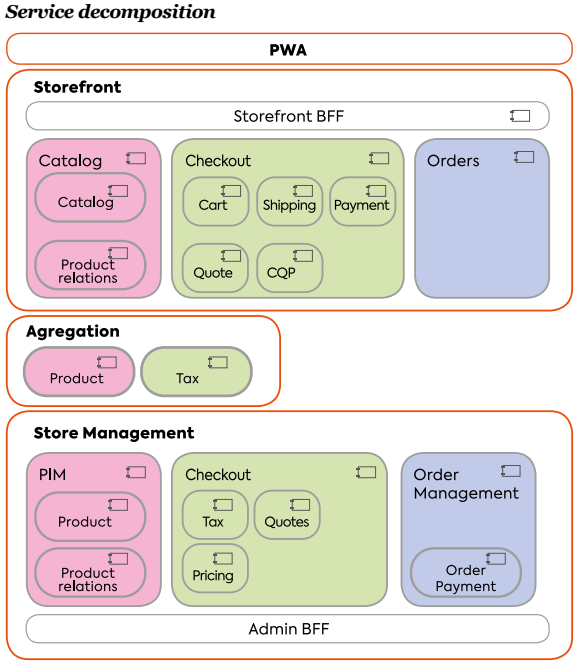
```
public function importShippingRate(AbstractResult $rate)
{
   $this->setCode($rate->getCarrier().'_'.$rate->getMethod())
       ->setCarrier($rate->getCarrier())
       ->setCarrierTitle($rate->getCarrierTitle())
       ->setMethod($rate->getMethod())
       ->setMethodTitle($rate->getMethodTitle())
       ->setMethodDescription($rate->getMethodDescription())
        ->setPrice($rate->getPrice());
   return $this;
}
```

This is just the first step in adding a single field for the shipping rates on the checkout page. In general, developers must know how to work with Magento extension attributes, di.xml, layout inheritance (for standard checkout UI), and use inheritance just to add one output field.

PROPOSED SOLUTION

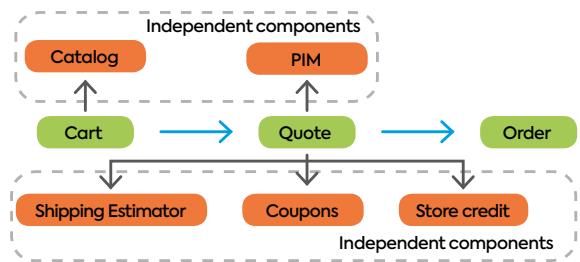
In the scope of the Service Isolation project, the Magento team is going to decompose existing modules and components, define clear API and bounded contexts, split up a monolithic application to Storefront and Store Management applications, and make components replaceable. The Service Isolation Vision (http://tiny.cc/945mbz) describes how the components will be separated and moved to different applications. Let's consider how an alternative checkout will look like in the scope of services decomposition.

Service decomposition



The solution assumes that only GraphQL APIs will be exposed to the storefront. Components like Cart, Quote, Shipping will have clear boundaries and their own APIs. This approach allows us to resolve and reduce the list of dependencies between modules. A quote can be created based on different factors like shipping address (each quote will have own shipping address), different product physical stores, and so on.

Data flow



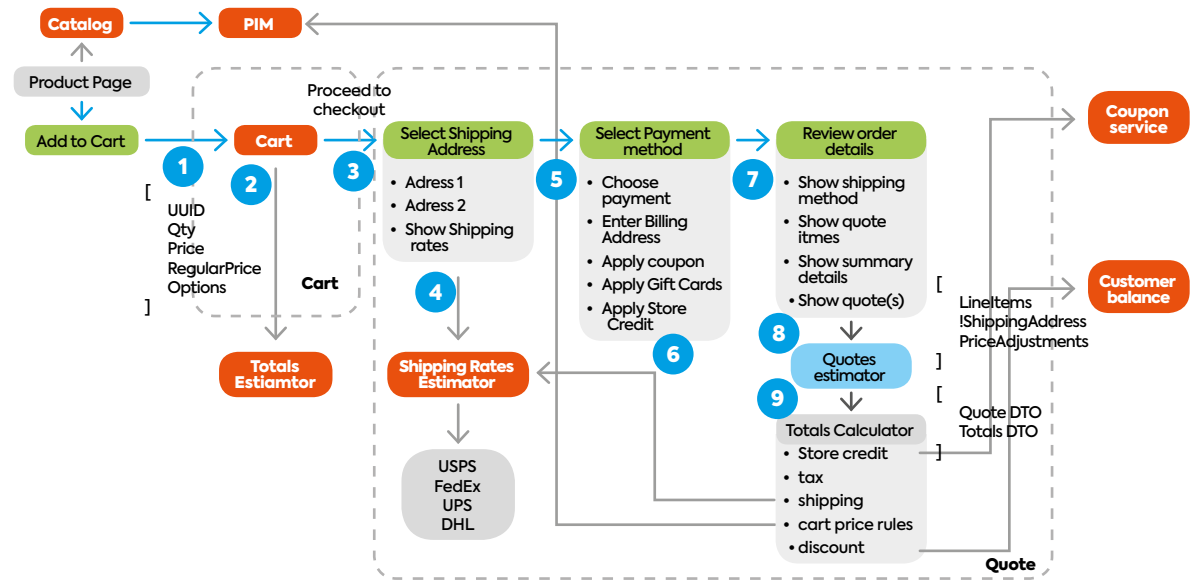
The Cart will be separated from the Quote and will have its own API. The Cart will depend on Catalog. Quote will have knowledge about PIM, the Shipping Rates estimator, Coupons, Customer balance, and other services for totals calculation. The Cart will also provide data for Quote. The Quote will provide data for the Order. The Order will not have any knowledge about the Catalog, Quote, and Cart components (see graph on the right page: Alternative checkout flow).

The Quotes Estimator will be the main entry point for creating a quote based on the provided input, and the Totals Collector will provide totals calculation based on the provided quote object and the configuration. The Shipping Rates Estimator will be agnostic to the quote object and will provide shipping rates based on input data like shipping origin, shipping destination, and dimensions of the items. The unified input data would allow us to use the same Shipping Rates Estimator for RMA, order estimated delivery, etc. without modifications. Also, the estimator will support retrieving rates for a specified shipping method, which would reduce the number of calls to other shipping carriers (the current implementation gets rates from all configured carriers).

CHECKOUT API IMPROVEMENTS

The unified API like GraphQL allows the developer to build storefront applications on any suitable technical stack like React, Angular, etc. while reducing the list of current JavaScript components. Another benefit: GraphQL API allows

Alternative checkout flow



1. An item that contains all needed attributes (for example, UUID, quantity, regular price and options) is added to the cart.
2. The cart calls the totals estimator for basic items price calculation. This calculation is needed only for a summary representation in the shopping cart. It does not make sense to check prices from PIM or Catalog.
3. "Proceed to Checkout" triggers the creation of a Quote entity. It does not contain an estimate of the totals.
4. After the customer specifies a shipping adress, the shipping rates estimator provides all available shipping methods for this address. If a customer wants to specify multiple shipping addresses, then multiple quotes will be created (one per shipping address) and a list of available shipping methods will be provided for all addresses.
5. On the select payment method step, a customer chooses the payment method, specifies a billing address while Magento applies discounts and gift cards, calculates customer balance, etc.
6. After a payment method is chosen and all possible price adjustments are specified, the summary can be reloaded dynamically based on changes for each item.
7. The summary block includes quote items, shipping addresses and other quote/multi-quote details.
8. The change of any or all price adjustments triggers a new quote creation and totals recalculation.
9. The Quotes Estimator triggers totals calculation. Each calculator gets all the needed data, like the actual shipping rate, product prices, tax rules, the current state of customer balance, etc.

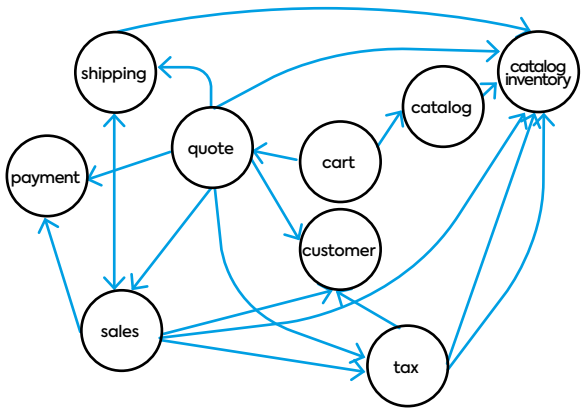
extending queries, so the same JavaScript component can retrieve more aggregated data via the same HTTP request.

A new strategy for a quote entity creation would allow supporting more scenarios out-of-box. For example, multi-address checkout can be built much easier and be more customizable, as quote entities will be created based on different factors. If a list of shipping addresses is provided, then each quote would be submitted as a separate order. In this case, we don't need to make additional customizations for payments integrations to support the custom multi-address checkout. Processing payments will be the same for both flows.

Improvements of quote component API would simplify the implementation of B2B scenarios like Requisition Lists and Negotiable Quotes. The quote component will be agnostic to the shopping cart representation, so 3rd-party integrations could provide their own cart implementation that includes B2C and B2B features. Different strategies to split up quotes would allow creating quote entities based on items to support Negotiable Quote functionality without overriding a lot of existing functionality. The totals calculations can be extended/re-used for different business scenarios.

A uni-directional checkout flow would reduce dependencies between components and increase application performance. The cart will not be re-calculated for each page load, only basic calculations will be performed for items in cart. The quote totals calculation can be grouped, like re-calculating all totals when all price adjustments are specified. Clear boundaries between components would allow re-using the components. For example, the Shipping Rates estimator can be used at least in three different scenarios without modification: estimate shipping rates for all available carriers, getting the actual shipping rate by shipping method, and handling RMAs.

Checkout Dependencies Graph



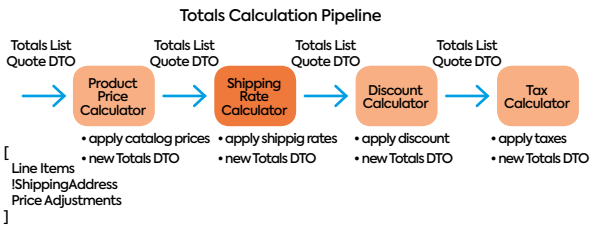
Immutable objects would allow a clearer code execution flow with easier support and debugging as well as automated testing. Using immutable objects would also allow these benefits:

- Introduce performance improvements like parallelization because the state of the original object will not be changed during execution. For example, the shipping rates estimation can be executed at the same time for all available shipping carriers.
- Fewer conflicts between components that use the same object. In the current implementation, plugins try to modify the same quote object.
- Operations will work with data instead of changing the state of objects.

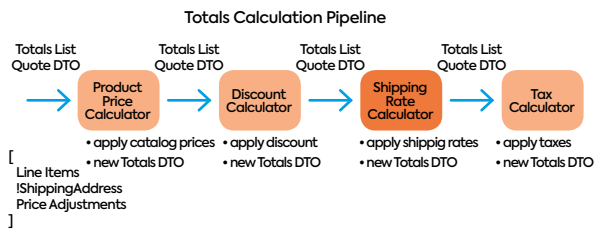
TOTALS CALCULATION IMPROVEMENTS

The current approach for quote calculation has multiple drawbacks. Like changes in the quote object, a quote totals collector is difficult to customize. It uses complicated logic to define the order of totals calculation (tax before/after discount, discount/tax rules for shipping, etc.), and it requires additional calls to 3rd-party systems to update shipping rate prices.

The proposed solution assumes that a quote will be an immutable object and will operate with a totals list. Each calculator will create a new totals object based on a totals list, and the order of each calculation can be changed in runtime. The list of calculators and their order can be visualized to better understand these calculations. The calculation of the totals might look like this:



Each calculator receives Quote DTO and a Totals List, calculates totals, creates a new Totals DTO with the calculated amount and adds it to Totals List. This approach does not change a quote object. It has a defined interface for totals and changing the order of calculation. Magento provides multiple configurations for changing the order of calculation. For example, discount can be applied before shipping amount or after.



Since the list of calculators and their order depend not only on configuration but also on factors like the presence of shipping address (virtual and downloadable products and gift cards do not require a shipping address) the calculations pipeline should be built-in runtime.

All totals will be persisted as a JSON structure into the database, which allows having dynamic totals structures for different entities like quote, order, invoice, and credit memo without increasing the list of database columns. For example, the current quote database table contains 28 fields related to different types of totals like price with/without tax, applied customer balance, amounts with/without base currency, etc. The new structure would allow storing all totals in one field. The currency exchange rate will be stored instead being duplicated in the amounts for the base and display currencies. Because all calculations are happening only in the base currency, the display currency is used for representing the amount on the storefront.

The amount values in TotalsList will be represented as integers instead of float values, thereby avoiding rounding issues like the one-cent issue while supporting zero-decimal currencies like the Japanese yen out-of-box. The TotalList will contain an amount in the store's base currency and the store view currency (display currency). The currency exchange rate is a float value. A storefront application will convert amount in the base currency to the store view specific currency and format.

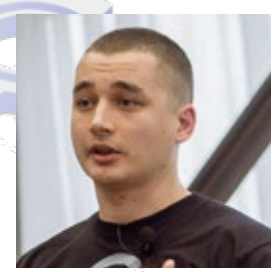
AN EXAMPLE OF TOTALS PRESENTATION:

```
{
    “amount”: 1002,
    “currency”: ”USD”,
    “display_currency”: ”EUR”,
    “currency_exchange_rate”:
    0.92,
    “is_applicable”: true,
    “code”: ”totals_list”,
    “totals”: [
        {
        “code”: ”subtotal”,
        “amount”: 902,
        “is_applicable”: true
        },
        {
        “code”: ”shipping”,
        “amount”: 100,
        “is_applicable”: true
        },
        {
        “code”: ”discount”,
        “amount”: 0,
        “is_applicable”: false
        },
        {
        “code”: ”grand_total”,
        “amount”: 1002,
        “is_applicable”: true
        }
    ]
}
```

CONCLUSION

The described vision introduces an alternative API in future Magento releases. It will be exposed to storefront applications as a GraphQL API. At the same time, the existing REST and GraphQL APIs will be preserved and supported. The proposed changes should not affect existing applications. The desired state: beginning no earlier than the 2.4.x release, Magento will provide separate GraphQL APIs for the existing checkout and for the new checkout. The existing API might be deprecated in the future. The main benefits of the proposed changes include:

- Clear boundaries between the Cart and Quote components. Separate APIs would allow to use them independently
- Multi-address checkout out-of-box, without the need to support two separate checkout flows
- Support of immutable multi-quotes allows more flow customizations like negotiable quotes and requisition lists
- A unified interface via TotalsListInterface for totals calculation
- Simplified storage of totals entity reduces the number of database columns
- Uni-directional flow reduces communication between components
- Quotes can be separated based on stock availability
- Improvements to API customizability and extensibility
- Performance improvements based on reducing dependencies between components and their communication.



Yevhen Sentiabov
Magento Architect

Magento 2 Architect. Engaged in WEB-development last 7 years, have been developing SAAS solutions, working on billing products for the VoIP market. I'm working in Magento last 4 years, during this time mostly focused on payment integrations, fraud management, sales, and checkout functionality. Now, as a part of Magento architect's team, I'm working on Services Isolation project and different area improvements. My hobbies are martial arts, airsoft and studying new technologies.

Alex Paliarush

Alex has been with Magento Commerce for 8+ years. During his service he was involved in development and architecture of web API frameworks including GraphQL for PWA storefronts, message queue, authorization framework and other components. He is currently a Magento 2 core architect and is interested in eCommerce and software architecture. His interest in developer experience led him to creation of one of the most popular devboxes for Magento 2 developers. Alex is a member of the advisory board for Magento 2 certifications.

ASYNCHRO MAGENTO

IN THIS ARTICLE YOU'LL FIND OUT:

- Why and how asynchronous communication was chosen in the Magento and which mechanisms Magento uses to achieve this?
- What is Event Driven Architecture, CQRS & Event Sourcing?
- What is Magento Queue Framework?

Oleksandr Lyzun

Over the years, Magento architecture has grown and become more complex. The main reason: merchants' requirements are becoming more complex. This complexity has led to an increase in application infrastructure, release cycles and the number of features that have to be maintained. For Magento's first decade, it was a monolith application which nicely covered mid-sized merchant needs. But as the system grows, the more difficult it becomes to maintain this monolith architecture.

NOUS

Beginning in 2018, the Magento architecture team has switched direction and is now trying to achieve a full-service decomposition of Magento Core. The idea is to split Magento Framework to separate services, where each service is fully isolated and independent and can be deployed as an independent application. One critical idea with this approach is the communication between services. Any given service must be able to seamlessly communicate with another service, or the monolith. Following one of the guidelines of Service Decomposition, the communication between services must be efficient and able to process asynchronously, if possible. This article is intended to give an understanding of why and how asynchronous communication was chosen and which mechanisms Magento uses to achieve this end.

LET'S HAVE A LOOK AT A COUPLE OF EXAMPLES HOW COMMUNICATION BETWEEN SERVICES CAN BE PROCESSED

It can be done in either a synchronous or an asynchronous manner. Synchronous communication means that all requests have to be processed in the same order that they were sent, and that the sender system must wait for a response from the receiver. This means the connection is kept open between the two services. If we are talking about a complex system with many services, the operations sequence may become quite long and lead to performance issues. Because of a high risk of potential process locks, such a system is very hard to maintain and scale. For example, after placing a new order, the system may have to execute a long series of post-actions:

First of all, to process all of these actions takes time and this means that the user experience will suffer because of long running operations. Secondly, the process requires that all operations should be completed without error. Otherwise, after waiting for an extended period of time, the user will receive an error. Long operations during checkout process are conversion killers; we should never give the buyers an opportunity to change their mind and cancel a purchase because of poor service. The main goal of any e-commerce system is to make the purchase process as easy as possible.

Thirdly, in the case when you are using the 3rd party systems, it may happen that an external system is not available or is having performance issues. This could mean that our purchase process would be fully blocked or it would take so long that customers might again cancel their purchase. As a result, the shop would be losing conversions every minute the system was not available or performing poorly.

With the same example, but asynchronously, we can charge the customer right away, place the order and process all required operations to complete the order by using background processes. We are giving the users a fast and easy purchase process, where they can place an order quickly with a minimum delay. This way, we are executing all required operations with background processes. In case of a problem with one of the operations, we can always report this information to the user or retry an operation on a background, on our end, if the problem was temporary. So, the only inconvenience we have left are failed operations which require customer input, but this can be solved with a clearly implemented error-handling process. Asynchronous communication will then look like that:

Service decomposition is only one, but currently the most common, example of where and how asynchronous communication can be used.

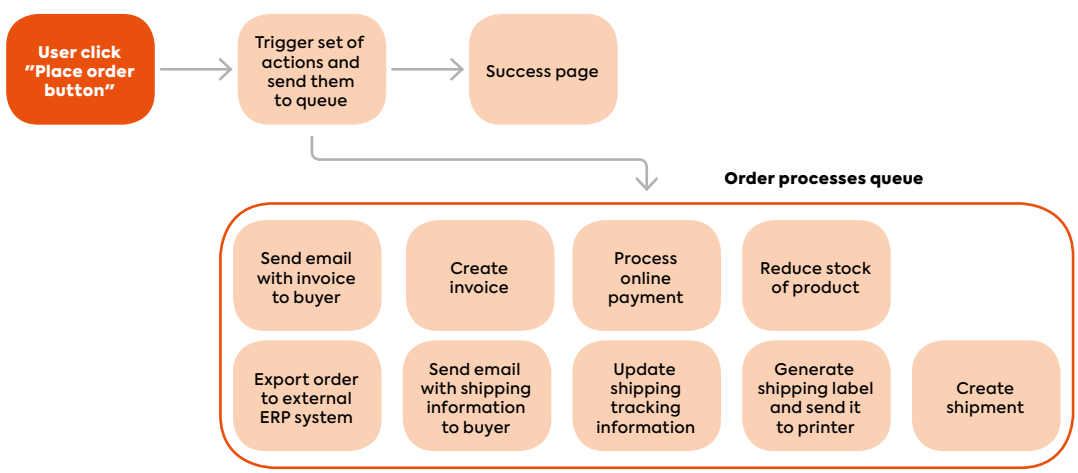
EVENT DRIVEN ARCHITECTURE, CQRS & EVENT SOURCING

Event Driven Architecture (EDA) is an architecture which is controlled by events. One of the main advantages of this approach is that we don't need to change the currently existing implementation (module) when we want to add or change the functionality of the system. When implemented properly, it will allow the user to add and remove modular functionality without affecting the stability of the system.

CQRS (Command Query Responsibility Segregation) – is an approach to separating database writes and reads. If we are talking about Magento, this means that commands are being sent to Magento but the result of these commands will never contain data from the objects you are communicating with.

On the other side we have queries that are getting information from the persistent storage. This may also lead to the situations where those two parts are not consistent, especially if we are talking about asynchronous communication where execution order may be different or may have some delays, depending on the current system loads.

Event Sourcing is the approach for storing the data whenever we make a change to the state of a system. We record that state change as an event, and we can confidently rebuild the system state by reprocessing the events at any time in the future. We will not go into too much detail about this topic, but we will try to tell about it within the scope of Magento.



The main ideas around Event Sourcing are:

- There are no limitations on the number of events that can happen with an object.
- All commands are immutable: state of events can never be changed after creation.

With Event Sourcing, you always know what happened with every object since its creation.

CQRS & Event Sourcing are usually tightly connected. And as we are talking about the full Service Decomposition in Magento, Event Sourcing currently looks like the best path. The aim of this is to change, correctly and efficiently, all communications between independent business processes. Every event that takes place in the system can have an impact on any number of different objects, depending on their purpose. So, Event Source architecture, if correctly built, will:

- give us control over all system processes,
- facilitate their planning and development,
- allow us to monitor and track the state of all objects and persist each write operation.

MESSAGE QUEUE FRAMEWORK

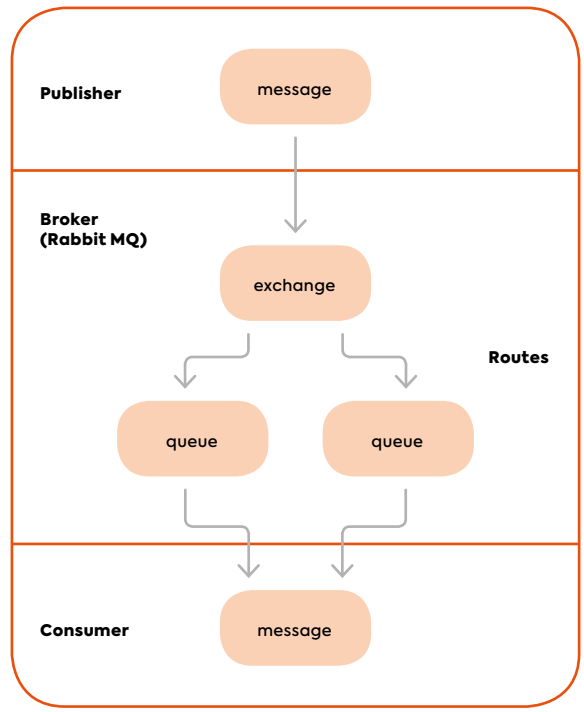
The main feature of Magento that allows asynchronous communication is the Message Queue Framework (MQF).

The main idea of Message Queues is to provide an asynchronous way of communication between the sender and the receiver, where they actually do not contact each other. In other terms – the sender places his message into the queue and the receiver will get the message only when it is processed by the consumer.

Starting from Magento Commerce version 2.1, out-of-box Magento has provided the MQF. The MQF is a system that allows modules to publish messages into the queues. Moreover, the MQF provides mechanisms for creating and managing consumers, which will take messages from the queue and will process them. The MQF comes with support for RabbitMQ queues. In addition, to give developers and merchants some flexibility in choosing servers requirements, the Magento database queue implementation can be used. Consumers will be run by cron and all messages will be stored in the database.

Starting from Magento 2.3, the MQF is also available in Community Edition.

The following diagram illustrates the MQF:



About messages processing:

- The publisher sends messages to the exchange.
- The exchange receives the message and sends this message to the appropriate queue. Magento currently uses "topic exchanges", where "topic" is a routing key for the message.
- The queue receives messages and stores them.
- The consumer will get the message from the queue and will execute it.

SYNCHRONOUS WEB API

Since the first version, Magento functionality uses a REST API, which gives developers and integrators access to almost all Magento features and can be used for smooth and efficient communication between Magento and the 3rd party systems. Theoretically this approach can also be used for communication between Magento and Magento Services, but it has several flaws.

- All REST requests are synchronous. This leads to the problem where each system that executes an API request has to wait for the response from the system to be sure that the operation was executed successfully.
- The problem: if operation requires a lot of time to execute, the whole process is slowed down.

Oleksandr Lyzun

The Magento Technical Team Lead at comwrap GmbH. In the last 10 years of working with Magento he developed, led and put live numerous of projects. In June 2018 he became a part of the Magento Community Maintainers Team and in 2019 was nominated with Mageto Master award. Last year he worked on projects related to Magento Message Queue Framework. His passion is to work on complex Magento projects, to provide tailored solutions to the digital challenges of e-commerce customers.

- If we try to send multiple API requests at the same time, this may have a big impact on the system performance
- REST API does not allow us to transfer multiple messages to Magento at the same time.

The reason why this happens is that the initial idea was to support a single point of customization, and it was assumed that if developers need to handle N entities – they will create a single message. But in the case of large data transfers and bearing in mind Magento complexity, which are a reason for database locks, this idea has failed. Those points were the main reason why the Community started to develop Asynchronous and Bulk APIs for Magento, which have moved Magento APIs to the next level.

ASYNCHRONOUS/ BULK API

In discussing asynchronous Magento, we definitely have to cover the topic of using asynchronous API as the main communication mechanism between services. Asynchronous API – is implemented on top of usual REST API. When the system receives API requests which have to be executed in an asynchronous manner, the system places them to the queue. At the same time, the consumer will read messages from the queue and will process them. Bulk API is an implementation on top of Asynchronous API. This allows us to send a single request with multiple objects in the body. The system will split those objects into single messages and will execute them Asynchronously one by one. The main idea of this API was to solve the problems of synchronous REST API.

1) As all Asynchronous/Bulk requests are asynchronous, the response time of all requests is faster than the response time of usual REST requests. And in the case of asynchronous requests, we see an approximately 30% of performance improvements; in the case of Bulk API this change is tremendous.
2) Bulk API provides the possibility to send multiple objects in one request. This leads to a reduction in the number of similar requests and makes the processing of operations faster.

Asynchronous and Bulk API functionality become available in version Magento 2.3. And we want to thank Magento partners: comwrap GmbH (Germany) and Balance Internet (Australia) who made this possible.

CONCLUSION

In the last few years, the evolution of Magento has been amazing. The changes which the Magento Team and the Magento Community are currently trying to implement are so big and so complex that the whole architecture of Magento has to be planned to a very high degree. That is one reason why the MQF is growing and there are still many topics which Magento and its community are trying to integrate. The CQRS and Event Sourcing design principles are the key to the future of Asynchronous Magento. The MQF is a functionality that will be used underneath the Event Sourcing implementation. Magento architecture still has a long and hard road to achieve its goals, but the start has been amazing.

The main challenge for building the Services Isolation in Magento is how to implement communication between services. One of the best ways to solve this issue is a "Smart endpoints and Dumb pipes" approach. Smart endpoints meaning that the main business logic happens behind the main endpoints, on a consumer level. And Dumb pipes are basically communication where no further actions take place, it is simply carrying data across a particular channel. Building stable and performant communication workflows between all parties is a key feature of a stable e-commerce solution. If you want to join the community and help us build the future of commerce, you are always welcome.

WHY CONTRIBUTING TO MAGENTO 2 MATTERS MORE THAN EVER

For the last 2 years, I've been a part of Magento Community Engineering. We are a small, external team tasked with maintaining Magento 2 on GitHub. During this time, I merged around 300 pull requests and participated in more than 500 issues, giving me front seat to how developers are using the platform and the challenges they face.

Miguel Balparda

Magento 2 was not the first open source project I helped maintain. For many years, I maintained Nexcess_Turpentine, a popular module for integrating Varnish and Magento 1.

My role as Magento Master involves traveling, assisting, and educating other developers from around the world. Though it leaves little time for sustained development and coding, it has allowed me to share knowledge with and learn from this incredible global community. Much of my work could be described as providing free support for troublesome issues, but it's just one of many powerful ways to give back to our Magento community.

THE PERKS (AND PERILS) OF CONTRIBUTING

While the use of open source software is itself one way of contributing to further development of that software, it warms my heart to see others jumping headfirst into helping the greater community. Doing so makes you a part of something bigger than your own project and helps make software you already love into something even better. Contributing is also one of the best ways to learn more about Magento 2, its internals, and potential paths for future development.

As a user of a free open source project, you are in debt. You got something for free and you are (probably) profiting from it. At some point, it seems natural to "pay" for it by giving back to the project and the community behind it. This is the true face of open source, where communities are critical for driving the development of a product that helps brands grow.

WANT TO START CONTRIBUTING, BUT DON'T KNOW HOW?

Most of the current needs are covered in the CONTRIBUTION.md file in the root of the GitHub repository, and most of the community engineering team is online on Slack to keep the lines of communication open between devs and maintainers. Look for the "good first issue" label, and you'll find plenty of easy tasks for newcomers.

Labels are just one way to search for issues and PRs. Explore the repositories under the Magento organization for other opportunities. Merchants and power users alike can find plenty of ways to contribute in side projects like MSI, translations in Crowdin, the underappreciated DevDocs project, and Adobe Stock integration, among others. The deeper you go, the more opportunities will present themselves!

As a Magento maintainer, my routine is pretty much the same as any other business professional. This in a key point. Many developers don't realize open source projects depend on the same rules, procedures, and standards common to most development projects. Some examples: just like you would for your own project, test your code before submitting it; remember your manners; templates are not optional and never delete them.

These and other "soft skills" arguably contribute as much to a project's success as the code itself. Magento 2 is one of the largest projects in e-commerce, with over 1000 contributors, thousands of open issues, and around 240 pull requests a month. Given the huge size of the Magento 2 community, professionalism and courtesy are vital for efficient production. Maintainers and core developers are people, not robots. Many maintainers volunteer their time — mostly unpaid — and don't deserve to deal with angry users frustrated with a bug.

HOW TO CONTRIBUTE

Even if you know little to nothing about writing code, there are many ways to contribute. It can be a small typo or a missing translation, but just forking the repository and actively working on your issue will provide insight into how the open source community works.

Want to start contributing, but don't know how? Most of the current needs are covered in the CONTRIBUTION.md file in the root of the GitHub repository, and most of the community engineering team is online on Slack to keep the lines of communication open between devs and maintainers. Look for the "good first issue" label, and you'll find plenty of easy tasks for newcomers.

WHETHER YOU'RE A DEVELOPER OR NOT, HERE'S A FEW OTHER TIPS TO PUT YOU ON THE PATH TO BECOMING A CORE CONTRIBUTOR:

- **When submitting your pull request, remember to make it as easy as possible for the maintainer reviewing your work.** Provide details to save them the trouble of asking for them.
- **Be concise.** Focus on just one issue at the time and don't add unneeded code and features. The simpler the code, the easier to review.
- **Treat others the way you want to be treated.** Be nice and don't be That Guy.
- **Reply to the questions asked by maintainers and others.** The faster you reply, the better, even if the answer is "I don't know but I'll find out."
- **Don't ask for timeframes or deadlines.** Just like you, maintainers have full time "day jobs" and most are unpaid for their work in the community. They'll get to your pull request as soon as they can.
- **Keep maintainers in the loop when you need a break.** If you're working on a pull request and don't have much time this week, let the maintainers know. Most maintainers close PRs with an extended period of no updates.
- **Join Magento Community Engineering on Slack (https://magentocommeng.slack.com/).** Maintainers are almost always available and happy to answer questions, so don't miss your chance to get answers in real-time.

Labels are just one way to search for issues and PRs. Explore the repositories under the Magento organization for other opportunities. Merchants and power users alike can find plenty of ways to contribute in side projects like MSI, translations in Crowdin, the underappreciated DevDocs project, and Adobe Stock integration, among others. The deeper you go, the more opportunities will present themselves!

Still unsure about how to start? Join us in any event! Contribution days, Meet Magento, MageTitans, MageX, Magento Meetups, and many other events will have someone from Community Engineering ready to help you.

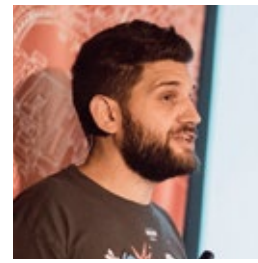
OTHER CONTRIBUTING PRO-TIPS

If you're a developer, take the time to explore your employer's outlook on spending company time to contribute. If necessary, dangle the benefits, contributing: 1) Develops your skill set, 2) Earns a name for your company within that community, and 3) Exposes your team to new methodologies and ideas from different parts of the world.

ADOBE AND THE FUTURE OF OPEN SOURCE

In May 2018, Adobe acquired Magento 2, and the future of open source within the community became unclear.

In this time of uncertainty, the key to keeping open source alive is to keep contributing ideas and code. Adobe has stated their commitment to free open source software (FOSS), but it's on us to keep demonstrating the value of our community. If we want Magento 2 to remain open source and free, we must be indispensable — good advice for any FOSS project but especially true for ours. All credible sources show that more than 50% of new Magento 2 code comes from our community, but now is not the time for complacency. Keep contributing, communicating, and giving Adobe reasons to view our community as an essential part of Magento's future.



Miguel Balparda

Magento Certified Solution Specialist, full time traveler, and sysadmin in his free time. For the last six years, he worked around the globe as a senior developer for some of the biggest Magento projects in more than 40 countries. Since joining Nexcess, Miguel helps maintain Magento 2 as a Community Maintainer and when he's not working, he loves to train and BBQ for his friends.

Commerce is a persistent, transformative force in human history – maybe the most important one that we have – and this can be observed at both macro and micro scales.

Commerce, Community, and Commitment

In the early 1970s, when ARPANET first connected kids from Stanford and MIT, they immediately set about engaging in this most fundamental of activities, and in so doing have the distinction of creating the first online commerce deal (for a bit of marijuana!). It is not hard to see the coupling of commerce and social evolution: the very first advances in communication allowed individuals to cooperate and then ultimately to trade. Fast-forward tens of thousands of years and this evolutionary force has helped drive development, adoption, and expansion of the Internet and the Web.

Anyone reading this column has likely felt the transformative power of commerce directly. Commerce overcomes borders, opens them up, and brings us together. One of the great pleasures of my role is to witness firsthand the impact that ecommerce in general and Magento in particular have had on so many individuals around the world. What started in California in 2007 is now a $5B+/year service economy. That's a lot of salaries paid, businesses built, vacations taken, retirements funded - lives changed, to put it succinctly. It doesn't matter that

this party began in the US. It immediately involved Eastern Europe and quickly spread through the rest of Europe, the Americas, Australia, Asia, and Africa. If only penguins could type and use the Web, we'd have Antarctica too.

Magento's success comes with a profound responsibility which each of us at Magento/Adobe feels every single day. We know that your best efforts deserve our best efforts, because we cannot possibly succeed without enabling your success. Our products and services are the basis of this great story, but it is a story that would never be told if you were not there to render it into what matters to the merchants, markets, verticals, and locales where the customers are.

It's also a story that could not progress without dialogue between us and you. That is why we invest millions of dollars each year in outreach and facilitation initiatives such as Community Outreach (me & Sherrie), Community Engineering (Max Yekaterynenko & team), and recently the Magento Association. In addition to every official channel out there, these channels are open for direct engagement unencumbered by commercial aims. And we are expanding the surface area of this engagement, having just consolidated the Adobe Open Source office under Max's team, continuing the efforts that Matt Asay and others started at Adobe long before the Magento acquisition. Stay tuned for more news in this space - soon.

In the meantime, know that we thank each and every one of you for being a part of this epic adventure, even as we continue pushing ourselves to be better than we were yesterday and to understand you and your customers more and more, day after day.

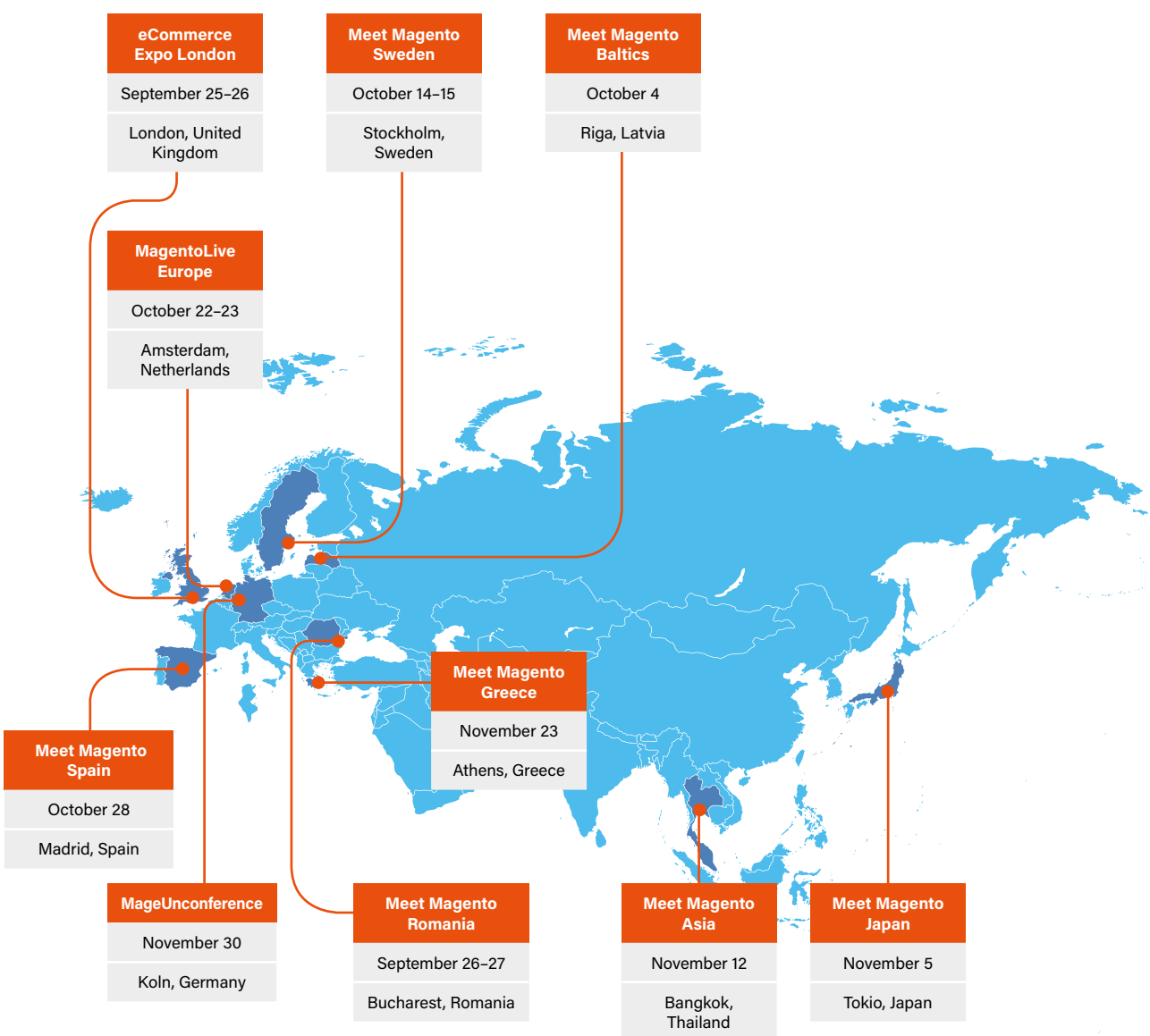
In closing, I will leave you with a request to continue spreading the word about our story. Each new voice, each new set of eyes & ears in our ecosystem provides more opportunity for us to move this story forward and to shape our future together, just as we have been. Do this with the confidence that you are inviting people into an ecosystem committed to open source, open dialogue, and co-ownership of our shared direction and expanding opportunities - together.

Ben Marks

Fifteen-year veteran of the open source ecommerce world. He is a developer, educator, and international speaker, having traveled millions of kilometers to speak with and learn from global commerce and developer communities. He is lead evangelist for Magento, an Adobe Company, and a board member of the Magento Association.

Magento's upcoming events

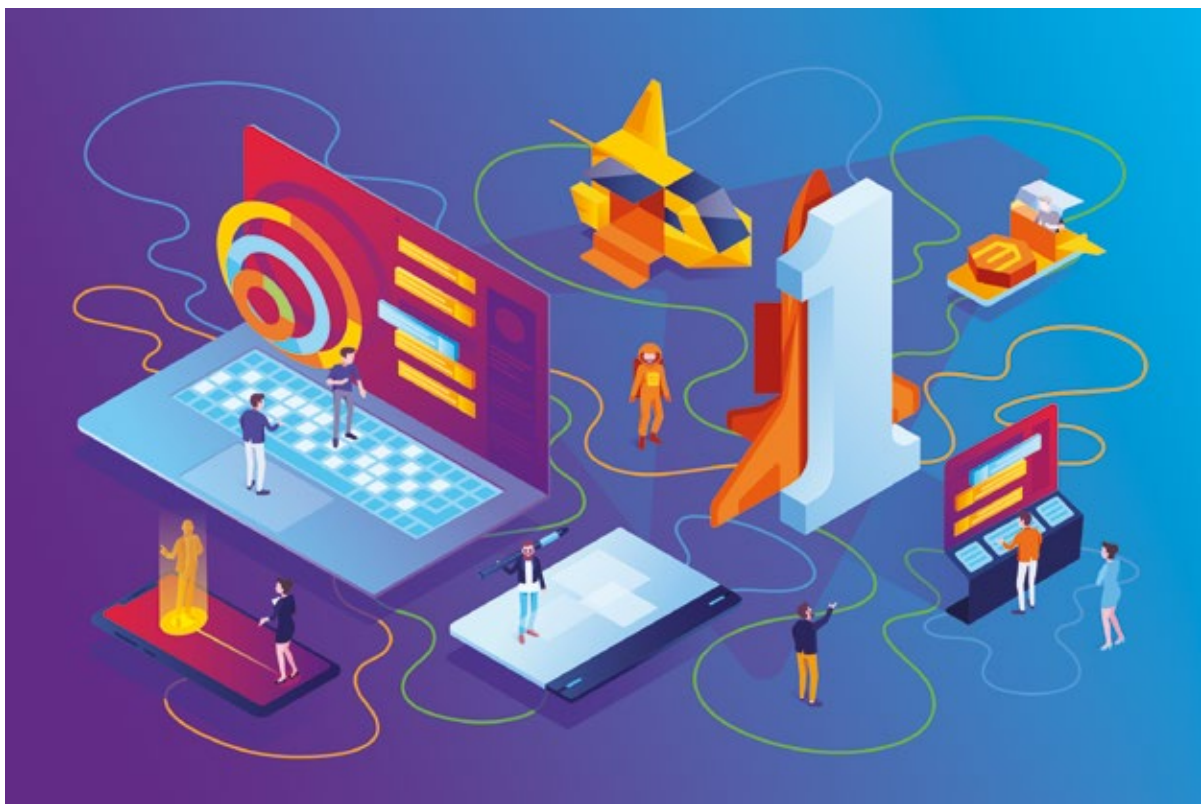
We have an intensive months ahead of us. Magento enthusiasts will have the opportunity to participate in numerous conferences in many countries of Europe and Asia. In order to make your life easier, we have compiled the most important events in a useful map. Grab your calendars and plan your next trip!



ATWIX

PROFESSIONAL

CONTRIBUTOR



What is it about Atwix that you need to know?

First – Atwix is the #1 Contributor to Magento of 2018 and 2019. We not just implement, but develop the Magento platform itself.

Second – no other Professional Solution Partner worldwide has more Magento 2 Certifications than we do.

But most importantly – we simply care and deliver. We always have an improvement for your store.

Can't wait to hear from you at hello@atwix.com

Emarsys is the largest independent marketing platform company in the world and the only marketing platform that knows your industry. With embedded, industry-specific turnkey solutions, our software enables truly personalized, one-to-one interactions between marketers and customers across all channels — building loyalty, enriching the customer journey, and increasing revenue.

What our clients are saying about us

Tupperware®

I think of all the email service providers and marketing automation technologies I've ever used, I have learned the most with Emarsys about different ways to approach the same issues, and, with you guys, that has revolutionized the way I work because I'm able to come up with answers quickly.

Kiara Sanchez-Mora
CRM and Digital Outreach Strategist,
Tupperware U.S. & Canada

BrandAlley

With all of our data in Emarsys, we have all the segmentation that plugs straight into our Ad strategy. Automatically matching content to every audience has seriously boosted our revenue, especially with the increased reach and new streams.

Alexandra Simion
Digital Marketing Manager,
BrandAlley

Char-Broil®

We talked about Emarsys' willingness to help train and be there to answer questions, but it's also just a vibe. We wanted to make sure that we found an ESP partner that truly was supportive and looking to help us create the best email experience for our brands and consumers in a very non-salesman-like way.

Paige Farrow
Senior Director of Marketing,
Char-Broil

To request a demo, visit www.emarsys.com today!

BORN *Digital Experience Agency*

COMPANY OVERIVEW

BORN is an award winning global agency that focuses on Enterprise, Commerce, and Experience Design, with over five hundred digital transformations under its belt in over twenty end markets that cover B2B and B2C segments. It is also the largest independent agency in the customer and brand experience space with offices and operations in eight countries. The company combines ten specialisms to produce high performing digital assets.

Clients include Cartier, Mont Blanc, Lindt & Sprüngli, Starbucks, Purina, Big Bus Tours, AB Inbev, Modells, Ardene, Ferragamo, Tag Heuer, Bulgari, Sotheby's, Maxim Semiconductor, HSS Hire, Textron, Medifast, Rodan + Fields, Lennox, PT Astra, Adastria, Tata Cliq, Singer, Lorna Jane, Razer, Glanbia and Jebsen & Jessen.

- **EMEA**

London (2)
+44(0) 207 520 8600
Birmingham

- **AMERICAS**

New York (2)
New Jersey
San Fransisco
Toronto

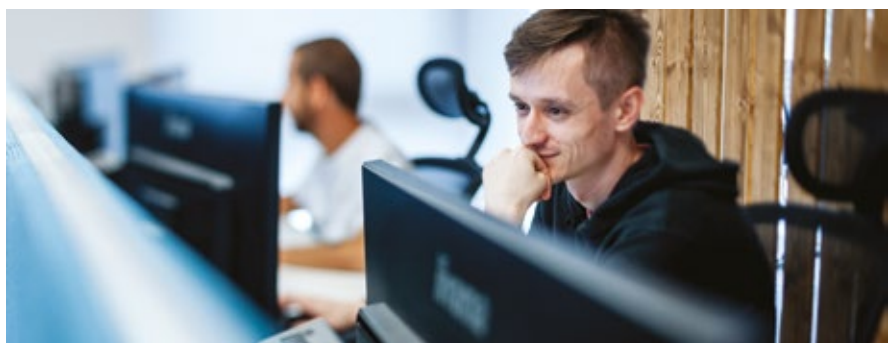
- **APAC**

Chennai
Bangalore
Pune
Hong Kong
Singapore
Tokyo
Kuala Lumpur
Nanjing
Sydney

www.borngroup.com

Strix®

Strix is a team of 100+ experienced consultants and engineers, specializing in consulting and implementation of leading open-source technologies, such as Magento Commerce. Our activities are focused on developing business solutions for B2C and B2B companies and helping them to migrate from traditional sales models to the omnichannel model.



Magento 2 TRAINED SOLUTION PARTNER

ENTERPRISE MAGENTO SOLUTION PARTNER

OMNICHANNEL MAGENTO CONSULTING

What matters is good handling of marketing communications and sale strategy, but only when aided by efficient product range and resource management. Many are the factors that can tip the scales to your advantage. And we happen to know the ones that can truly add value to your business. Selling over the Internet is a tough business, one that imposes refining your sale model. We help our clients to take the decisions, through which they will best fit sale strategies to their brand, organization and clients themselves. Used by thousands of businesses around the world, analytical and concept tools (Business Model Canvas) help us gear up to sell in the fast-moving Internet environment.

UX & DESIGN

We believe a sound design of an e-commerce platform is just like a well designed car. It blends this fine shape, an unparalleled sense of satisfaction that driving it gives you, and the efficient cutting-edge technology. And to achieve it, we need to be highly specialized. A product and business strategy, client's needs analysis, drafting, designing, prototyping, you name it – this variety calls for comprehensive knowledge and skills. And a variety of professions, for that matter. Engineers, architects, designers, sociologists and humanists – all of them are on our team. A multidisciplinary team, we aim at solving specific business challenges. We leave no room to chance and have each decision discussed through and through. Our experience tells us best solutions are born through hard work and selected from a good few dozen of concepts. We do not hide behind catchy presentations but focus on iterating projects so that interactive prototypes can prove their worth for real users.

MAGENTO DEVELOPMENT

What makes clients pick us is the expertise of our implementation team, not its size. Our consultants and developers boast both certificates and hands-on experience; entrusted with tasks, they will deliver solutions, regardless of how difficult they are. There is more to the implementation process than just programming – it is also making decisions that translate into development, sales and scalability of the e-commerce business. Therefore, people who make up our implementation team are developers, but also business consultants and managers. Our cooperation with the Client thrives on our partnership. We realize mutual respect, understanding and trust condition the success of the implemented project. How swiftly we work is all down to simplicity. We do not overdo bureaucracy but concentrate on efficient prototyping of implemented solutions so that their business value can be quickly noticed and evaluated.

- **UK**
  London
  +44 780 32 38 601
- **POLAND**
  Krakow
  Rzeszow
- **CZECH REPUBLIC**
  Prague
- **LITHUANIA**
  Vilnius

CLIENTS

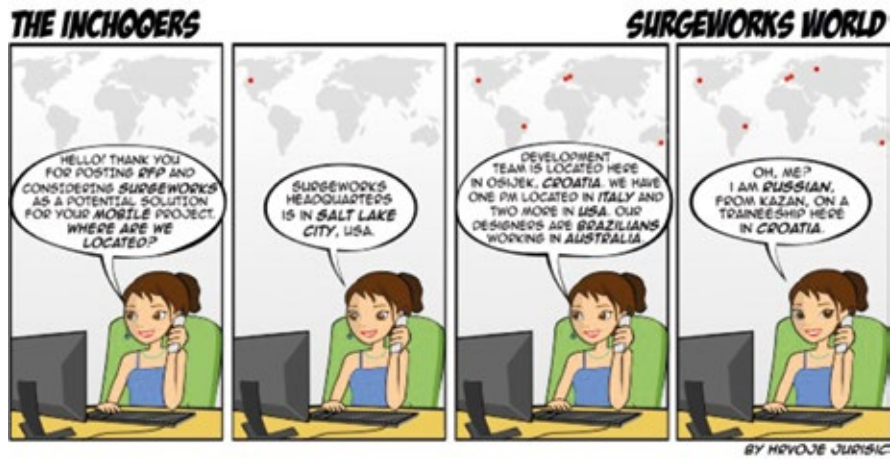
- Castorama
- Tous
- Decathlon
- Inter Cars
- Nestlé
- Super-Pharm
- Mint of Poland
- Semilac
- Lancerto

CONTACT

contact@strix.net
WWW.STRIX.NET

MAGE ZINE

/ MageLite /



By Inchoo Team

MAGEZINE STAFF

MAGEZINE EVANGELIST
Łukasz Bajsarowicz

CONTENT EDITORS
Katarzyna Gądek
Paulina Jóźwik

MANAGING EDITOR AND PRODUCER
Karolina Gajzler-Polak

DTP DESIGNER
Michał Hojnacki

COVER ILLUSTRATION
metaborg studio

PHOTOS AND ILLUSTRATIONS
stock.adobe.com

WWW.MAGEZINE.CO

ergonode

New open-source PIM on the market

- Ready to use for +150k SKUs.
- Excel-like product management.
- Desktop PWA ready.
- Core technologies: Vue.js, Nuxt.js, PostgreSQL, Symfony

www.ergonode.com

MAGE ZINE

Visit
our website
magezine.co
and subscribe
to newsletter

Proudly made by Strix