



The architect's guide

to core banking.

Insight paper

Learn about the background of core banking and how adopting cloud-native technology is finally within reach for banks.

five°degrees

A future-proof and legacy-free architecture.

Wouldn't that be wonderful? Unfortunately, anyone who works in technology knows there is no such thing as future proof. After all, there was a time that a terabyte required a million-dollar investment, and we could not imagine ever needing more storage.

The real challenge is not to build an architecture that is future-proof. The challenge is to build an architecture that is agile. One capable of adjusting to the change in customer demand. One that can meet increasing requirements in terms of operational excellence, and compliance?

You have likely already adapted your front-end to meet a higher standard of customer experience. But in the long term, you need more than good looks. The entire architecture needs to be truly agile. This means replacing static legacy technology in the core with modern, cloud-based technology.

It was already in August 2019 that McKinsey identified the urgency for incumbent banks to shift from their traditional core banking systems to a modern and cloud-native platform. We have now reached a stage where innovative core banking suppliers can offer solutions to this urgent demand.

 'McKinsey - Beyond digital transformations: Modernizing core technology for the AI bank of the future'



**cloud-native
core banking.**

01	The target architecture: a composable landscape	4
02	The definition of cloud native	7
03	Upgrading strategies	9
04	Selecting the right modular vendors	12
05	Managing data in a composable landscape	15
06	Experience matters	17
07	Some best practices	20

01 The target architecture: a composable landscape

A four-layer architecture

When building a digital bank, we identify four layers within the architecture:

The presentation layer

This is the front-end. This is the layer clients and employees use. In general, the client portion of this layer is already hosted in the cloud or at a minimum built on non-legacy technology. Mainly because ever upgrading devices and operating systems demand this layer to be updated frequently and adherent to the most modern standards.

The orchestration layer

This layer handles client-related processes and data. It defines how the other layers interact with each other.

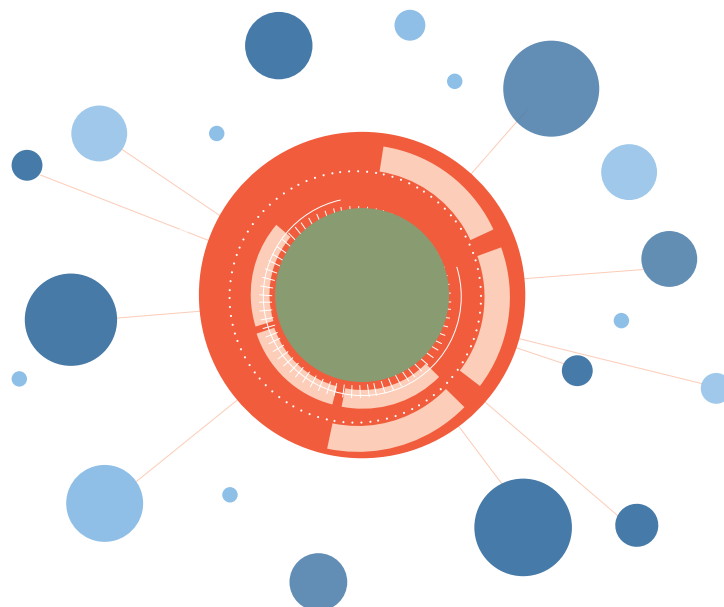
The core

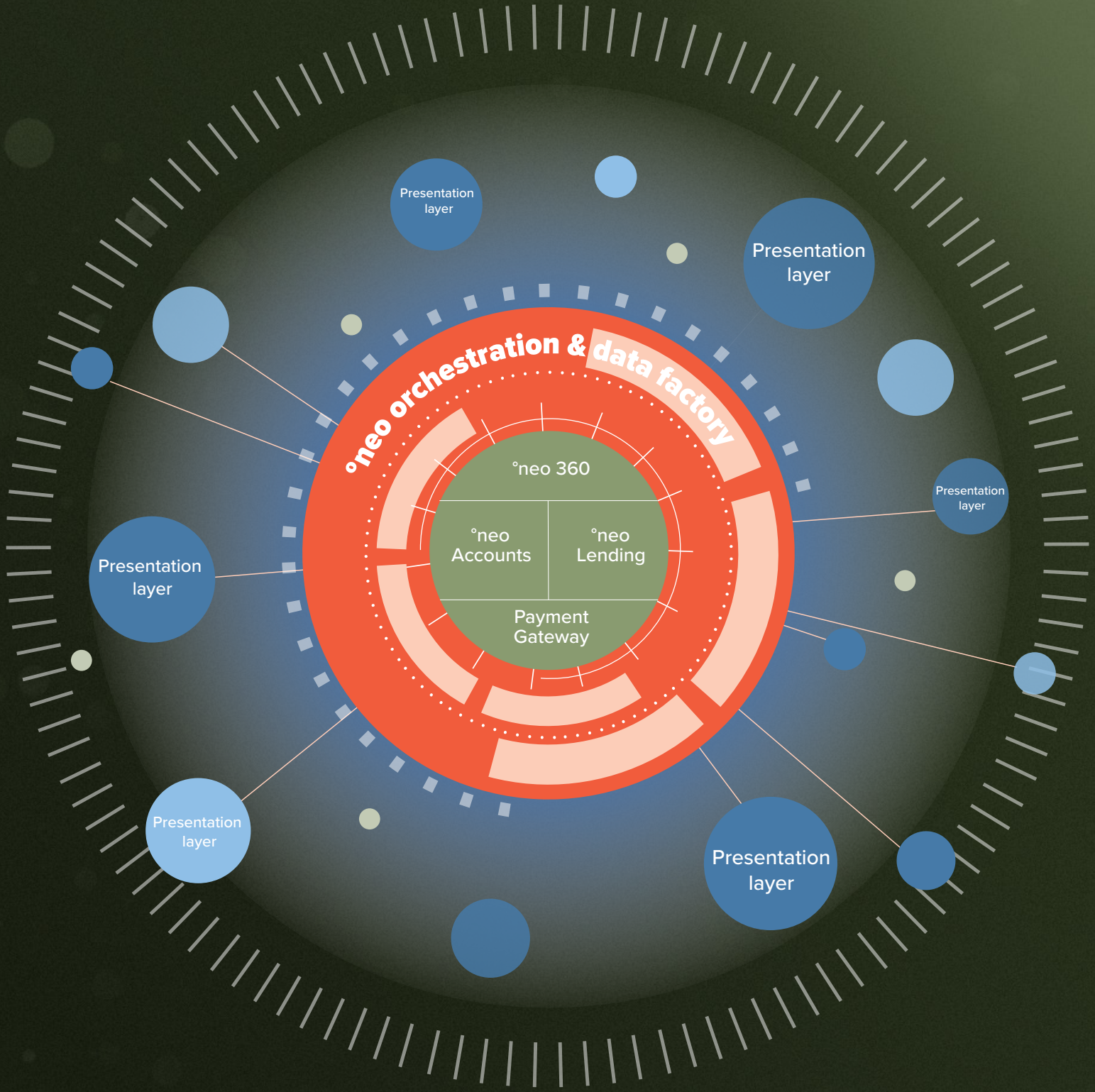
This product layer contains all rules that define products and the calculations that follow those rules. It contains the true essence of banking.

Data

This contains client data, and transactional data.

For the best result, the layers have a modular setup, based on microservices, all connected via APIs. This creates both resilience and agility. It creates a composable landscape where components can be freely taken out and replaced for a more suitable component. Also, updates and improvements can be made to a subset of the landscape without the necessity of updating everything. This way, it is easier to fulfil future requirements we are currently not even aware of.





02 The definition of cloud native

The term cloud native means different things to different people. That is why it is important to define it:

What is *SaaS*?

SaaS application is delivered over the internet and maintained by the technology provider. Compared to on-premise technology, a SaaS application is not installed locally, which eliminates related software maintenance and hardware management. The application runs on servers in a data center; “the cloud”.

What is *lift and shift*?

It is possible to use the cloud as a hosting environment only. Basically, you mimic an on-premise situation in the cloud. Regardless of where something is hosted, the technology can be containerized to make it easier to deploy portions of the application, rather than the whole in one go. Since with the lift and shift approach you keep the actual application the way it is, you don't use the specific cloud services. So therefore, you don't reap the full benefits of the cloud.

What is *cloud native*?

Cloud native is an approach to creating and running an application that fully utilizes the cloud computing delivery model. Compared to technology that is lifted to the cloud, cloud native is built for the cloud and usually comes with significant benefits in terms of efficiency, openness, and scalability. We believe that cloud native means you choose a specific cloud vendor and optimize the use of the managed services provided by that particular vendor. ^oneo is native to azure.

What is *cloud agnostic*?

Cloud agnostic is the opposite of choosing a specific cloud vendor to optimize the use of managed services. You develop without commitment to a specific cloud vendor and use only services that are available to all. Good to know is that also cloud-agnostic platforms can be integrated with cloud-native platforms that are native to a specific technology provider (e.g. Azure or AWS).

Ensuring enough cloud-native capabilities

To ensure the system you buy has enough cloud-native capabilities, include the following two things in your vendor review:

1. Do a *proof of concept*. Get a sandbox to test some potential integrations.
2. Have the proof of concept include a business rule and product very specific to your organization. Make sure the component offers the right configuration dimensions to enable your specific business.

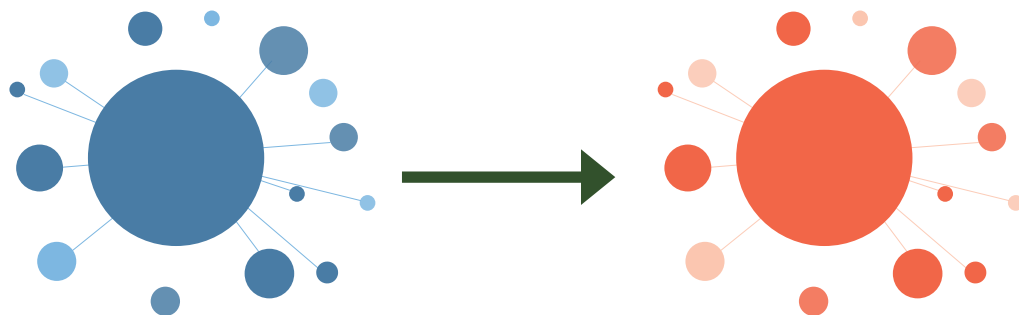
03 Upgrading strategies

How does one transition from traditional architecture to microservices-based architecture?
In short, there are three strategies:

- A. Big bang**
- B. Eating the elephant**
- C. Satellite structure**

Big bang

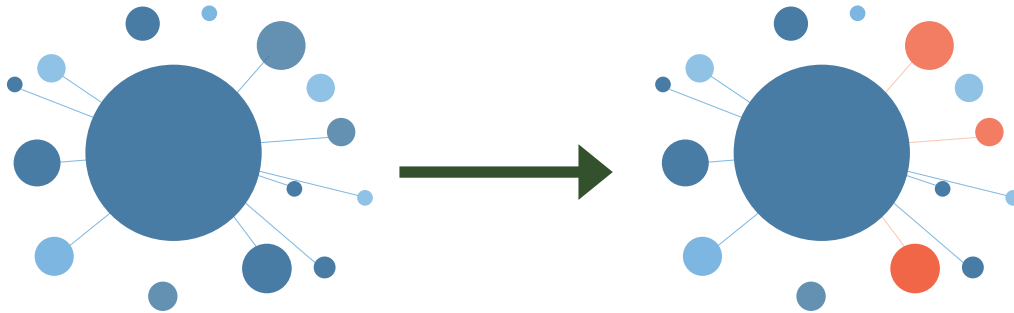
With a big bang, you do a full replacement of the entire core. This can be necessary due to big issues with the current core system. Either it is too vulnerable in terms of security, or it is impossible to comply with regulatory requirements.



- Higher risk
- Extensive data migration
- No benefits until project fully completed
- + Entire architecture is modernized

Eating the elephant

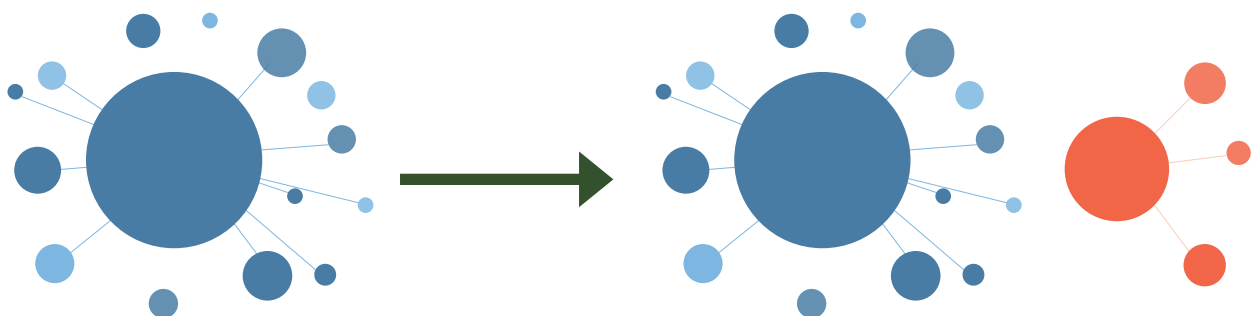
One bite at a time. That's how you eat an elephant. By replacing segments of the business at a time, you gradually upgrade the architecture. Because not everything is replaced simultaneously, you have the opportunity to select the latest and most excellent vendors at every crossroads. The end result in theory is more modern than the end state in a big bang approach. But there is the risk of never being done.



- Progress is made in small steps
- Dependency on legacy systems remains for several years.
- + Less risk
- + Benefits are reaped instantly

Satellite structure

The third option is creating a new proposition, built on new technology. Legacy is not holding this new proposition back.



- Dependency of original brand on legacy systems remains for several years
- Benefits only reaped within new brand
- Additional costs, not replacement costs
- + Allows for a leaner proposition than the original one
- + Technology can proof itself before introducing existing client base



04 **Selecting the right modular vendors**

Taking a best-in-breed approach means selecting vendors for your components. There are several elements to be mindful of when selecting your vendors. We have discussed them in our paper “How to select your cloud-based vendors”, which includes important questions to include in your RFI when interviewing vendors. Here is a summary of the ten selection principles.



For more information, [download your paper here](#).

1. The cloud choice

It's important to check if your vendors offer their platforms and solutions

- cloud agnostic,
- cloud specific to a particular cloud e.g. AWS or Azure native, or on-premise with the ability to host in the cloud.

Both the first and the second can refer to their solution as being cloud native. Make sure you know which one is offered.

2. The granularity of the system

Be mindful of the granularity of the system offered. Find a balance between the safe, overseeable structure of a monolith and the agile, scalable but potentially chaotic microservice structure.

3. Ecosystem-friendly

Of course, your new system has APIs. Make sure to take a close look at:

- What is included in the APIs: is it just data, or also events, processes, and more.
- The support offered for those APIs, like searchable documentation, code examples, and cookbooks.

4. Data encryption

Security is the top priority. You know this. So, make sure all data is encrypted. When it comes to cloud solutions, make sure that encryption covers both data storage and data in transit.

5. Scaling (automatic or manual)

This is a true cloud quality. Cloud-native tooling has automatic scaling. A lift-and-shift solution does not. This is a clear distinction between ‘cloud ready’ and ‘cloud native’.

6. Availability

Always check the SLA (service level agreement) for availability guarantees. A vendor of a cloud solution should be able to provide 24/7 uptime and service. Just imagine that percentage of downtime happening in the middle of black Friday and base your selection on that.

7. Deployment cadence

A vendor can offer periodic or continuous deployment. You want small changes every day, and continuous innovation of your landscape, rather than big upgrades every six months that come with much work. This means taking a continuous deployment approach. This is not a distinction between cloud-based and on-premises. An on-premise solution could just as well have continuous deployment. It's a quality that tells a lot about the way a solution is organized.

8. Single codebase or multiple codebases

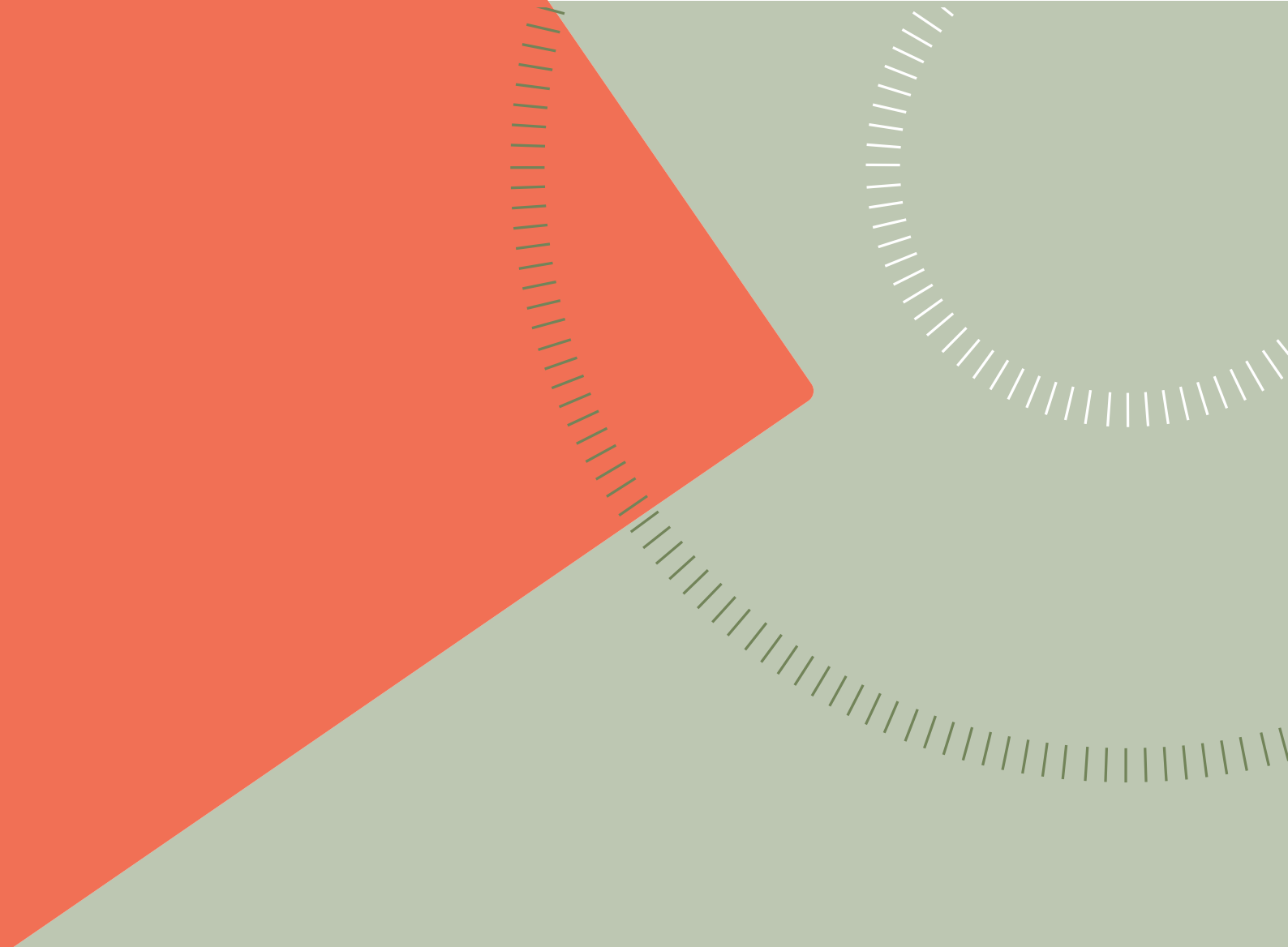
Your landscape is unique to you. It may require specific things from your vendors. Like custom fields or processes. Check if that requires custom code or not.

9. Testing

A well-tested system is vital. Make sure you know how changes to the solution are tested. Is there manual testing, automated testing, or hybrid testing?

10. Low-code configuration

The configuration enables you to meet custom requirements without creating multiple codebases. Some configuration requires a lot of coding experience or is poorly documented, which creates a black box of capabilities. As with the API, you want good documentation with clear information on the scope of the configuration.



05 Managing data in a composable landscape

Data is the new gold. Your new architecture should enable you to access that gold in the best possible way.

With a traditional monolithic system, data access is relatively simple. With modular or composable banking, every component has its own data storage. How do you avoid it becoming a big mess and your data being everywhere? The answer is threefold.



A consolidated data warehouse.

All data from your component-based architecture should be collected in a single concise system.



A human-readable data model.

Your data, stripped from irrelevant data fields, should be accessible in a data model that allows for analysis. A real person should be able to make sense of this.



Multiple data access types.

You should be able to connect to your data warehouse using more than just an API. As mentioned, you should have an export function for your reporting. But you should also be able to flag data events and initiate tasks within your landscape. Regardless of which module within your architecture is needed for that task.

But most importantly, your data must remain your own to access. In some cases, data is locked in an application and can only be accessed with help of the vendor.

What we did: data can be unlocked via three methods:

APIs – every data field is accessible via the API. Custom fields are automatically included in the API.

Events – every data change is an event. Any element within the ecosystem can subscribe to an event and trigger a response.

Database reporting stack – a copy of the database is available at any given time to allow for data analysis and reporting. This can be PowerBI or any other analysis tool of your preference.



06 Experience matters

A great user experience brings operational efficiency and employee happiness and retention. This in place can result in customer satisfaction and retention. So it is of great importance, even for a core banking system, to offer a great user experience.

When modules and components are created in a monolithic system, they are usually created by the same people working on the core software, resulting in a relatively consistent user experience, visual design, and code base.

Design consistency is crucial to user experience. If you plan on integrating third-party modules, it's almost certain that they won't fit your own product's design out of the box, so additional work will always be required. How much work depends on how customizable they are; hence, customisability is a powerful consideration when selecting which modules to use.

When integrating the design of third-party modules, there are two high-level approaches:

Make integrations obvious

Here you don't try and hide the fact it's not native to your product. This is basically telling the user "We didn't make this, but you can still use it." If you go this route, make it visually clear that integrated functions are 'different'. You can do this by wrapping them in a different container or including the third-party logo. This approach will be necessary if your selected modules are not customizable.

Blend them in

Choose modules that can be customized and style them as close to your own design as possible. How well you can do this will always depend on how customizable the modules are.

How to prioritize module selection with respect to design:

In composing your banking landscape, it is likely you run into various vendors offering more or less the same. Here are three guidelines to prioritize with respect to design.

1. Functionality first

It seems an obvious one, but functionality comes first. This is where we deliver the core value to users. Aesthetics take a back seat to users 'getting the job done'.

The alternative is *Form over function*. If something doesn't offer the required functionality, no amount of impressive design is going to compensate for that.

2. Visual customisability

Even a slight variation in spacing, sizing, and font styling can shatter the user experience, and make it obvious that this piece ‘doesn’t belong here’, so the more customizable, the better.

The pieces that should be customizable, in order of importance:

- Colors
- Font styles and sizes
- Buttons styling (size and shape)
- Spacing (usually margins and padding)
- Additional styling (drop shadows, 3D effects)
- Animations (hover behavior, transitions, etc.)

3. UX Customisability

It’s rare that you’ll encounter modules that allow you to customize user flows and where and when certain elements appear, but they do exist. In such cases, you’ll be able to change the placement of buttons and input fields, which when coupled with visual customizations, allows you to create an experience similar to that of your core product.

Ultimately, the quality of the user experience is only partially defined by visual consistency. The best user experience is created for your users and validated by your users. We recommend you speak to your users regularly and validate each integration as part of your product development process.

What we did:

“When creating °neo we employed intensive user research and validation practices and they gave us absolute confidence in our designs, and our product.”



07 Some best practices

Finally, we want to leave you with some **best practices** for transitioning your core system.



Talk to your peers

Gather information and talk to other financial institutions. Include Fintech initiatives and incumbents. Get as many insights as you can and become familiar with the vast offering of components available. Include your CCO in these conversations to amplify your business case.



Finding the right partner

Aside from modular vendors, make sure you find a trusted core banking supplier to help you with the transition. Essential questions to ask yourself are: Is there a cultural fit? Is the platform based on modern technology? Do they have a vision for the future of banking? Do they understand the day-to-day reality of your business?



Perform a risk assessment

With potential partners, perform a risk assessment. Define assessment criteria, examine the organization, and make sure you have a clear overview of any risks. Don't forget to map the dependencies on this partner. Are you free to configure any specifics for your organization or do you need their assistance in your operational activities?



Designing and testing workflows

Design and test workflows with your vendor. Creating a proof of concept is an efficient way to do so. It gives you the possibility to see and experience how a specific element works, test functionality, and determine if the selected vendor can help you achieve your goals.

About Five Degrees

Five Degrees was founded by bankers in 2010. Our launching customer KNAB was based on our technology to become the first challenger bank in Europe in 2012. Soon incumbent banks throughout Western Europe, the United States, and Canada also chose Five Degrees as their digital banking technology provider. In 2018, Five Degrees added lending to its product suite by the acquisition of Libra EHF, the market leader in lending technology in Iceland. Today, over 40 banks in Europe and North America use our technology.

Over 150 colleagues work from our offices in Amsterdam, Reykjavik, Lisbon, and Novi Sad. After a decade of on-premise solutions, we now offer our product °matrix as a SaaS solution. We launched our latest product °neo in 2021. The °neo core banking platform was built for the cloud from the ground up to continue to allow our clients to meet ever-changing market demands

Let's start a conversation. [Connect with us.](#)

We are happy to show you our cloud-native core banking platform and see if we are a good fit for your business.

Connect with us

 www.fivedegrees.com

 sales@fivedegrees.com

 (+31) 088 008 6400



five°degrees

An aerial photograph of a city, likely Taipei, featuring a prominent circular building with a central blue pool. The building has a white, perforated exterior and a brick-patterned interior. The surrounding area is densely packed with various urban buildings, including residential blocks and commercial structures. The image is used as a background for a bank advertisement.

**Be the
best bank
you can be.**

five°degrees