API Economy in Leroy Merlin

Sergey Lega, Leroy Merlin Vostok
Sergey Lega
Enterprise Architect
Leroy Merlin Vostok

Projects:
API Platform, Service mesh
API Economy is combination of value driver which business can get out of APIs usage

Boost your product strategy and innovation
- Standardized communication with APIs
- Create effective communication between teams and software components
- Boost TTM for new features / product -> benefit your product strategy.

Unleash additional revenue streams for your existing assets
- Let other market participants use your data to enhance their offering
- Extend your offering: API is another way to expose your products or services

By the end of the day, it’s a platform enabler
Allow 3rd party developers/partners to build their solutions in top of your existing products
<table>
<thead>
<tr>
<th>Standardization of interaction between product teams</th>
<th>We need to boost TTM for frontend applications</th>
<th>We need to securely expose our APIs</th>
<th>We have a lot of legacy apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Multiple teams with asynchronous release cycles*</td>
<td>* SDLC of Mobile / web app should be detached from APIs*</td>
<td>* We have partners and clients who use our API*</td>
<td>* Legacy apps have limited customization options*</td>
</tr>
<tr>
<td>* One application use multiple served by different product teams*</td>
<td>* Sometimes mobile / web app needs to perform orchestration of APIs*</td>
<td></td>
<td>* Legacy apps frequently use outdated protocols / formats*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
API Platform high-level components

**Management console** – Provides GUI and administration facilities to control the API platform behavior across the organization and get access to monitoring & analytics data.

**Developer portal** – Web application, point of communication between API consumers and API providers. Stores complete API documentation, provides ability to register consumer application and subscribe to API products.

**Task management module** – Provides clue between API management and release/task management tools such as JIRA, release system (custom) and so on.

**Service mesh** – application network based on sidecar pattern, enforces the point to point managed communication between microservices running on PaaS platform.

**Orchestrator** – Experience API layer. Provides ability to configure channels API experience from existing microservices by mashups and data transformations.

**Edge Gateway** – expose APIs outside the organization, perform data filtration, ensure security, rate limitation, logging and provide basic API transformation capabilities.
API Platform high-level components

- Dev portal
- Admin console
- API
- Task management
- Apicurio
- JIRA
- API Gateway
- Orchestrator
- Legacy
- Microservices

Leroy Melrin Vostok
API Platform high-level components

- Dev portal
- Admin console
- Apicurio
- Task management
- JIRA
- API Gateway
- Orchestrator
- Legacy
- Microservices

- Standardization
- Security
- Legacy apization
API Platform high-level components

- Dev portal
- Admin console
- Task management
- Apicurio
- JIRA

API Gateway

- Orchestrator
  - API Orchestration

Legacy

Microservices

- API Orchestration
API Platform high-level components: edge gateway

- Transformation (SOAP<>REST, XML<>JSON, modification of body/headers)
- Security (Quota, Rate limit, threat protection, etc)
- Service discovery & conditional routing
- Analytics collection
- Tracing
API Platform high-level components: task management & automation

- Dev portal
- Admin console
- Apicurio
- Task management
- JIRA

- Subscription
- Documentation
- User guides, blog, etc

Orchestrator

Legacy

Microservices

Leroy Melrin Vostok
API Platform high-level components: task management & automation

- Apigee operations
- Analytics, tracing, debug

Dev portal
Admin console
task management
JIRA

Apicurio

API Gateway

Legacy

Microservices

Leroy Melrin Vostok
API Platform high-level components: task management & automation

- Dev portal
- Admin console
- Task management
- Apicurio
- JIRA
- API Gateway
- Orchestrator
- Legacy
- Microservices

- Workflow automation (Task in JIRA, RBAC, so on)
API Platform high-level components: task management & automation

- Design APIs
- Collaboration

Dev portal
Admin console
API Gateway
Orchestrator
Legacy
Microservices

Apicurio
JIRA

Leroy Melrin Vostok
## Governance process for master systems

### Canonicals

1. API intended for publication on Dev Portal
2. Designed in accordance with the requirements of the company Leroy Merlin
3. Grouped by canonical objects on DevPortal

### Experience API

1. APIs developed by the “for themselves” team (including the orchestrator)
2. Not intended for publication on Dev Portal
3. Used by one team only.

### Rules and limitations

#### Canonicals

1. API must be compliant check list
2. API must be defined to a canonical object
3. API must be challenged by Integration Architect
4. If the API is published on the Internet, then it should be checked by the IT security department.

#### Experience API

1. Orchestrator APIs cannot be published on DevPortal
2. If the API is published on the Internet, then it should be checked by the IT security department.
Governance process for master systems

Publish

- Publish to dev
- Update / create docs in Apicurio
- Approved by SOA Arch?
  - No
  - Promote to test / preprod
  - Tests ok?
  - No
  - Upload docs to Dev Portal
  - Notify subscribers
  - Approved by Security?
  - Yes
  - Promote to external prod
  - No
  - Promote to internal prod
- No

Subscribe

- Navigate to Dev Portal
- Find API, select quota, subscribe
- Approval type
  - Manual
  - Auto
  - Approval by API Product owner?
  - No
  - Receive key / secret
  - Yes
API Platform high-level components

- Mash-up logics (backend for frontend)
- Analytics
- Tracing
- Autotesting module

Leroy Melrin Vostok
Orchestrator

Digital App
And its team

Front-end specific logic (mash-ups)
Back-end for front-end

Drag & drop GUI
Detailed tracing module
Monitoring & alerting module
Automated testing module

Provided out-of-the box as a packaged solution

Back-ends
And its APIs

Orchestrator

Leroy Melrin Vostok
Channel receive the complete pack of containers and configurations on PaaS environment – prod/development ready

Dev/Test Env

Prod Env

CI/CD

- GitLab
- Jaeger Tracing
- Jenkins
- Orchestrator
- Monitoring & Alerting
- Test system

Project Pipeline
Endpoints & Policies
Tests repo

Leroy Melrin Vostok
Demo time
Orchestrator UI demo

Orchestrator UI
Orchestrator Tracing demo

Tracing Demo
Orchestrator Analytics demo

Analytics Demo

Leroy Melrin Vostok
Orchestrator use-case: Store mobile

- Products (availability, order, cost of a project, …)
- Services (availability, SLA, terms, contractors, …)
- Operation with customers and their orders
- Analytics (plan/fact, sales forecast, …)
- HR (RBAC, org structure, colleagues, …)
- Warehouse and service operations (cross-docking, bills, invoices,…)

- All business logics lives in orchestrator

- All data objects live in backends

Leroy Melrin Vostok
Decreased TTM for new projects and features

1. Create new API
   x3-x4 TTM reduction through usage of Orchestrator

2. Search for docs & get access
   No time wasted on investigation, actual docs are always published in dev portal

3. Publish API
   Standardized process of publication and security approvals, SLA – 2 hours
Improved stability

1. Investigate fault in application

   All inbound / outbound requests are logged in orchestrator with full context
   All requests through API GW are logged as well

2. Limit SLA per application

   All subscribers to API are known, for each of them there is an SLA

3. Identify “problem” interfaces

   Since all API traffic goes though API GW, thus troubled interface are known through analytics

Leroy Melrin Vostok
Improved operational efficiency

1. Redundancy elimination
   - Redundant components and interface are seen, so they could be eliminated

2. Orchestration is much cheaper than microservice
   - Infra ~ approx the same, but without overhead on support and DR
Additional revenue streams

1. APIs for large B2B clients

Integrate LM RU APIs into internal information systems of large B2B customers
Thank you!

Email: api-platform@leroymerlin.ru,
      sergei.lega@leroymerlin.ru
https://www.facebook.com/lega.sergey
Telegram: @slega