divante /empowering eCommerce

A comprehensive migration guide
presented in the form of conversation
between two CTOs:

Switching to Headless eCommerce with Vue Storefront & commercetools.

Piotr Karwatka
CTO at Divante and Vue Storefront Co-founder

Halil Köklü
CTO at LoveCrafts

Partners

-lovecrafts-

Vue Storefront

What's inside?

CTO's conversation

Piotr Karwatka talks with Halil Köklü about implementing headless eCommerce - on the real-life case study of LoveCrafts.

Halil Köklü
CTO at LoveCrafts

LoveCrafts is one of the largest crafting community websites and marketplaces in the world, shipping to over two hundred countries.

Halil and his passionate team are weaving together commerce, community, and content for the world of makers.

Prior to joining LoveCrafts in 2013, Halil led technology at Rocket Internet and Namshi, the largest online fashion retailer in the Middle East.

Piotr Karwatka
CTO at Divante

Before starting Divante in 2008, Piotr was already passionate about open-source. As a CTO, he has not strayed from this path.

He co-founded Vue Storefront, the open-source PWA frontend for any eCommerce, which is based on Vue.js.

On a daily basis, Piotr is responsible for technology solutions for leading eCommerce brands. At Divante, he has supported companies like SAP, Levi Strauss, Marc O'Polo, and Zadig&Voltaire in delivering top-notch solutions to their customers.

“The industry is moving from all-in-one platforms to a modular proposition.”

Halil Köklü

Monolithic systems are out-of-date. It is high time to go Headless!

Monoliths are incapable of keeping up with the pace of globally scalable businesses. Headless Architecture based on an API-first approach makes technology better serve the business needs.

What technological stack does LoveCrafts use currently?

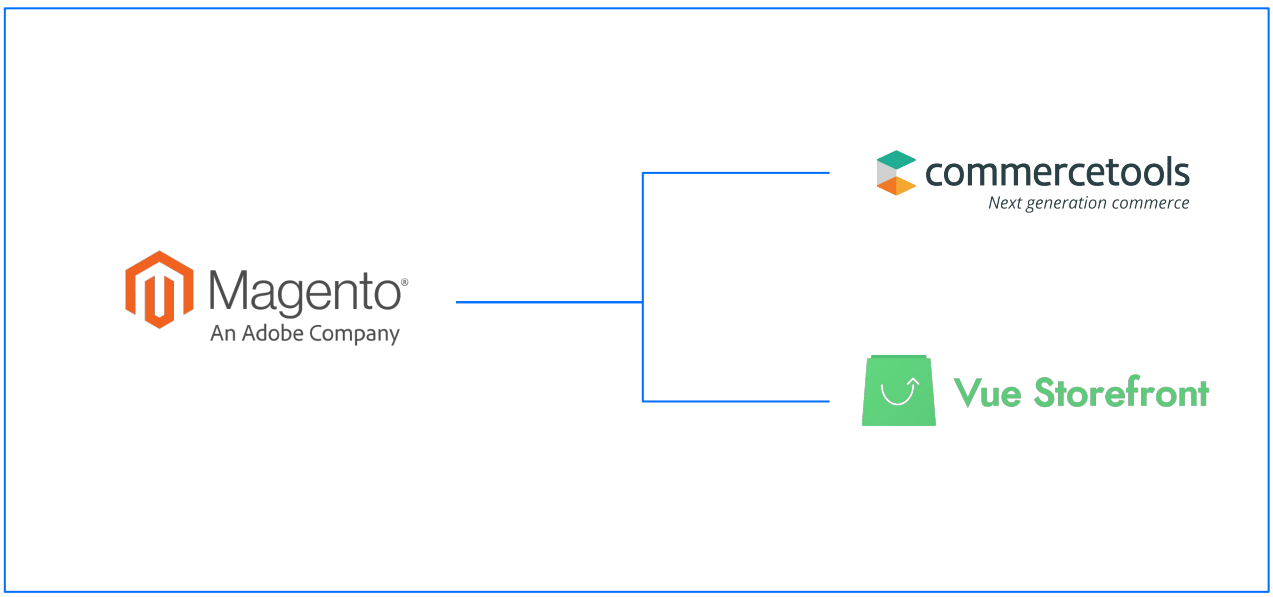
Piotr Karwatka: LoveCrafts was using Magento, and you know this system from the inside out. However, as we all started to realize, there are growing doubts about the future (especially given its end-of-life in June 2020) of this and similar all-in-one systems. What technology stack do you currently use and why is it not just Magento?

Halil Köklü: The transactional part of our product is still run by Magento and we have spent many years scaling it from an infrastructural as well as functional perspective.

Yet, it needs significant investment to run a truly global business at scale with this. Things like multi-warehousing and proper multi-currency and taxation, as well as performance tweaks for indexing, caching, bulk changes, and rendering, are not achievable overnight.

Magento is, however, only one part of our stack. We have built the community and content parts of our product bespoke with, amongst others, Symfony. There is also a good set of internal systems running both business and infrastructure processes.

We are operating in the AWS cloud using various AWS services. We do infrastructure-as-code, continuous delivery, and so on.



LoveCrafts migration scheme

Why is Headless Architecture an answer to the limitations of monolith-implied technology?

Piotr Karwatka: That is why you eventually decided to change the platform and go headless with PWA as a frontend. just to clarify, what is your definition of Headless Architecture and why it is now a best solution?

Halil Köklü: The simple answer is that platforms providing business value such as eCommerce and content management systems are coming without a frontend, the head. They instead provide value through fast, extensive APIs. If done right, all functionality, including admin processes, is available through APIs. You've got full control over the frontend; you can focus on building great user experience and you are not held back by rigid constraints of the backend platforms used.

You can have multiple frontends or touchpoints like native apps, kiosks, in-store, IoT or whatsoever interacting with the same APIs, so you don't end up implementing the same processes in many places.

This is the reason why more and more eCommerce platforms support GraphQL to provide this feature at scale. This is, in contrast, to let's say model-view-controller (MVC) applications where business logic can be accessed by including code. This does not work if the consuming code is not the same, is on a different server, written a different language, etc.

Amongst other benefits, with the headless approach, you can scale the frontend and backend separately.

In the case of Magento 1, you have code and business logic in the controllers, models, and views. It's a nightmare for consistency.

Headless Architecture is, however, not new at all. At the very least, if you have a frontend talking to APIs, you have some element of this already in place.

I would say Headless Architecture and headless commerce are buzzwords for the emergence of API-first or even API-only off-the-shelf solutions.

So, we are not talking about bespoke implementations but major eCommerce software vendors adopting this trend. The market is moving from all-in-one, one-stop or full-stack platforms to a bring-your-own-frontend, modular proposition.

Check out The New Architecture in eCommerce, ebook which presents headless and other modern technological approaches

Why is the scalable API crucial now?

Piotr Karwatka: The APIs weren't a key priority for the established platform vendors and, in eCommerce, the platforms are very often taken as a shortcut to the market.

Halil Köklü: Definitely. Having APIs on its own is not sufficient if they are not designed for scale. Good luck with consuming product data for your frontend via SOAP in Magento 1.

Contenders like commercetools, who have started with headless, had a good head start and are spearheading this as it requires a lot of investment to migrate full-stack to headless.

Larger players are waking up to this as they cannot rely on their market dominance alone anymore, but the transformation is either incomplete or with more or less success.

Here at LoveCrafts, the community product has been built headless since we launched it in 2014. We have an API with some neat features like an expandable query method similar to GraphQL. A Symfony-based frontend, as well as native apps, are consuming from this API.

In 2018, we streamlined our editorial content from WordPress for the blog and Magento CMS for static pages to Prismic, which is a headless content management system.

Back in 2011 and 2012, I was personally involved in launching various eCommerce businesses with Rocket Internet's Alice & Bob platform where Alice is the frontend and Bob is the backend. It was not as tidy as one would expect today; however, it was still somewhat novel. The original authors of that platform went on to other ventures but if you want to check out the latest generation of that, search for Spryker.

LoveCrafts' journey into Headless Architecture.

Choosing commercetools backend with their open-minded API-first approach was inevitable, but not a quick step. The decision process had to take into account all possible long-term scenarios, potential consequences, and possible challenges with the migration.

When did the idea of going headless spark in LoveCrafts?

Piotr Karwatka: The idea of separating the backend layer from the frontend must have been seeded at LoveCrafts a long time ago. It is not an easy decision, giving the scope of work that the migration process required.

Halil Köklü: Honestly, as the engineering team at LoveCrafts, we have been talking about this architecture since the beginning. Obviously, the idea evolved over time with the technologies available.

Whilst we adopted that target architecture in community and content, the commerce part has not made it yet for various reasons. The Magento frontend still dominates the user experience.

At LoveCrafts, we are weaving together commerce, community, and content for our makers. You can't really fragment the user experience by which backend or legacy system is used. All these three pillars are closely interlinked.

Until now, we were trying this with technologies like edge-side includes (ESIs), but it's not working that well from a performance perspective.

Running multiple frontends intertwined, as we ended up with, has many productivity and performance issues. These include starting from sharing CSS and JavaScript to ensuring they are up to date.

We need what we call a backend-agnostic, unified user experience.

What were the reasons LoveCrafts decided to go with commercetools?

Piotr Karwatka: Choosing the right backend was the first step you had to take. What convinced you to bet on commercetools?

Halil Köklü: It was not an easy or quick decision, and we gave this a due process.

A platform change like this does not happen every day and we want to make sure that we will not only cover our needs now but also gain the ability to scale for the future without the need of migrating again for another 5 or 10 years.

We did market research and looked at 30 platforms, of which we reviewed 10 in detail by speaking to solution architects and going through each of our requirements.

The main factors for going with commercetools were actually straightforward:

- Chemistry, approachable, lack of marketing gibberish
- API-only approach
- GraphQL support
- Modularity – when something does not meet your expectations or you have new use cases, you do not need to do a full migration again
- Great and complete documentation
- Really good overlap with our requirements
- Sensible and scalable product data model – e.g. translations are represented in localizable attributes rather than requiring a new store
- Flexible pricing model
- Staging capability for product data, which allows making changes and QA'ing

How did the due-diligence process look?

Piotr Karwatka: Before you had been able to establish all of these advantages, you had to do in-depth research. Can you describe the entire process that eventually led you to these conclusions?

Halil Köklü: The first step is to fully understand your business, product, and technology stack. We are fortunate that most of the original developers and product managers are still with us. I have implemented many of the defining parts of the product, too. So there is a lot of knowledge, meaning we are able to capture the entire picture of our needs. If you have joined recently and are pushing for change, you are going to have a hard time. If you outsource this work to an agency, the results are only going to be as good as the communication and project specification provides.

We ended up with a list of over 100 use cases and requirements, including user features, technical capabilities, back-office functionalities, and licensing and support questions.

Some of those come down to your complexity and product vision. How flexible does it need to be? Are your use cases unique?

We then looked at the market and compiled a list of platforms to review.

The first options on the list were to stay with Magento 1 or upgrade to its successor Magento 2.

Magento 2 can be used headless, but what percentage of the functionality and how many of the marketplace modules are set up for headless? It was all just very unclear.


“Headless Architecture – or as I prefer to call it, API-first technology - is all about serving the business needs.”
Halil Köklü

How were the doubts resolved?

Piotr Karwatka: It seems that the more you dived into the subject, the more questions appeared. Did industry reports from Gartner of Forrester give you any clues?

Halil Köklü: We looked at Gartner Magic Quadrant and Forrester, and it gave us the first impression of those platforms. At their websites, you don”t get much info as most are behind sign-up walls. To gain the most valuable info, you have to contact them directly.

With 30 platforms noted at first, it was quickly clear that we needed to narrow down the list by asking ourselves a few fundamental questions:

Are we going for open source? SaaS? API-only?

This is actually where the implementation starts. It is so fundamental to your plan that you have to be confident that it works with the use cases, surrounding stack, business processes, skills, budget, etc.

Staying on Magento 1 was not an option due to the imminent end-of-life. It would also have required a huge investment in building performant APIs to make it headless. Since it’s already outdated, you would need to make it work with newer technologies

By the way, PHP 7.2 goes end-of-life in November.

Although upgrading to Magento 2 seemed natural, we decided against it.

The data migration would be much simpler; however, the database structure and its challenges are almost identical to Magento 1. We would have to rewrite all our customizations and performance improvements again, thus risking worse performance.

Not to mention that the question "is it going to scale for the future?" would still be open.

Why was upgrading to Magento 2 not an option?

Piotr Karwatka: Also, in your case upgrading to Magento 2 would mean pretty much the same the amount of work as migrating to a totally different platform.

Halil Köklü: We knew Magento 2 does not meet the requirements of our use cases. We would have to spend time building that functionality, and I am not talking about some basic user features but fundamental changes. Once done, we would need to tune for performance.

So, answering your question, at least in our case, it would require the same amount of work.

Also, with Sylius also not making the cut, there were no open-source solutions left in our list. The first thing we struggled with giving up was access to the code. We had some heated debates around that.

.

Was SaaS the obvious choice?

Piotr Karwatka: Eventually, you decided to switch to SaaS. Even though, as we said, upgrading to Magento 2 would cost the same amount of works, it still looks like a bold decision. Did you simply conclude that it is the best delivery model?

Halil Köklü: Unfortunately, it's not that simple.

In general, a closed-source platform deployed on-premises or, in most cases, in your own cloud account, only makes sense if you don't trust the vendor's SLAs.

Otherwise, you end up monitoring, configuring, upgrading the platform yourself, without really having a clue about the platform itself. If you can trust the product and the vendor, SaaS works well.

However, going for SaaS was still not straightforward for us, since none of our SaaS purchases were without surprises or disappointments. And, by that time, you usually have invested too much to step away or you are tied to a contract.

The team was pretty worried about this. Are you really suggesting putting the core of our commercial business into the hands of others? What if they are down? What if we have similar issues like with our other SaaS solutions?

Management added further questions around what happens if the vendor goes bankrupt or their parent company decides to shut them down.

“We are shipping to over 200 countries across the globe from a single multi-lingual website. There is no time of the day without traffic.”

Halil Köklü

How did you address your concerns about specific providers?

Piotr Karwatka: You were nervous about going for SaaS but the remaining platforms on your list are all in exactly in that model. How did you proceed?

Halil Köklü: By doing due diligence, in terms of research but also getting our hands dirty. Many vendors we talked to didn't even provide access to their documentation, some didn't want to do proof-of-concepts (PoC).

The first thing we were looking at was if they are really API-first. I doubted most have started with APIs or API-first. It turned out most of them are full-stack with an API on the side. So they have dedicated APIs for headless which does not cover all the functionality. In fact, you would need to always differentiate between full-stack and headless when looking at any of their articles.

For example, BigCommerce has this. So you look at their documentation and find a dedicated headless section with only two APIs, fetching products and checkout.

The next main thing we were looking at was hard platform limits, which you can only circumvent with expensive workarounds.

In the case of the aforementioned BigCommerce, they have a limit on how many options a variant attribute can have. This is the attribute you use for defining what the variants a product has, like a size or color. The limit with BigCommerce is 250. And that is on the project and not the individual product. So if you have only one variant attribute, you can only have up to 250 variants. We have products with up to 500 variants. Plus, we have over 60,000 colors in the catalog thanks to editorial color names.

In the case of commercetools, the size of a product record in their database is limited to 16MB. This includes two stages (staging, production) and all variants and prices. Attributes defined on a product level and not a variant level are still copied to all variants. So, depending on your data model, you may not be able to have as many variants, attributes, or even locales as you need.

What sources do you use in the due-dilligence process?

Piotr Karwatka: Looks like you are suggesting a certain level of distrust towards the official claims made by SaaS providers?

Halil Köklü: I suggest double-checking everything.

Don’t assume ever that something is standard and should be right there.

Multilingual nowadays should be standard, right? Well, BigCommerce does not have that, Shopify Plus did not have it until very recently.

Check if the documentation is complete and understandable, and if the APIs and representations make sense.

Make sure everything either already fulfills the requirements or use cases you have or can be achieved easily, such as building it yourself in a modular fashion. Don’t expect change requests to be delivered.

Ask to see the roadmap.

Be familiar and confident with the customizability.

The challenge of cloud-first platform customization.

The joined forces of Vue Storefront and commercetool turned out to be the best way to provide LoveCrafts with the flexibility essential in fast-paced eCommerce industry.

What are the ways to customize a cloud-first platform?

Piotr Karwatka: commercetools, as you mentioned, are quite self-protective. Can you elaborate the process of customization based on LoveCrafts experience?

Halil Köklü: It differs significantly between vendors. They can actually provide you with ways of customization; if they don't, then you can't customize. You own your frontend, so the scope of this problem is for the data and APIs you are using and not the look and feel.

I found that commercetools offered the best customizability options. However, they are the least able to make changes to their platform just for you. There is no server instance whatsoever where they make exclusive changes for you. Yet there are few things you can do.

First, you can do some things with the data model, such as signals to the frontend or a background process. For example, if you want to trigger something for specific orders you can use a special order status.

Second, you listen to messages of change events or you poll the latest changes. You set up a message queue and bind a consumer or an AWS Lambda function. This way you can build an ERP or payment integration.

Third, you can use what commercetools calls an API extension. Where the previous method was asynchronous as in it happens after the event has happened, API extensions are synchronous and usually called before an event has been completed, such as an order. So you can interfere by rejecting something if it does not meet certain criteria. These extensions can be built with cloud functions like Lambda, too.

Fourth, you can manipulate the request to and response from commercetools API calls either in the frontend or in an API gateway or middleware; the latter is preferred as you do not clutter the frontend. It is also recommended by commercetools.

Why did Vue Storefront turn out to be a good fit in terms of commercetools customization?

Piotr Karwatka: You eventually decided to support the Vue Storefront Next community by investing in the project and funding the commercetools integration with the VSF Next core itself.

Halil Köklü: It's actually funny. We have spent so much time discussing headless commerce options that, when we have decided on commercetools, my next topic was to decide on a frontend. Nigel, the co-founder, joked "you have convinced us to headless and now you say we need an actual head"?

I think we had four options:

First, to extend the community frontend which is built on Symfony and Backbone to render the commerce pages like product page and checkout. However, the technologies were rather outdated so it felt like we are missing out on opportunities here.

Second, to use a somewhat traditional CMS with a commercetools integration like Hippo. But no one had experience with that and we would probably be back at square one with rigid structures defining our user experience.

Third, to build our own frontend. But then again, how long is it going to take?

Fourth, build on top of an existing but modern and flexible backend-agnostic frontend.

A colleague mentioned Vue Storefront to me and we were excited straightaway. I messaged Patrick and the next morning we were talking.

We discussed that VSF is relying on normalization to be backend-agnostic, so things like product data are indexed into an ElasticSearch cluster which VSF can read from.

We said, "Hey, commercetools already has great APIs, we don't need that. What do you think"?

And Filip said, "Hold on, let me share my screen." He then pitched us the concept of VSF Next on the spot.

VSF Next interacts with backends through contracts, each integration responsible to implement the contract. These contracts are composable, so you can swap loading e.g. categories from a different backend than products. Or you can go and build your own composable.

It sounded exciting, almost too good.

Read the article about Vue Storefront accelerating enterprise commerce initiatives with commercetools.

Why is an open-source model so important?

Piotr Karwatka: You chose to engage in open-source product instead of relying to others. Seems like open-source is in the LoveCrafts DNA.

Halil Köklü: Definitely, especially since we have been using open source projects from the start.

We have a couple of people who contribute to open source as core team members at various projects.

However, to be honest, we never really found a good case for contributing other than reporting and occasionally fixing bugs to open-source projects. Writing Magento extensions did not seem right.

We were planning a few launches but other projects beat us on time with a better product that we felt was not right.

The team was super excited to finally find something we can shape and have a long-term impact.

See video about Vue Storefront Next architecture.

The migration process step by step.

With the decision made, all that was left was to get the job done. But you need to gather the right team before you hit the road.

How did the actual implementation process look at LoveCrafts?

Piotr Karwatka: An implementation process of that scale surely required a variety of different skills. How did you deal with it?

Halil Köklü: As we are strong believers in sustainable product development, and we are not employing contractors or outsourcing projects, handing this project to a Solutions Integrator was out of the question, but everyone on the team got the chance to prepare. It does not matter if you are a Magento specialist or Frontend developer, you got to learn to write and deploy lambda functions, speak GraphQL or know Vue. All of them have also attended commercetools training.

Since we had more flexibility, we introduced an architecture review group to review the team's approaches in the form of pull requests to our internal documentation. We were trying to find consensus before implementation, to avoid surprises later.

We knew that transferring eight years of development into a new stack wouldn't be as easy and fast, so we had to descope as much as we could. This was for both user experience and backend processes. For the user experience, we went through a prioritization process to decide which features would be in the minimum viable product, which would followed, and which would be removed completely.

In terms of backend processes, we have committed to make as few changes as possible to avoid disruption to our colleagues and keep the scope small. Integrations and workflows can take long enough to justify its own project, so putting all these into the scope was not feasible. We decided to sync as much data as possible from Magento to CT. This allows keeping both platforms in sync for parallel operation. The focus is on migrating the user experience and backend processes will follow.

3 of the 5 teams are product development teams equipped with all skills to build user experiences and backend. We also have a platform team which makes sure the infrastructure, CI, logging tools, and so on are working; they constantly try to automate themselves. The enterprise systems team takes care of the majority of the internal products including the ERP, PIM and BI tooling, and so on.

What is LoveCrafts release strategy?

Piotr Karwatka: You are in the migration process. How does your release plan look in order to mitigate all the risks?

Halil Köklü: We are making changes to the user experience both from a PWA perspective but also reduced functionality due to MVP, and we want to get feedback as early as possible. We have built the infrastructure to deploy VSF Next to all the way to production as early as January. We also keep deploying the latest master of Storybook as a static, internal site so that our colleagues can see the UI changes we are working on.

We are about to deliver a “Try beta” link in the header which will be disabled as a feature flag for now.

When we are ready to share our work so far with the rest of the business, we are going to enable it for our colleagues on production. With the emphasis on production. Our colleagues are one of our most critical but also knowledgeable users, so we are keen to get their feedback. Doing things in production, albeit hidden, ensures you are building it right. Also, it allows our product designers to hold user testing sessions.

After we have completed what we call the Public Beta scope, a subset of the MVP functionality, we are planning to open this Try beta functionality to our users to get feedback as early as possible.

Once the MVP scope is complete, we are going to start split testing, sending a portion of the traffic to the new frontend. We can then increase that with confidence.

For the user accounts, we have integrated AWS Cognito in Magento and we are integrating Cognito into VSF.

Magento orders are going to be imported as historical orders into commercetools.

When we are happy with the results, we plan to switch over locale by locale to make sure we don’t have an SEO impact. In case of delays or complications, we have some backup plans.

How you manage content in the headless era?

Piotr Karwatka: Content is a pretty important part of the LoveCrafts service. I've noticed you're using Prismic. Can you say something more about the way you structured the content of the sites using this tool?

Halil Köklü: Like others, we have noticed that HTML is not really content :) The most valuable content is lost in it, but you have to gain full control over the experience on multiple devices.

We have agreed on different content or article types and have implemented slices in Prismic.

The frontend decides how to render the pages. They look stunning!

Prismic works well for us so far and is quite affordable.

.

The desired LoveCrafts architecture.

The new architecture which the LoveCrafts developers are working on will consist of commercetools as a backend platform, Vue Storefront, as a PWA frontend, and...

What are the key building-blocks of the new LoveCrafts architecture?

Piotr Karwatka: LoveCrafts system will be transformed, and the list of changes is not limited to commercetools and Vue Storefront. What is the desired architecture you are planning to develop?

Halil Köklü: Let's start with the frontend. We have a repository where we include both VSF and Storefront UI. We extend Storefront UI with our own components and customizations.

By default, VSF interacts with the commercetools GraphQL API directly, yet we are introducing a Graph gateway and point the composables to that gateway.

The Graph gateway is based on Apollo Server. We federate to Prismic, commercetools, and soon to our community. This allows us to do a single call from the frontend to fetch data from multiple backends, such as an article. In the future, we plan to introduce multiple services that wrap CT services to avoid a monolithic gateway. It also allows assigning ownership to certain services to teams.

The composable architecture of VSF, as well as, the Graph gateway, provides us with an option to potentially migrate away from commercetools partially or fully. The frontend does not need to change.

By having a middleware, you can avoid business logic in the frontend. An example of this is that you need to provide a supply channel, a warehouse, when adding an item to the cart. If you have multiple warehouses, there's a preference decision. This can be done in the middleware instead of the frontend. If you have multiple frontends, this is a lifesaver.

We decided not to use the authentication capabilities in the customer module as if we ever have to leave commercetools, we won't have a way to check if the entered password is correct. So we'd have to ask people to reset their passwords, which is not a great experience.

We reached out to Auth0 and Okta but they were horrendously expensive. We decided to go with AWS Cognito which does not require a contract and is pay-as-you-go.

divante /empowering eCommerce

We build eCommerce solutions for present and future industry leaders… just like you.

Contact us for free online workshop to choose the best ideas for your business.

hello@divante.com

FORBES DIAMONDS Forbes

DELOITTE TECHNOLOGY FAST 500 Deloitte.

TOP 10 MAGENTO DEVELOPERS Clutch