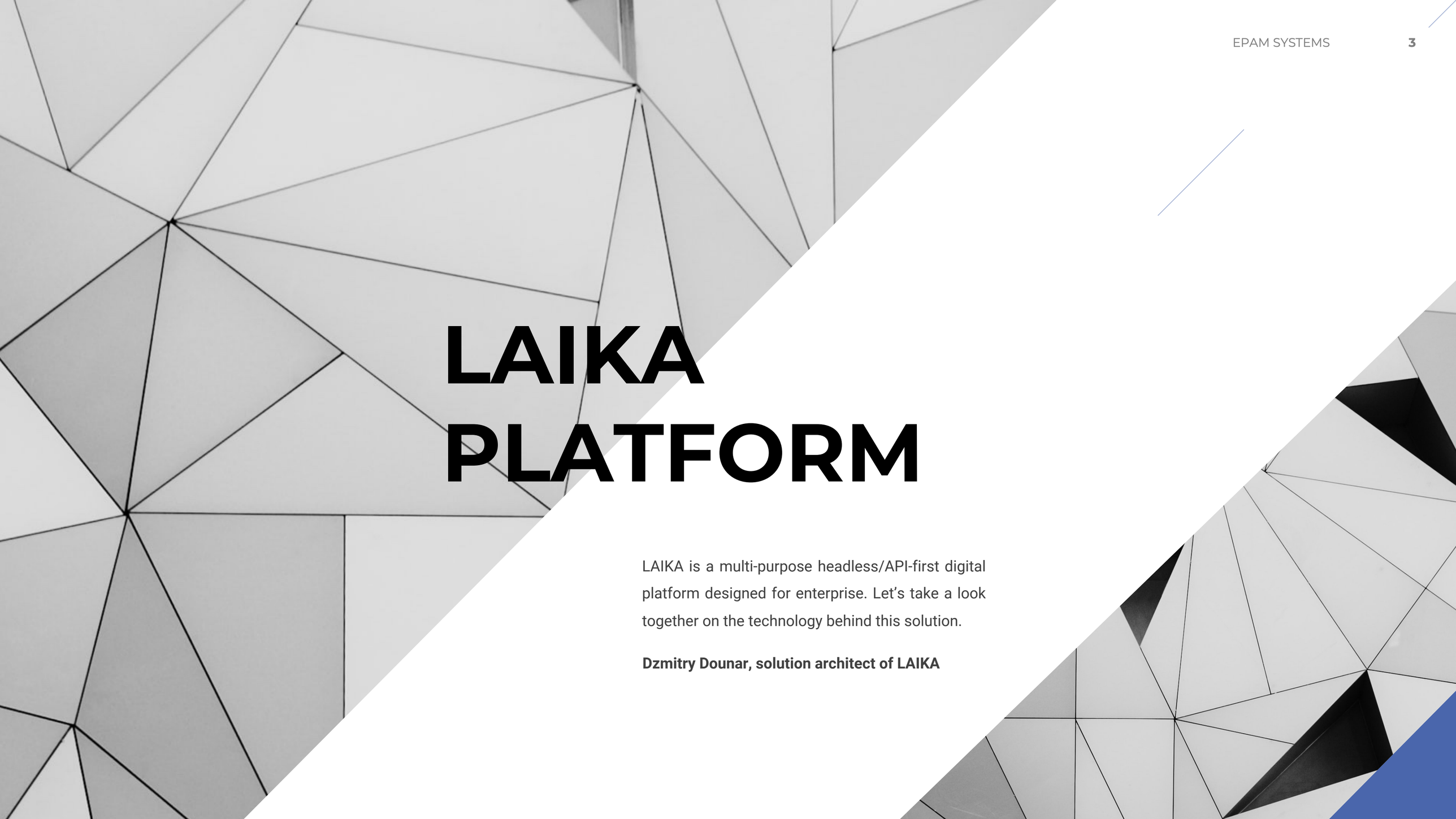

LAIKA

LAIKA TECHNICAL INTRODUCTION

Modern Scalable Flexible Extensible Enterprise-ready

CLICK TO CONTINUE



LAIKA PLATFORM

LAIKA is a multi-purpose headless/API-first digital platform designed for enterprise. Let’s take a look together on the technology behind this solution.

Dzmitry Dounar, solution architect of LAIKA

CODE BASE

127K+

Lines of code

LAIKA platform is a pretty big and mature project. It’s not something built as an accelerator that should be finished, it’s a full-featured product **fine-tailored from scratch** by team of professionals working more than 10 years in digital asset management and product information management business area.

70+

Services / applications

As we will discuss further, LAIKA is a modern microservice-based platform so it based **40+ microservices** and **10+ libraries** created to cover some independent part of functionality. But not only: platform also has a set of reference applications for DAM, PCM, MCM, digital publications as well as various tools.

13+

Man/years invested

It’s not a lot taking into account the functionality our platform provides. At the same time it was **built by the team of professionals** that working with such solutions for years that’s why we achieved such results from the end of 2017 when we started this project. LAIKA is a mature platform designed for real-life usage.

CLICK TO CONTINUE

OUR STACK

MICROSOFT .NET CORE LTS

Our solution is based on the **Microsoft .NET Core 3.1 LTS**. It's a modern open-source and cross-platform framework that is used in all of the components of the LAIKA. **Microsoft ASP.NET Core 3.1 LTS** is used for web applications and web APIs. Microsoft made a huge step forward in comparison with previous issues of framework and now it's possible to run .NET Core applications on all major operating systems: Windows, Linux and MacOS.

At the same time **it's not required to know .NET framework** and languages of this platform (C#, F#): all functionality is available via REST API, so in our case .NET is just a runtime that should be installed to run an application.

01. MONGO DB

In our case Mongo is a main storage of the data that platform has. It’s document oriented database it perfectly fits for storage or objects such as assets, products, marketing programs and others where metadata model may vary from object to object. We also using transactions across replica sets for critical operations. And it’s perfectly scalable and it is a good advantage for real-life usage.

02. ELASTICSEARCH

Elasticsearch is a modern and powerful search engine. This component used for search, filtering, faceted search and full-text search purposes. It also used as a primary log storage while Kibana is used as a frontend for log view. LAIKA code is based on NEST: an ORM library for Elasticsearch.

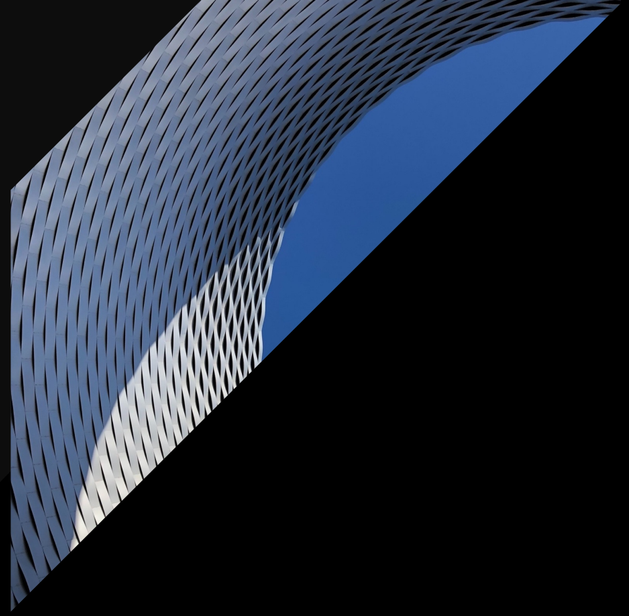
03. APACHE KAFKA

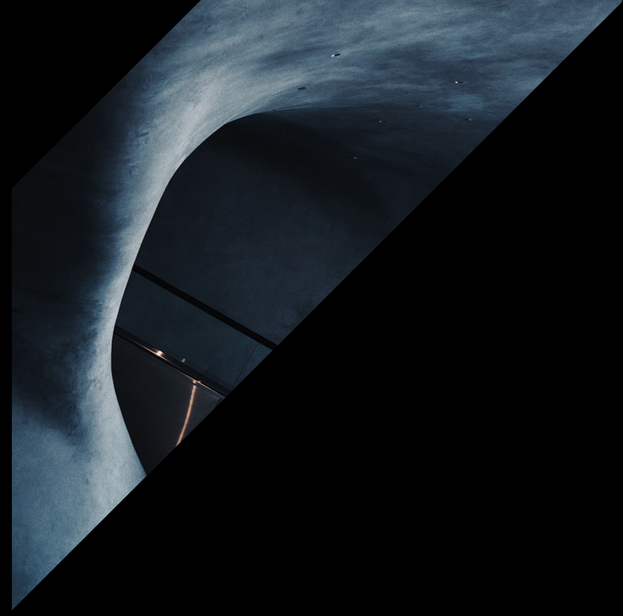
Kafka is a backbone of the whole platform. We’re using it as a message bus for all of the events that happens across the platform. It’s a fast and scalable solution to synchronize changes between microservices and to avoid direct calls between them. At the same time it is a great extension point: external modules could listen for events in the bus and react.

04. REDIS

Redis is also important part of the platform. It stores user-related and session-related information as well as used as a shared session between different instances of the services or applications. Since it’s in-memory with persistent storage it’s a good choice to store frequently accessed data (like session-related values).

MEDIA ENGINES





MEDIA PROCESSING ENGINES

To support as many file types as possible in terms of preview and built-in metadata LAIKA using various media engines for different types: **graphical** (ImageSharp, ImageMagick, Skia, exiftool), **document** (Microsoft Office, GhostScript), **video** (FFMPEG) and **3D** (GLTF2). Integrations with other engines to support more file types is possible and available.

ARTIFICIAL INTELLIGENCE

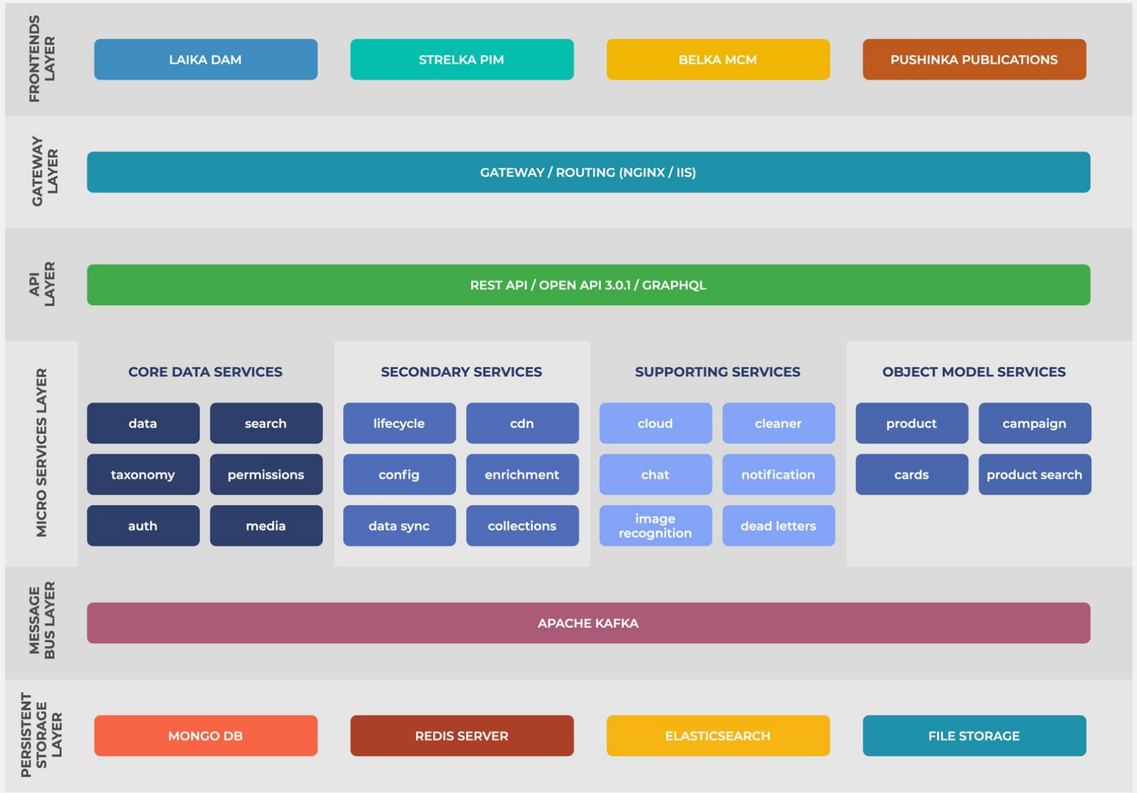
LAIKA using AI engines to enrich content with additional metadata, identify objects on images or to generate smart AI-based crops. It's possible use **cloud services** (like Google Vision or Azure Cognitive Services) for metadata enrichment or use **built-in AI engines** for smart crops, similarity analysis, duplicate analysis and quality checks.

CLICK TO CONTINUE

PLATFORM DESIGN

Main solution design goals were: scalability, maintainability, extensibility and performance during usage of our platform as a part of enterprise environment with large amount of data.

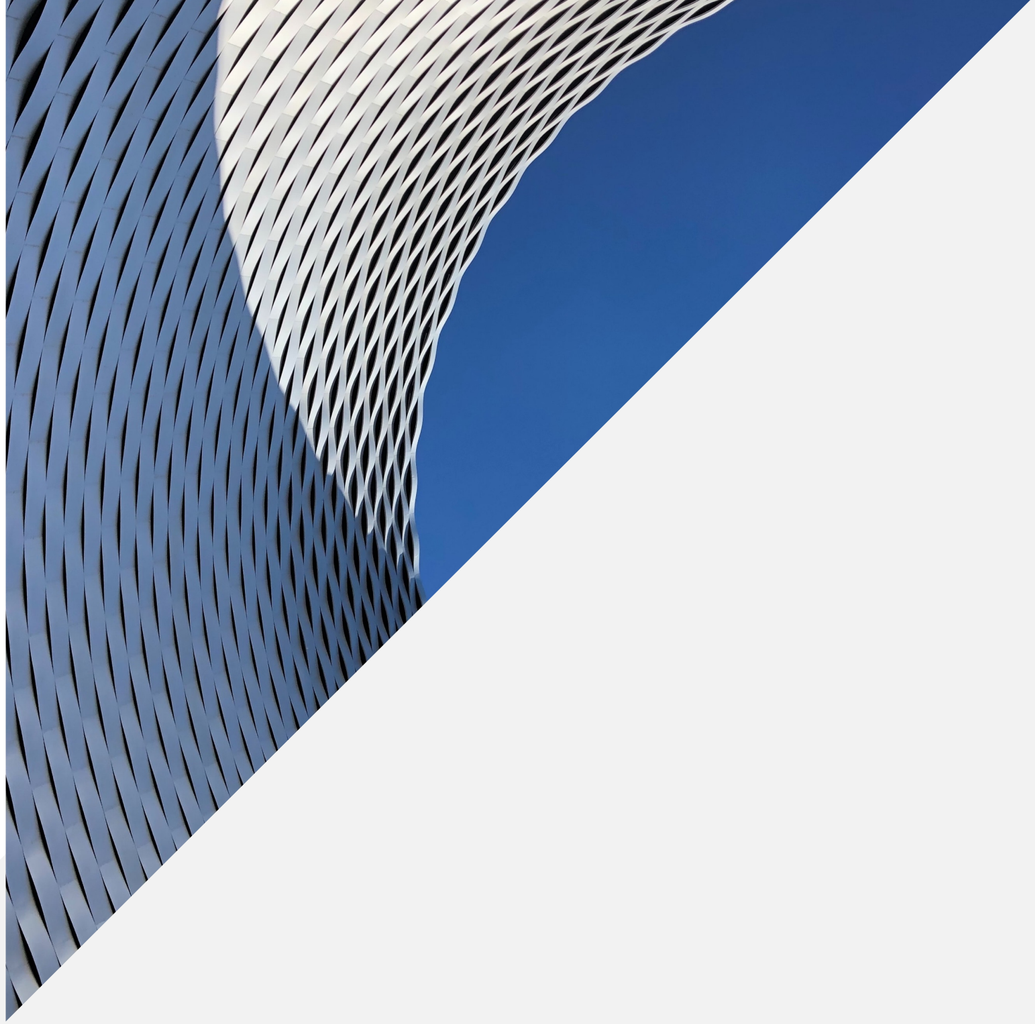
Dzmitry Dounar, solution architect



PLATFORM DESIGN

Our platform implements microservice architecture as a key principle of the design. Key components represented as layers.

It means that platform divided to a set of independent microservices which are responsible for a separate part of functionality, so each component could be implemented, scaled or changed independently. Such approach is applicable for platform extension: it’s possible to implement new microservices to cover new functionality.



API FIRST

LAIKA is API-first or headless. It means that **all of the platform functionality is available via backend calls**. It also means that the most of the business logic is also located on backend, so to implement new UI application you just need to call backend to execute some functionality or perform some actions. With such design UI application simply gather and displays data (so it’s quite fast to implement a new custom UI).

HOW TO?

HOW TO CONFIGURE

LAIKA has it’s own configuration service that stores all of the settings that’s related to the platform execution. It’s available via API and also has a convenient UI for administrative purposes.

HOW TO EXTEND

Usually the new functionality should be implemented as a new microservice with its own logic. For example, if you want to combine assets to some object like publication and work with it you’ll create a separate microservice that describes such object.

HOW TO INTEGRATE

To integrate a new component to LAIKA platform you could just call some methods from LAIKA APIs directly (that might be useful in some cases, like store the file or create new asset) or integrate with Apache Kafka to listen for some events.

CLICK TO CONTINUE

REST API

Whole backend implements HTTP REST so it could be used from **any programming language** that support REST calls (like Java, Ruby, Python, React.js, Angular.js, Javascript etc.). So you could create your extensions and UI applications on any programming language and framework that you prefer. **API is self-documented** according to **OpenAPI 3.0** standard: all endpoints have built-in Swagger UI with description of methods and arguments.

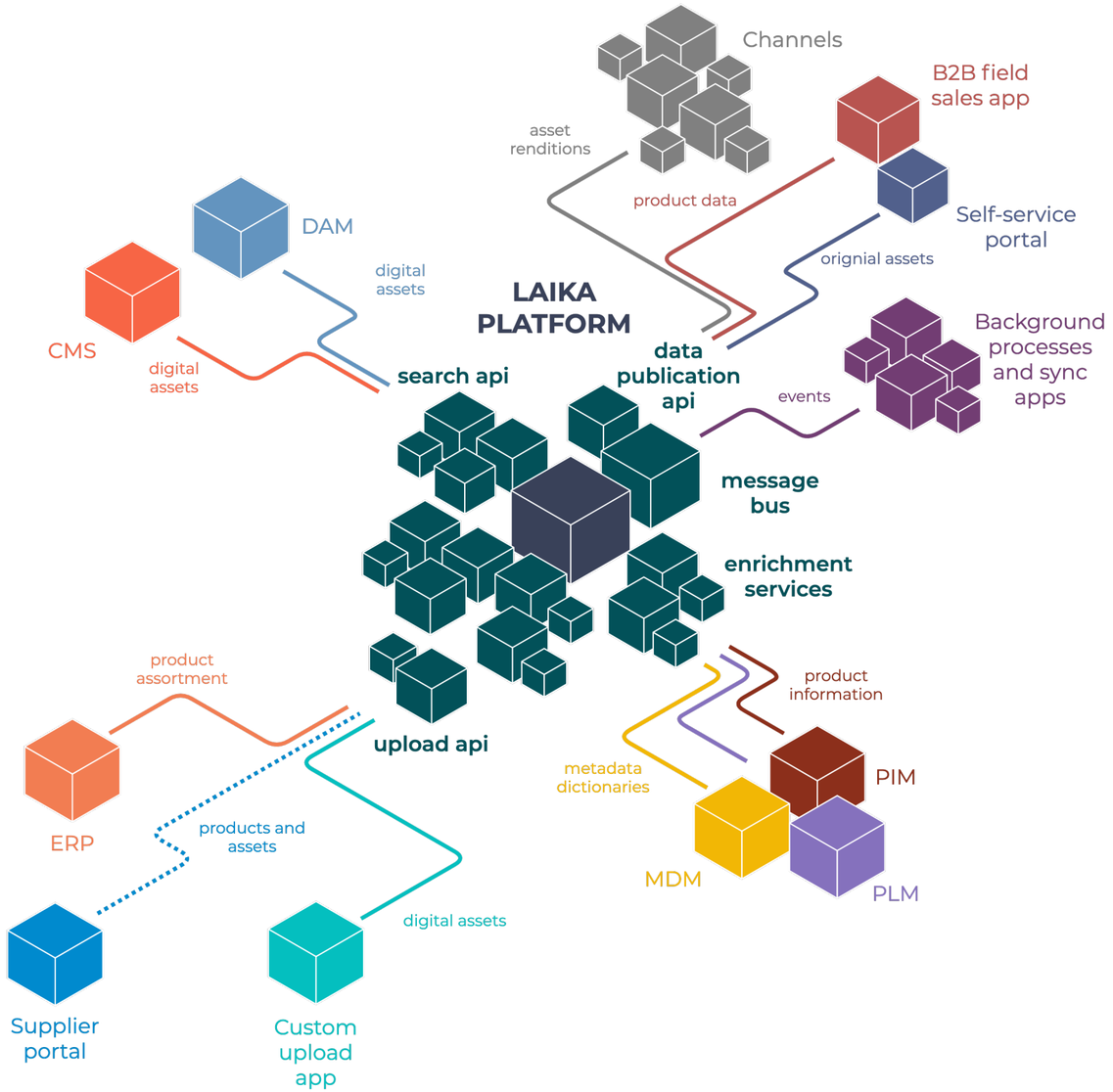
CLICK TO CONTINUE

GRAPHQL

GraphQL is used for the communication with search microservice and allow to build search queries in simple and efficient way.

GraphQL is a modern query language for REST API implemented initially by Facebook that allow you to define the query and response format in the request which is ideal for the integration with LAIKA search subsystem. LAIKA has built-in UI for query creation and testing.

CLICK TO CONTINUE



INTEGRATION CAPABILITIES

It’s easy to integrate LAIKA into existing ecosystem: the whole platform designed having in mind such cases.

Diagram on the left side of the slide demonstrate two main approaches for the integration: direct calls via REST API (so custom applications could be implemented using any language or technology), or by connecting application to message bus (Apache Kafka) to use event-based integration model.

TECH AGNOSTIC

REST API COMMUNICATION

LAIKA is open for integrations and extension using any programming language or framework that support REST calls. So it's not important what tech stack used by LAIKA itself, you could create new modules or applications on **any programming language or framework** (like Java, .NET, Python, Django, Ruby, Scala, React.js, Angular.js or any other). All of the communication goes via HTTP REST protocol and have no language dependencies. Even other platform components such as databases or storages have connectors to the most of the popular programming languages.

EVENT-BASED MODEL

It's also easy to integrate any custom applications into LAIKA ecosystem by implementing event-based communication. For this purpose custom application should be connected directly to **Apache Kafka message bus** to send and receive events. It also could be done from **the most of the modern programming languages**. Kafka supports drivers for C++, Python, Go, Erlang, .NET, Ruby, Node.js, Java, Perl, PHP, Rust, Scala, Clojure, Swift and many others via REST or even stdin/stdout. So you could create your own module or application to track all changes from LAIKA platform and handle them in your application.

CLICK TO CONTINUE

WEB API SECURITY

LAIKA REST API using different authentication model for API communication that in case of end-user authentication. Authentication could be performed using one of the standard Web API security protocols: **Basic** (using of Authorization header for requests), **JWT** (token-based authentication) and **HMAC** (hash-based authentication), so it’s possible to select the most suitable approach. At the same time all of the requests will be registered with some Requestor ID for logging and audit purposes.

SECURE

LAIKA has built-in mechanisms for user management and permissions that allow to build extremely complex data structures (including overrides and inheritance). At the same time it’s possible to connect LAIKA login with any **SSO** provider that implements **OAuth 2.0** protocol. Each storage also use built-in authorization and authentication mechanisms. All user-related data (like name or email or address) is encrypted to meet GDPR requirements.

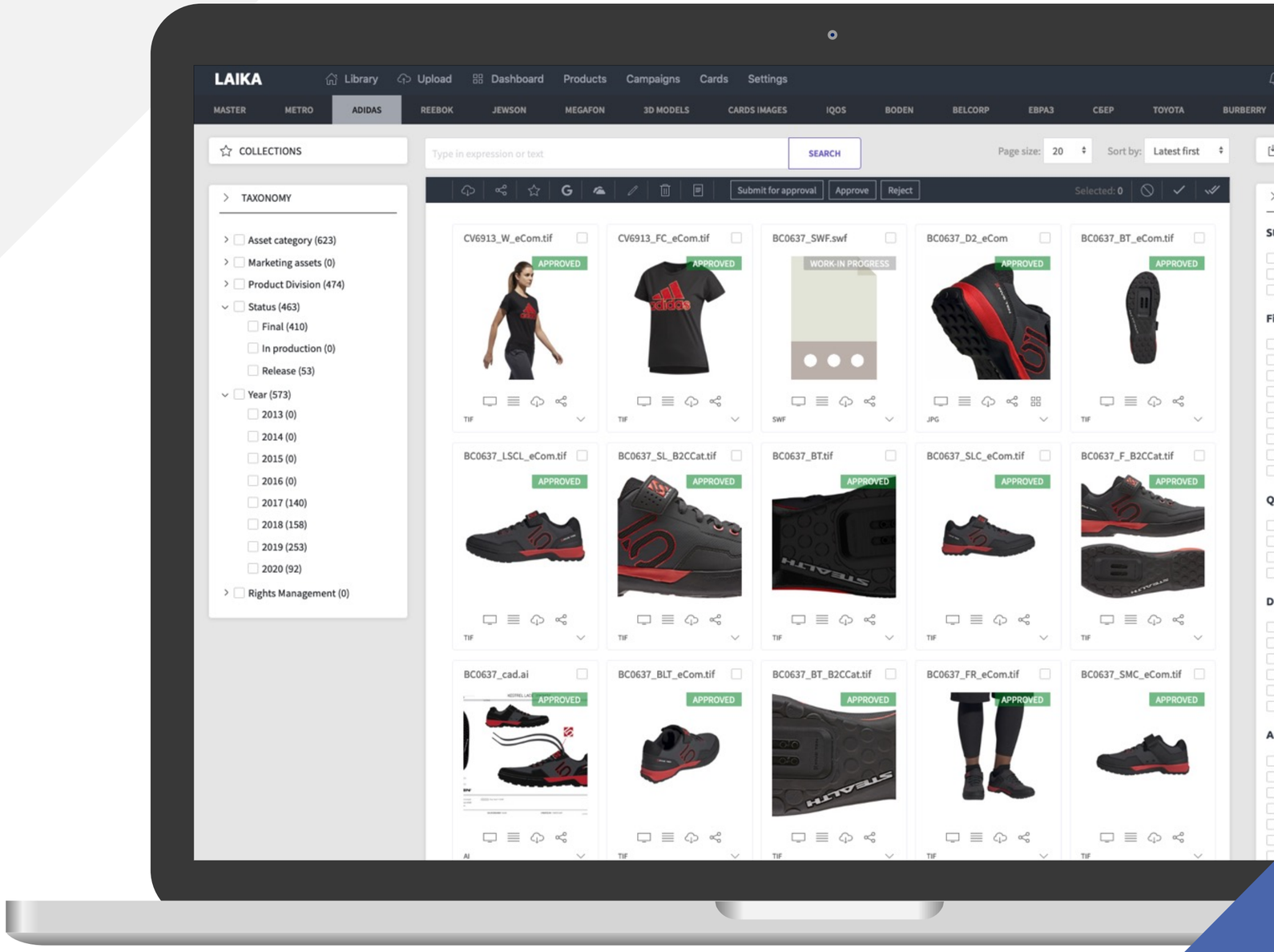
PERSONALIZED USER INTERFACE

REFERENCE APPLICATIONS

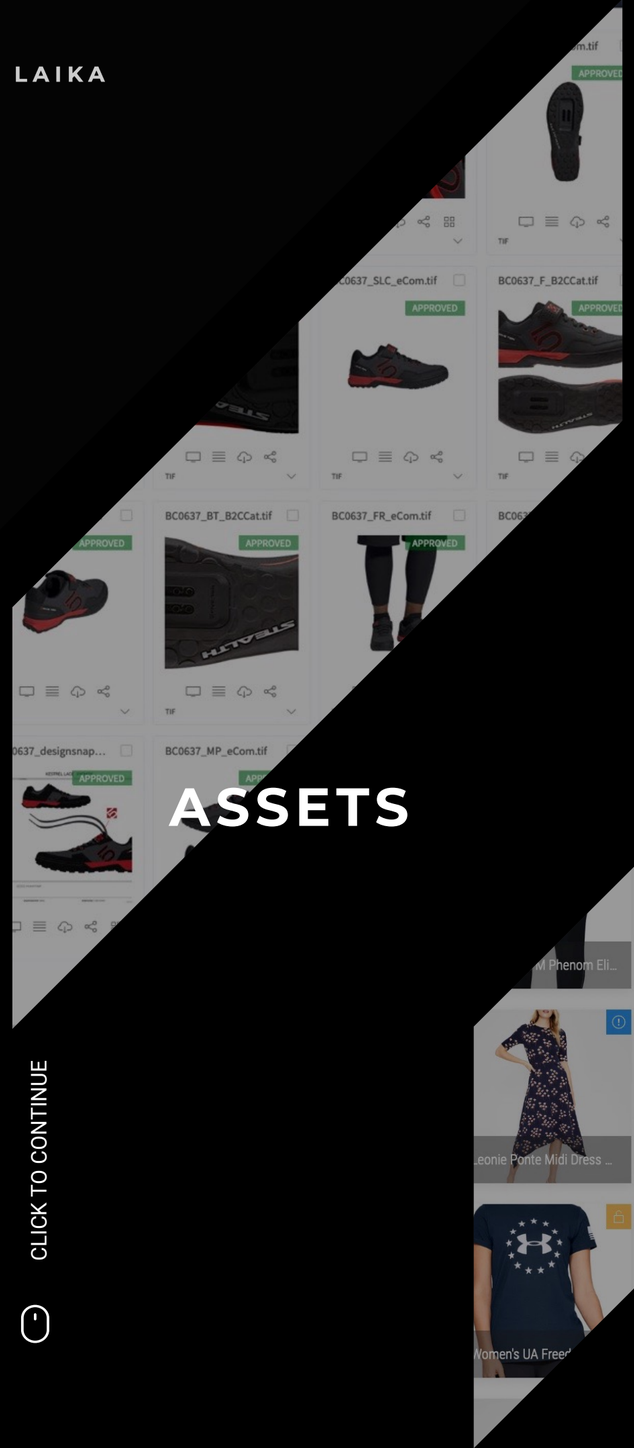
LAIKA platform comes with a set pre-built reference applications that provides functionality available on backend. These applications covers wide range of activities: LAIKA (asset management), BELKA (marketing campaign management), STRELKA (product asset management), PUSHINKA (digital publications design).

CUSTOM APPLICATIONS

Since LAIKA is a headless platform you could always create a new custom UI without big effort: the most of the business logic and features concentrated on the backend, so UI just displays and provides the data. Sometimes the best solution is to implement a simple lightweight UI for some user role rather than onboard these users to a big complex application. It could be done on any programming language or framework (Java, .NET, React.js, Angular.js, Ruby, Django or other).

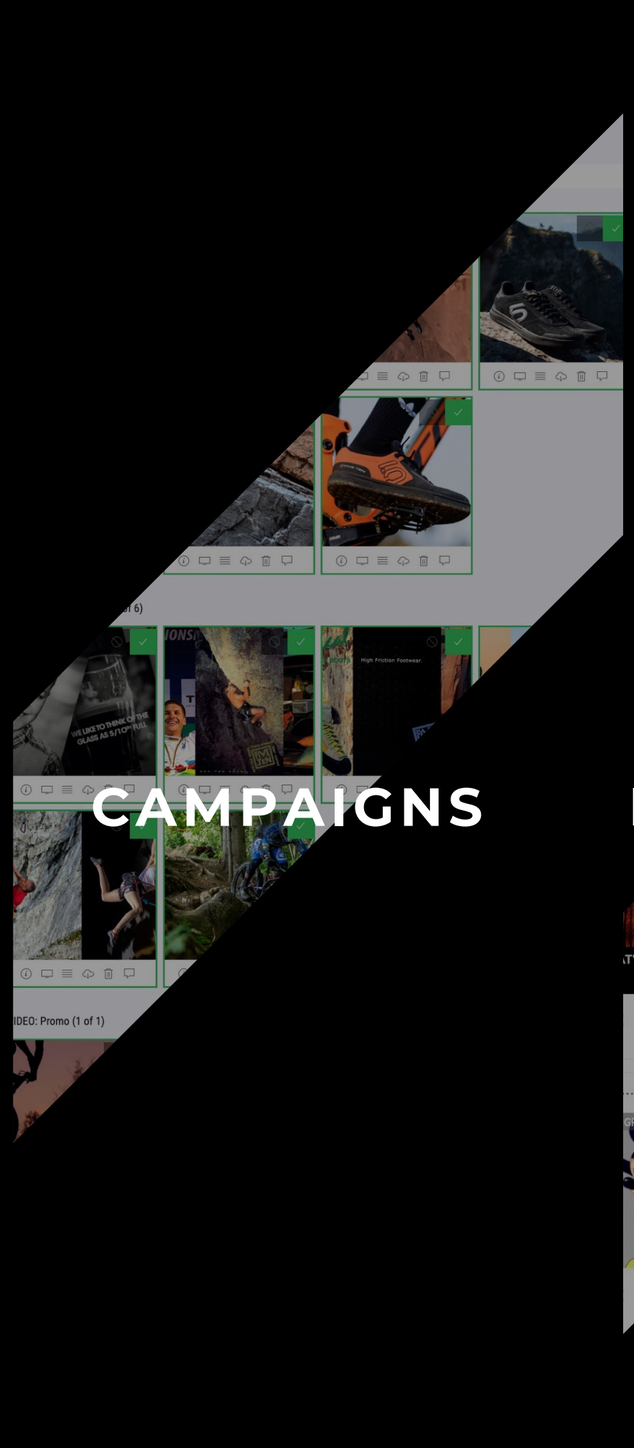


CLICK TO CONTINUE

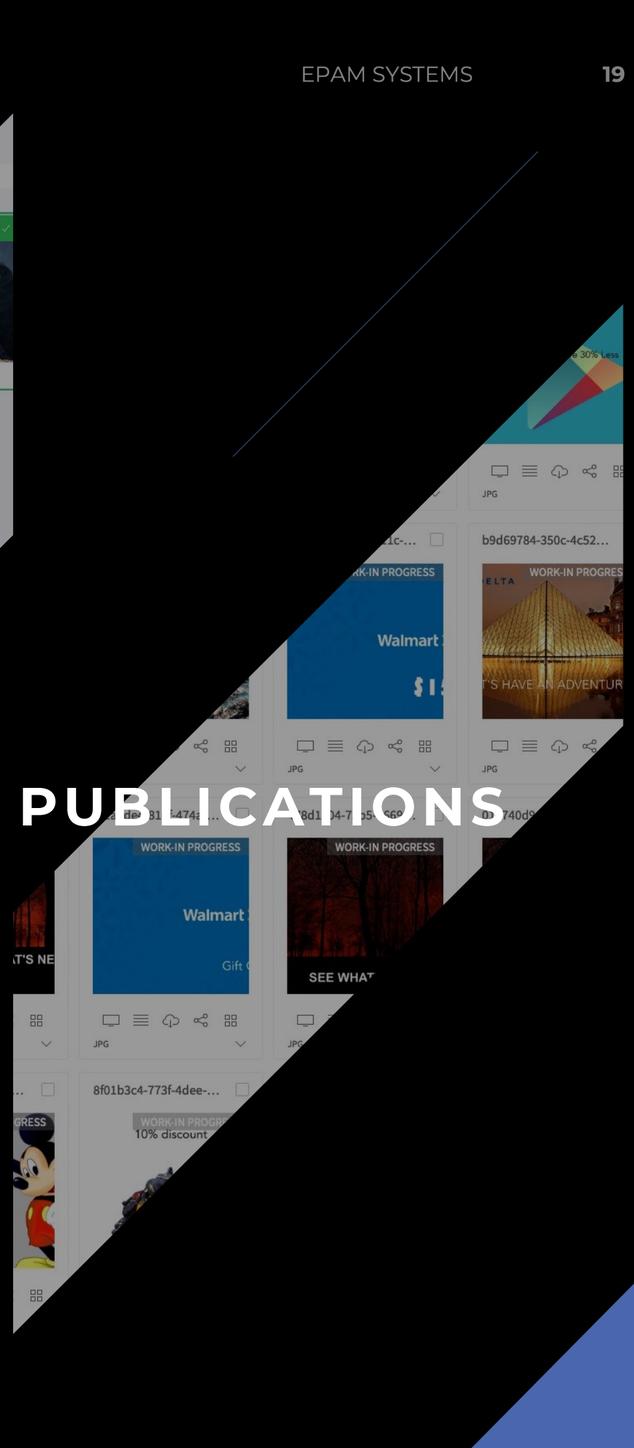


ASSETS

PRODUCTS



CAMPAIGNS



PUBLICATIONS

CLICK TO CONTINUE

HARDWARE

Platform agnostic / Cloud agnostic

LAIKA is platform agnostic and cloud agnostic. It could be hosted on Windows, Linux and MacOS. So you could deploy LAIKA almost anywhere: on-premises, on cloud VMs or in hybrid scenario. We minimized references to cloud services so LAIKA could be hosted in almost any cloud: your private cloud, Amazon Web Services, Google Cloud Platform or Microsoft Azure (as well as in other conditions). Our team tried to keep platform as flexible as it possible but if needed it’s possible to connect platform to use more cloud services for storages or databases etc.

VM deployment scenario

You could simply deploy LAIKA to Windows or Linux virtual machines together with all storages and databases, just copy the files. You could use only 2 VMs (of AWS m5a.large shape or 2CPU and 8GB RAM together with disk for your asset storage) and you could have LAIKA up and running. In this case one machine will be responsible for the hosting of the application while the other will be responsible for storages and databases. For sure, it's a very cheap and simple installation but still might be useful for development environment or for usage of LAIKA for some small group of users.

Auto-scaling cluster deployment scenario

In opposite, LAIKA could be deployed as a part of Kubernetes-managed auto-scaling cluster. It's a preferable solution for real-life production usage with large amount of users and large amount of data. It's also recommended for the situation when application will have some load from time to time so it could be scaled appropriately. For this case we have Terraform-based IaaC approach to create and manage cluster while LAIKA is represented as a set of Docker containers under Kubernetes control. For sure, it's more expensive scenario but it's ready for any challenge.

XCOPY deployment

You could just manually copy LAIKA applications to some location and map IIS (under Windows) or NGINX (or other web server under Linux) to have LAIKA up and running. Just as simple as that. Storages should be installed manually in this case, but it’s a one-time operation and this process is well-documented.

Automated deployment

Since it’s possible just copy LAIKA application files to have it up and running it’s possible to automate this process using any scripting language that is applicable for your case. It’s also easy to integrate LAIKA to your existing CD pipeline to automate the process.

Infrastructure as a code

For real-life usage we propose to use a different approach. LAIKA team is using Terraform scripts that fully describes and maintain infrastructure as a code, so if any changes needed you just change values in Terraform configuration files and then apply these changes by executing the script again. All the infrastructure including storages will be maintained automatically in this case.

Designed for real-life usage, so all of the maintenance activities simplified as much as it possible.

LOGGING

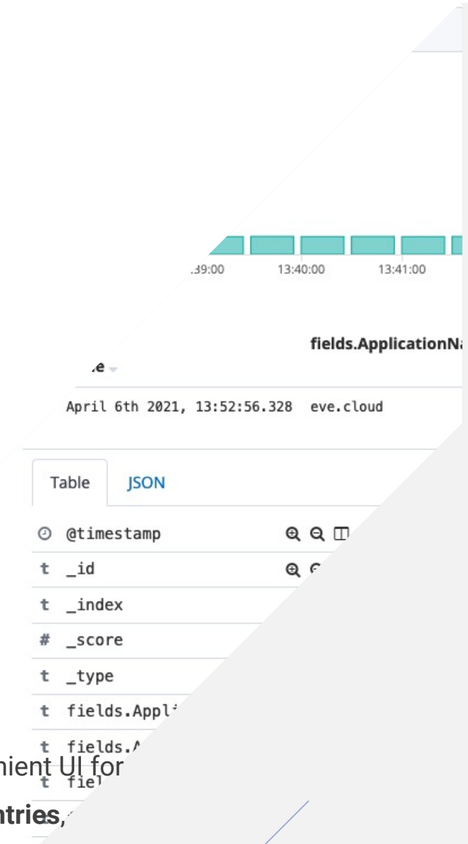
All logs platform-wide collected, stored in Elasticsearch and available via Kibana with powerful features such as filtering, sorting and real-time display. Each log entry has Requestor ID and Correlation ID to identify the call chain.

MONITORING

LAIKA has built-in telemetry that provides insights not only from machine stats but also from the application itself, powered by Prometheus and displayed in Grafana.

BACKUPS

In case of LAIKA backup procedure in the most cases a just file-based backup that could be done by using some simple procedure as rsync or using some advanced services.



KIBANA

Kibana provides a convenient UI for navigation through **log entries**, searching and filtering. You could create dashboards, charts or see the events in real-time.

GRAFANA

Grafana displays the **telemetry** data provided by LAIKA platform via **Prometheus** to display it as a various charts on the dashboards. You could see the system performance in real-time.

PERFORMANCE

NUMBER OF ASSETS	**6M+ assets**
Number of users	9K+ users worldwide
Disk space consumption	15TB+ of data
Multi-regional usage	24 countries

This is just a sample of platform performance based on the real-life usage experience by **one of the largest retailers in the world**. Platform is scalable, so if you need store more data it’s possible and only hardware dependent.

CLICK TO CONTINUE

OUR KPIs

THROUGHPUT	137rps max. / 71rps avg.
Uploads per hour	4 000
Publishes per hour	3 000
Searches per hour	9 000
Asset previews per hour	20 000
Downloads per hour	2 0000

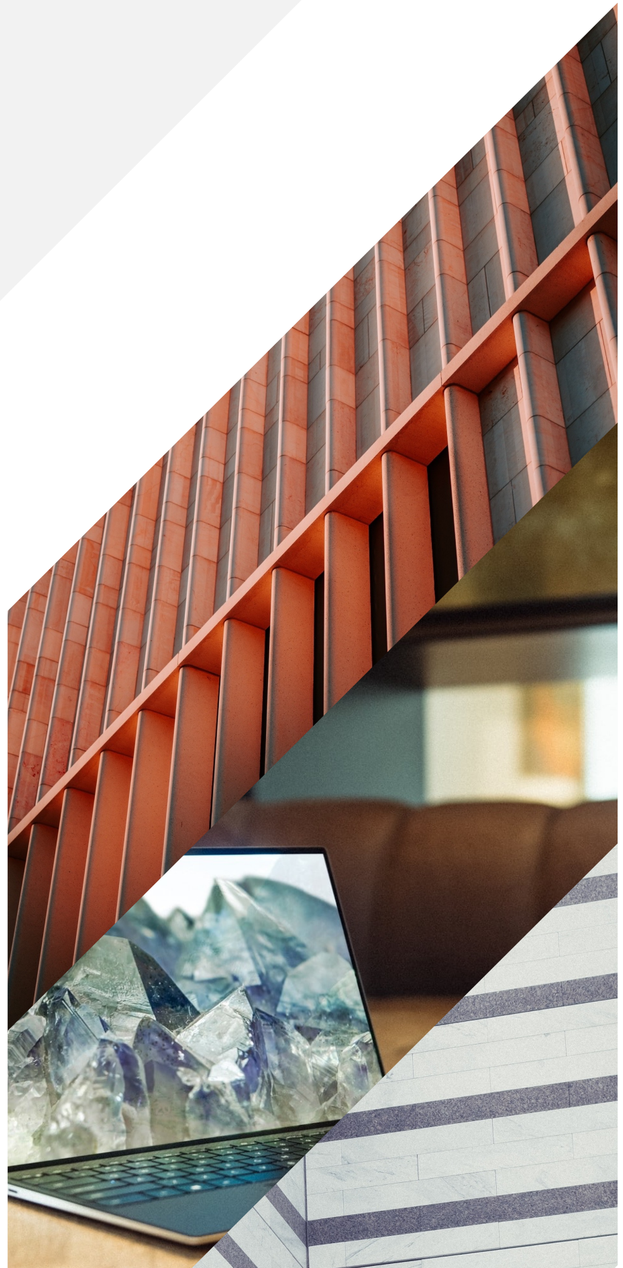
The numbers above are done based on our **standard performance test** for the condition similar to the previous slide for the scenario that includes 40% of upload / download operations and 60% of asset management activities. Since platform is scalable it’s highly dependent on a hardware so **better results are possible** if needed.

CLICK TO CONTINUE

UNIQUE FEATURES

001 Virtual taxonomies

It’s possible to define a complex taxonomy tree that will cover all use-cases, but to display users only small subset of such tree for better user experience.



002 Security model

Permissions model supports inheritance via taxonomy tree but with various override cases: like for example priority-based overrides for exceptional case definition.

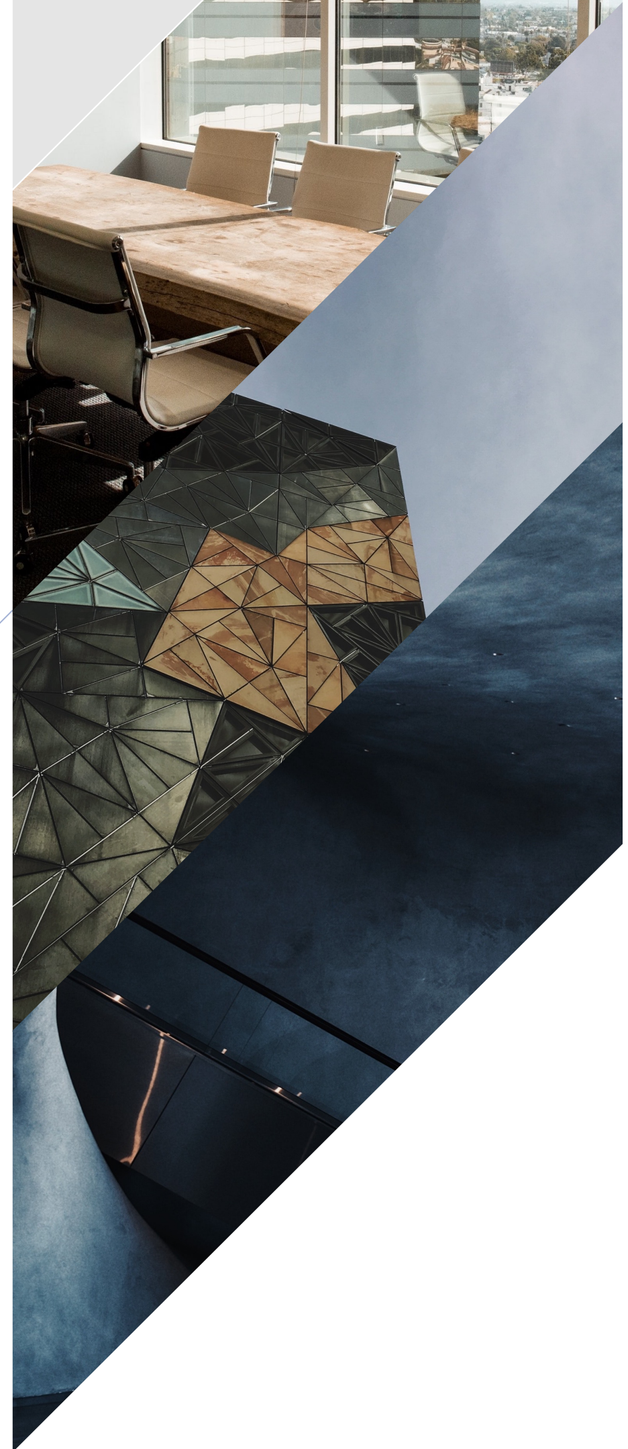
003 Extended file format support

There is no limitations on uploading various file types to LAIKA. For most of them like images, videos, RAW files, documents and other preview will be generated.

CLICK TO CONTINUE

004 Audit subsystem

LAIKA has built in audit subsystem that tracks all actions performed in a system in stores it in blockchain-based storage that guarantees that this data will remain unchanged.



005 Asset delivery

Using CDN sync subsystem LAIKA could generate various resolutions of assets and then supply it to different cloud storages with CDN on top of them to support omnichannel delivery case.

006 Bulk edit via Excel

Usually bulk edit functionality implemented as a set of web tables, but why not to use all the powers of Excel instead? LAIKA will create an Excel sheet that could be downloaded and updated but there is no need to upload it again: once you click save all of the changes will be applied to LAIKA directly.

THANK YOU

We hope that this presentation was useful and provided answers that you looked for. If not, please don’t hesitate to contact us.

YOURS LAIKA PLATFORM TEAM