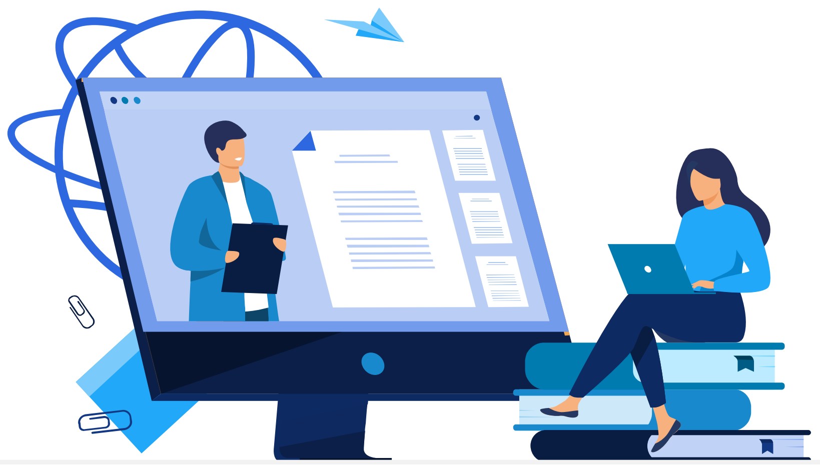
Building headless eCommerce with best of breed approach

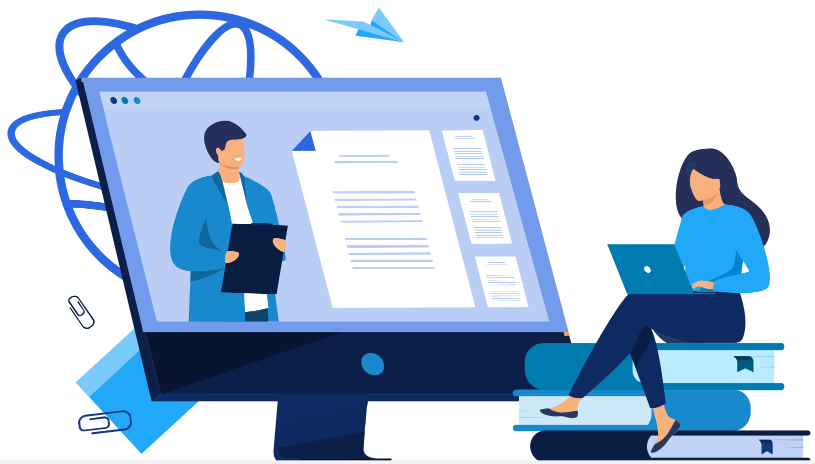
BitBag × Sylius



About the e-book and authors

We created this e-book for established SMBs and Enterprise companies to guide businesses through the modern eCommerce approaches. It contains an overview of the most widespread problems businesses meet while choosing the right direction and technologies in building the eCommerce platform. Solutions that will last for years yet comfortable enough to customize whenever needed to meet the company's strategic goals.

Over the years, we have worked with dozens of businesses, seeing the same problems being repeated over and over again. Spending enormous amounts of time and money while working with legacy approaches, and handling the consequences of bad management decisions while the software was not yet even created.



About the e-book and authors

This e-book will help you better understand the current market possibilities and how you can align the new technology with your business strategy. It will explain how to choose the best tools possible for each job to keep your ROIs as optimal as possible and prevent reinventing the wheel - in the era of many SaaS and open-source driven markets.

There is a good chance that you are not the first one facing a similar problem. Building a good strategy in online sales is all about connecting the dots in the right order. We will try our best to make it easier for you by sharing our experience.

About the e-book and authors

Paweł Jędrzejewski

Founder at Sylius. His work as a leader at Sylius is focused on helping businesses and development teams break-up with their legacy eCommerce platforms and outdated development processes. He has spent the last 10 years starting, designing, building and running bespoke eCommerce projects for mid-market and enterprises. Currently there are more than 2.500+ of them.

Mikołaj Król

CEO of BitBag, one of the fastest growing European technical companies focused on delivering highly scalable eCommerce solutions for businesses using the best of breed approach. Huge fan of open-source software and even a bigger hater of start-from-scratch monolithic approach while building big and complicated solutions.

What is the headless approach?

In a nutshell, the headless approach is all about separating the business responsibilities but not unitizing them. In the past, the software was built to handle solutions as an all-in-one package, which in the long run produced problems with maintenance, performance and, as a result, drove businesses into technical debt. For many, this meant that the cost of maintaining the software was higher than the value it brought.

The same was true for new features development with one additional problem simultaneously - due to the overall system complexity, any customizations broke independent parts of the software without anybody knowing about it. This exposed the application and its data to leaks that, over time, it became less secure. Also, complex software became harder to update, which made the problem even worse.

What is the headless approach?

At the beginning of the 2000s, the Microservices-driven approach was supposed to solve this problem. With a big success of its implementation by companies such as Spotify, Netflix, or Amazon, it became a gold bullet that was supposed to solve any problem the legacy, monolithic approach caused. In reality, microservices worked great for some companies but for those, who wanted to evolve to fast, ended as disaster. Many businesses could not handle it due to a lack of resources, time, and experience.

Over time, no one followed the microservices principles, and it quickly caused businesses to get back to the entry point with an additional hitch of a complicated decoupled architecture. This, as a result, caused higher maintenance costs (as specialists that could handle the Microservices architecture properly are more expensive to hire) and did not necessarily move things forward, often ending as a complete disaster that shut down the initiative.

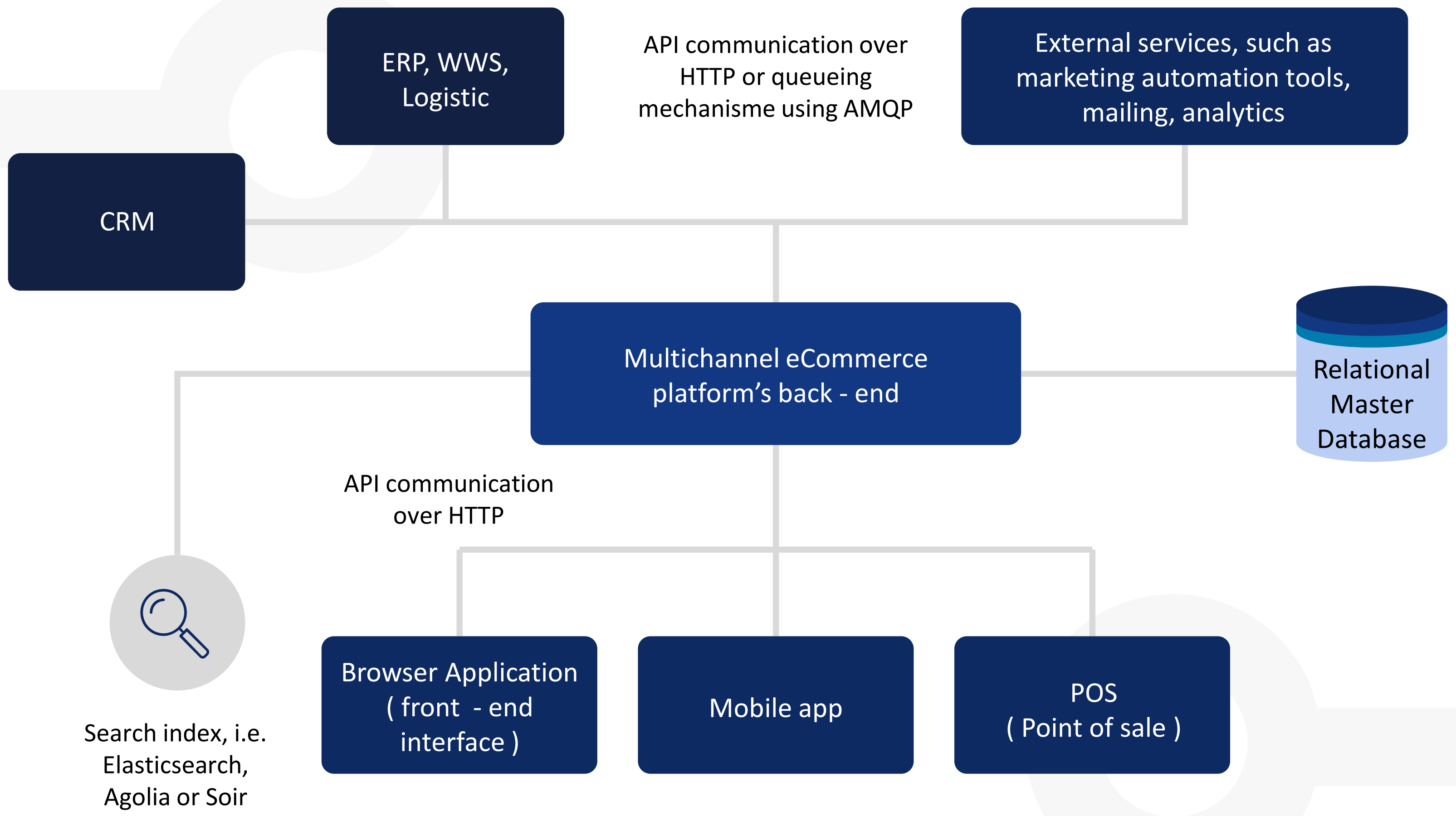
What is the headless approach?

The headless approach simplifies what microservices made complex. The most significant difference between microservices and headless is that the system itself does not have to run single unit responsibility but rather a business responsibility. Simultaneously, it is flexible enough to replace it with running using external services or applications to serve the purpose. Each chain of the headless architecture does not have to run a single database instance or a container as long as the master data access is handled correctly. All the actions are accessed via a separated front-end user interface, a browser (Single Page App/Progressive Web App), mobile application (iOS, Android), a Point Of Sale, or any other custom user touchpoint. In most cases, the glue between services, software, and user interface in eCommerce is the eCommerce platform itself. It aggregates the requests from users, accesses the data and information about processes from external sources. ERP, WMS, provides data for search, analytics, and interfaces to process online sales both on the customer and business side.

BitBag

Sylius

What is the headless approach?


ERP, WWS, Logistic
API communication over HTTP or queueing mechanisme using AMQP
External services, such as marketing automation tools, mailing, analytics
CRM
Multichannel eCommerce platform's back - end
Relational Master Database
API communication over HTTP
Search index, i.e. Elasticsearch, Agolia or Soir
Browser Application (front - end interface)
Mobile app
POS (Point of sale)

What is the headless approach?

The headless approach gives a list of benefits over the monolithic approach, such as:

- More agile management due to decoupled responsibility,
- Integration with legacy and modern software in a decoupled environment,
- Less risk for regression,
- Better user experience on a mobile, browser, and other interfaces,
- More stable architecture due to a better testing environment,
- Better performance,
- Possibility to use the best tools in the market for each area (best of breed approach),
- Better debugging and edge-case handling.

What is the headless approach?

At the same time, the headless experience lists several benefits over microservices:

Lower entry point due to a simplified architecture

Lower costs of implementation and maintenance

Easiness to integrate and scale

Often a better performance

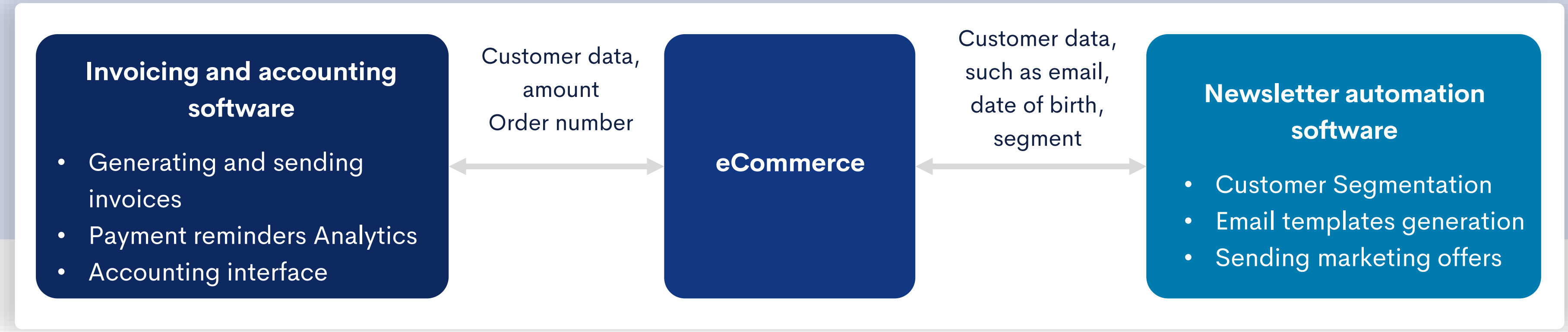
Easier to track

In most cases, better adjusted to eCommerce once used properly.

What are the best of breed and Self-contained Systems approaches?

The best of breed approach represents a strategy of using software that handles a specific problem in a decoupled environment in the best way possible. Just the only needed data is used to process the problem properly, doing its best to cover all business scenarios, including edge cases. One of the most popular best of breed software examples is Software as a Service (SaaS), such as newsletter service, invoicing, or even CRMs.

Let's take a newsletter and invoicing software as an example.


Invoicing and accounting software
• Generating and sending invoices
• Payment reminders Analytics
• Accounting interface
Customer data, amount Order number
eCommerce
Customer data, such as email, date of birth, segment
Newsletter automation software
• Customer Segmentation
• Email templates generation
• Sending marketing offers

What are the best of breed and Self-contained Systems approaches?

The best of breed approach represents a strategy of using software that handles a specific problem in a decoupled environment in the best way possible. To achieve this, it uses the data to process the problem correctly, doing its best to cover all business scenarios, including edge cases. One of the most popular best-of-breed software examples is Software as a Service (SaaS), such as newsletter service, invoicing, or even CRMs.

What are the best of breed and Self-contained Systems approaches?

The Self-contained Systems architecture describes how each chain of a decoupled headless architecture works. Based on the above example, a Self-contained System should:

- Be an autonomous web application,
- Be handled by a single team,
- Communicate asynchronously whenever possible,
- Provide an API interface,
- Include only necessary data and logic based on that data,
- Have its own user interface and work correctly even if the interface does not work at the time,
- Share no business logic with other SCS (be independent)

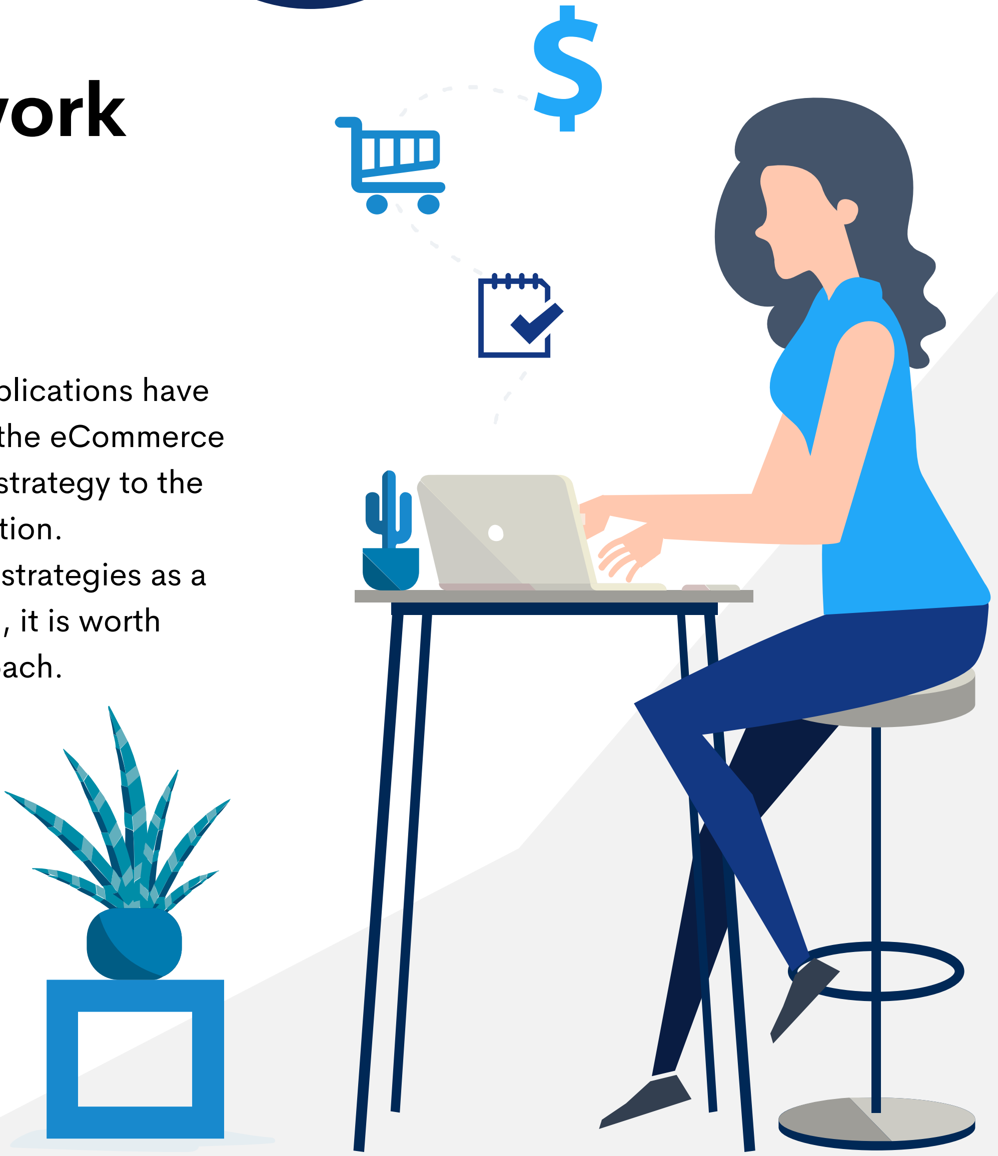
Microservices vs. Self-contained Systems

At first glance, microservices and Self-contained systems might look very similar. They should be, as SCS is just a simplified version of Microservices-oriented architecture. The main difference is that SCS is usually bigger than Microservices and, therefore, it is easier to develop and maintain it, especially if people in the IT team have more than a single responsibility within the whole system.

As mentioned above, the critical problem behind microservices was that it was usually used in a wrong context. It also took a lot of time to discover that it might not be the best solution considering all factors, such as the time to market, team experience, or capacity and efficiency. In summary, SCS is a sweet spot between the monolithic approach and microservices.

Which strategies work for each segment?

All of the mentioned ways of building online applications have their pros and cons. Usually, the key to making the eCommerce implementation successful is to apply a proper strategy to the appropriate segment or current and future situation. Sometimes, it is even sufficient to use all of the strategies as a long-term vision, but before such a call is made, it is worth knowing the best time to apply a specific approach.



Which strategies work for each segment?

The monolithic approach is the quickest way to reach the market, and as long as it does not contain many legacies, it is good enough to start with it. It is even better if the software is being built in a modular way so that it is possible to migrate specific parts of the infrastructure to the next level or replace some responsibilities if needed. Currently, many eCommerce platforms provide a lot of basic functionality out of the box, which you can either extend or replace with a third-party service. Moreover, a server rendering application is not a bad idea at first, given the time and price benefit it gives.

Businesses usually decide to follow the monolithic approach once the time and budget pressure are high. This is true in two scenarios. First of which is a startup that needs to provide a Proof of Concept within a specific timeframe to produce real market value. The second, and the most popular one in the SMB/Enterprise segments, is a business that failed the migration process. The technical debt cost starts to produce high maintenance costs and limits the strategy implementation. Whenever a business strategy gets slowed down by technology, it is worth considering whenever the taken approach is right for the current situation.

Which strategies work for each segment?

A well-built monolithic application should be easy enough to migrate to a Self-contained System. Practice shows that in the era of many great tools available over the web both as SaaS and open-source, using the SCS is a more reliable, easier to implement, and rational way to go. The SCS approach is usually well adjusted to an SMB/Enterprise eCommerce with a higher customer volume and orders. It is less risky than the Monolithic approach and much more straightforward than microservices, which usually are too complicated for most businesses to use, even on the Enterprise scale.

Decisions about the architecture design could be adjusted following Agile principles. As any advanced eCommerce project is a living instance, which needs to adapt to the continually changing market, it is essential to apply a proper strategy that will live up to it. In 2020 Self-contained Systems architecture is one of the most popular, most reasonable, and therefore probably the best way to build scalable eCommerce systems that will last for years, not limiting the business at the same time. It is also super easy to migrate from a modular monolithic architecture to an SCS, which is much harder in microservices. On the side note, you can also migrate from SCS easily.

Team Strategies

One of the critical factors in making an appropriate decision while choosing the implementation strategy is work capacities, directly related to time and budget.

Considering a business has at least a small IT team before they decide to build the solution internally, it is worth considering a few risks associated with this approach:

In the case of migration, the internal IT team might have a lot of experience with the legacy platform, not necessarily the modern solutions. In the long run, this produces the risk of diving into technical debt.

When hiring a new team, the responsibility of training new engineers is taken by the business. This might cause the project to become a significant and expensive R&D.

The IT team might be too small to develop a well-designed solution in the required timeframe. It does not make sense to hire people to fulfill the project requirements for just a few months, and postponing the go-live is too expensive for the business.

Engineers experienced in many technologies at once might be very expensive to hire. Due to the constantly changing IT market, their broad experience might get outdated quickly, and the quality of their work to price ratio might not be acceptable.

Team Strategies

It is often a good idea to consider an external partner who has experience with technology and can scale the experience and timeframe on an optimum overall price. It is much easier for an external team to jump into business requirements and adjust the technology accordingly than the other way around. Likely, an experienced partner has already faced a similar problem in the past. Having someone with ready patterns by your side will prevent you from making bad business decisions, which might turn out to be pricey. The same experience should also shorten the implementation time and its overall cost.

No matter if you decide to work with an external partner, it is essential to align a proper implementation strategy knowing all the risks.

Need help with choosing the best approach to headless ecommerce implementation?

CONTACT US

Sylius

Tymoteusz Stengert

tymoteusz.stengert@sylius.com

+48 790 206 671 or schedule a **call**

Headless solution for the most demanding businesses

DISCOVER SYLIUS

BitBag

Mikołaj Król

mikolaj.krol@bitbag.io

+48 786 988 882

Development of the headless eCommerce solutions

ASK ABOUT HEADLESS