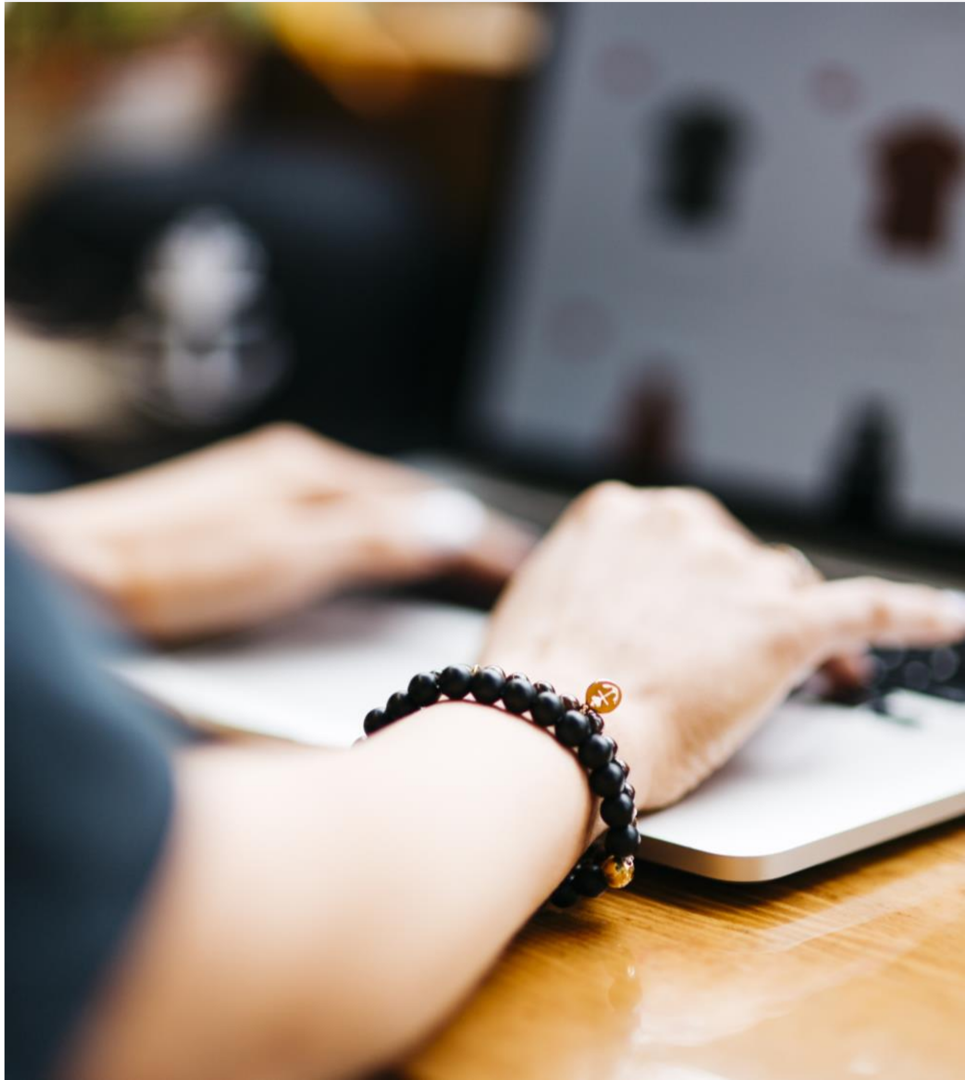
PayPal System Architecture

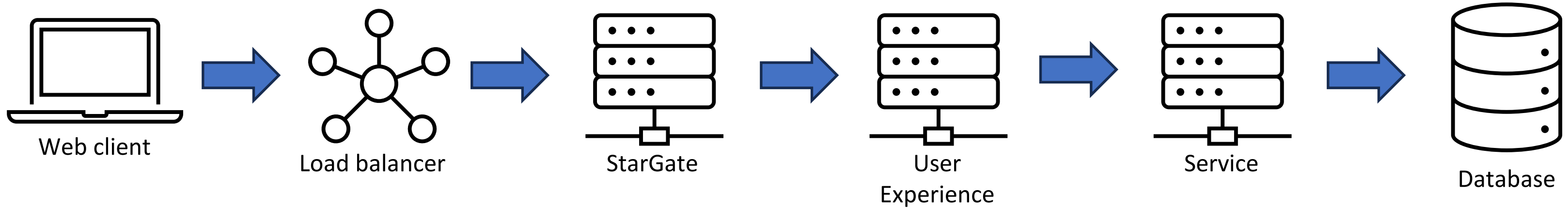
Customers

How do customers access PayPal services?



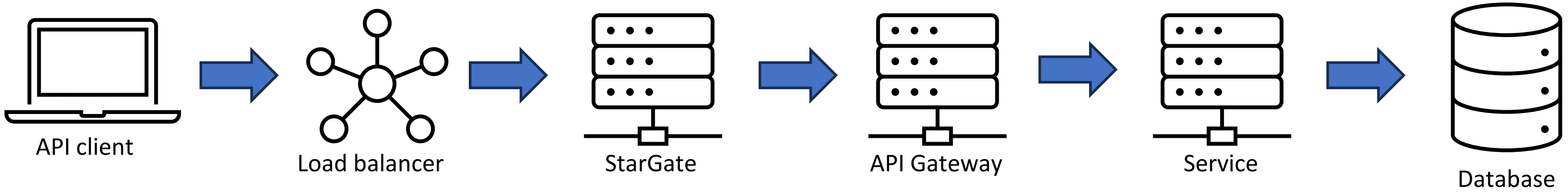
PayPal System – Web clients

How do browser clients work?



PayPal System – API clients

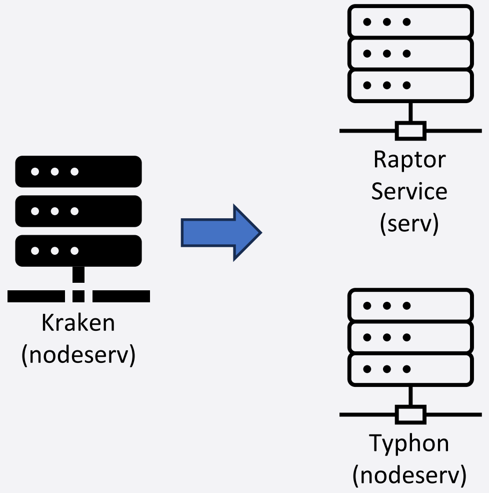
How do API clients work?



Application Types

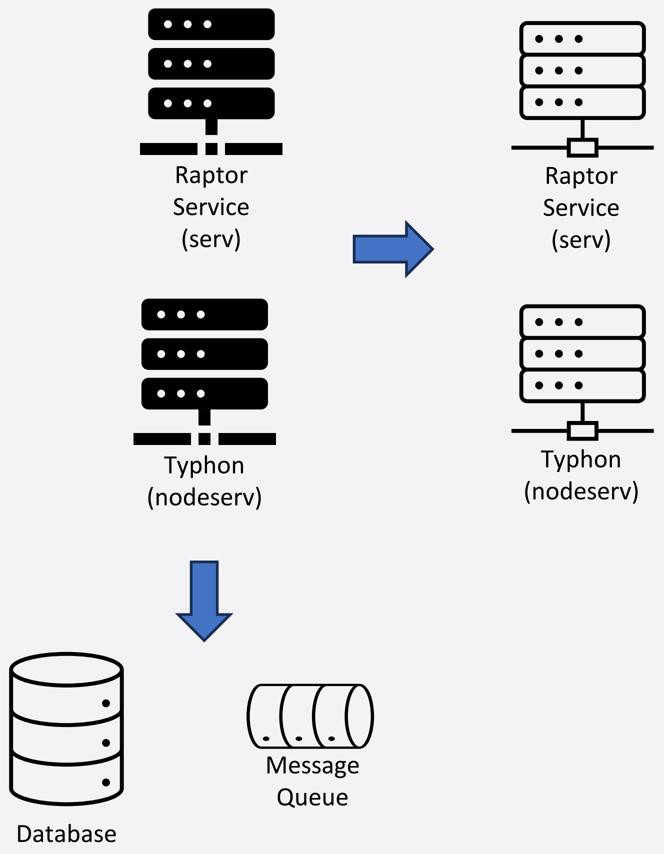
Application Types

Experience services for handling user presentation



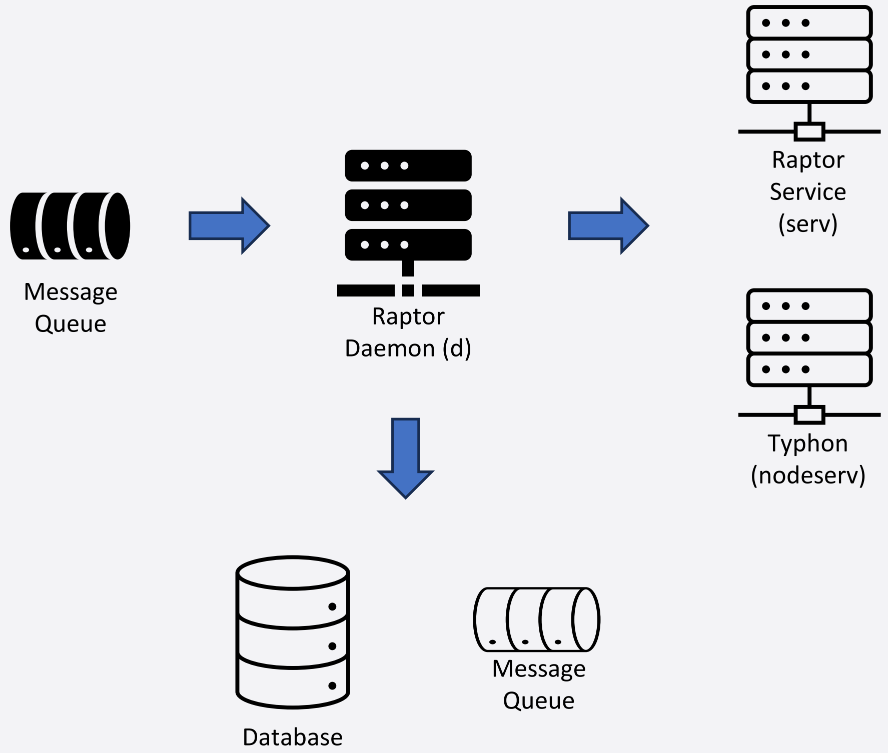
Application Types

Services core business logic



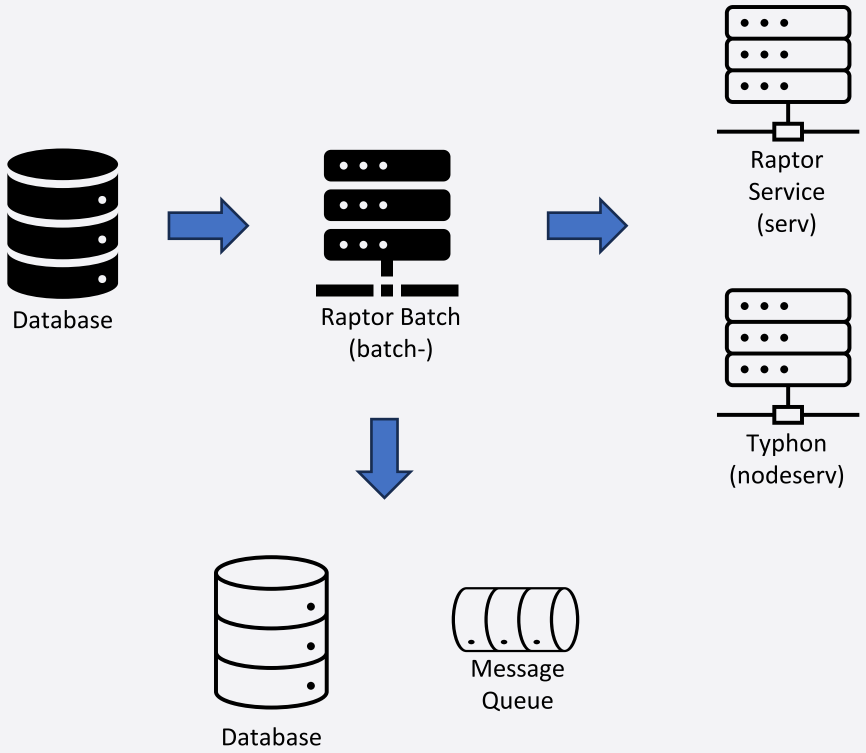
Application Types

Message daemons for asynchronous processing



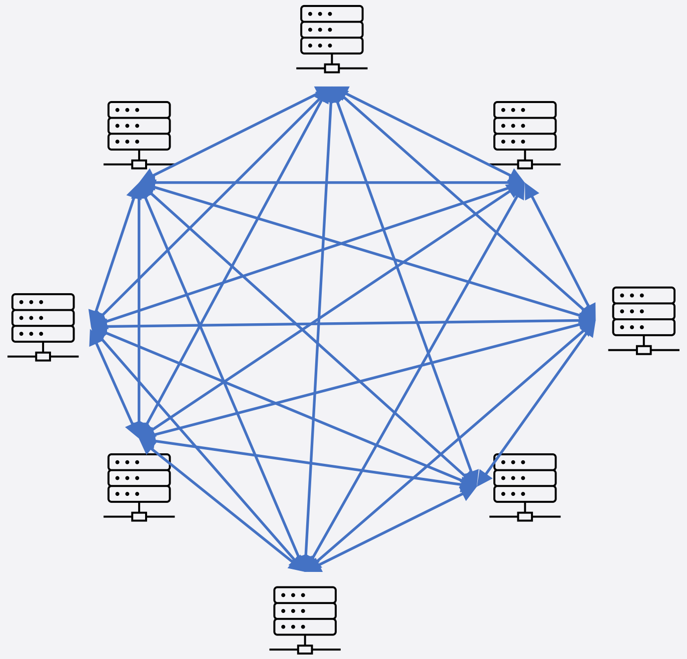
Application Types

Batch services for processing bulk data

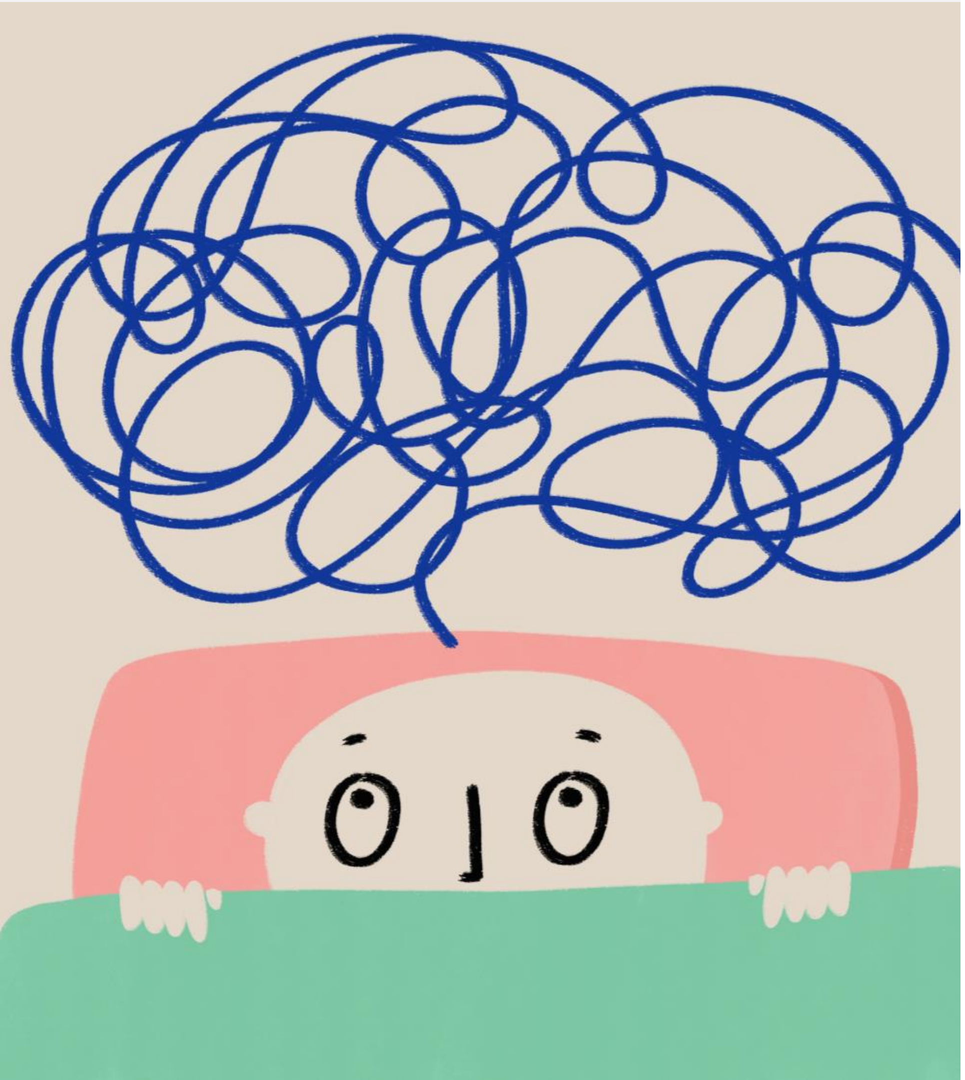


Microservices

Worst case microservice architecture dependency graph



Enterprise Architecture



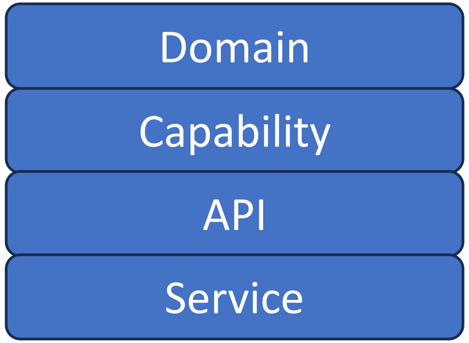
Enterprise Architecture
The system is so complex, how can you understand it?

Enterprise Architecture

Decomposition

Decompose the system into parts that are easier to reason about

- Domains – Group of related capabilities
- Capabilities – Business functions that provide specific value
- APIs – Interface definition for a subset of a capability
- Services – Implementation for an API

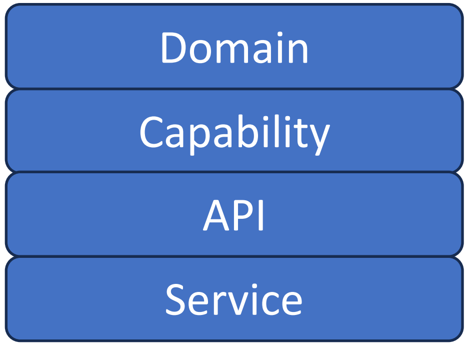


Enterprise Architecture

Usage example

Once the enterprise is decomposed into domains, capabilities, APIs and services, it is now possible to understand the system at different levels of granularity.

- Domain level – What domains depend on each other? Use the capabilities to understand **how** they depend on each other.
- Capabilities – What are the business functions provided by a domain?
- APIs – How are the system components integrated?
- Services – What is the physical architecture of the system? Where does the business logic live?



API Types

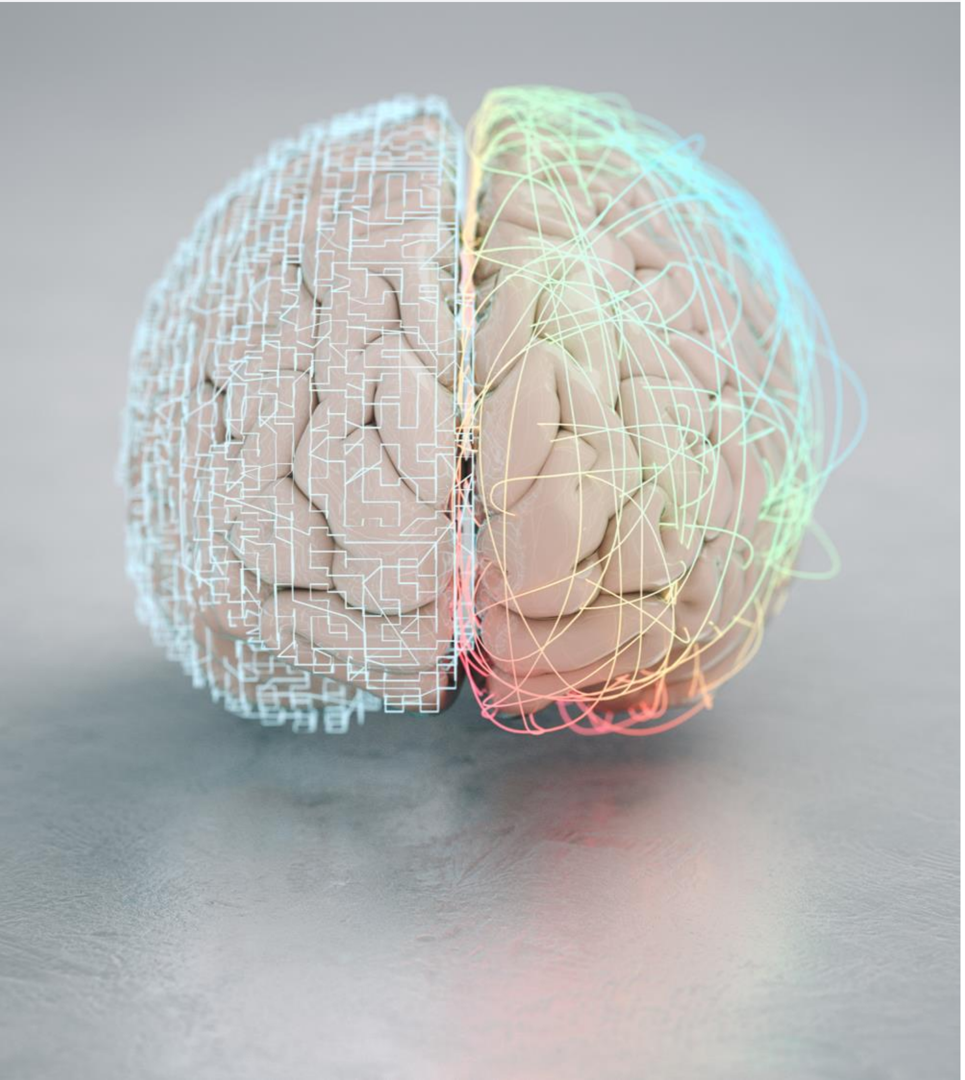
What are the different styles of API?

1. REST – API style that leverages www patterns. Represents resources of the system as nouns in the HTTP path and operations on those resources as HTTP verbs, e.g. **/accounts POST** – create an account. Written using the OpenAPI 2.0 (soon to be OpenAPI 3.0) standard. Used for synchronous calls.
2. GraphQL – Represents resources of the system as a graph. Structure of the returned data driven by the client. Mainly used for user experiences.
3. SOAP – Expose free-form operations using an XML documented format. Used for synchronous calls, has mostly been replaced by RESTful APIs.
4. AsyncAPI – Define message producers and consumers for asynchronous operations. Messages typically defined using the CloudEvent format.

Architectural Concerns

What are the other architectural concerns to know about?

1. **Security** – PayPal has two main security mechanisms as the application level; **PayPal Security Context** which is established on the calls to StarGate and propagated through the system and **Application Context** which is established when one application calls another. **PayPal Security Context** is used to authorize the call as a whole based on the external system making the call and uses OAuth2. **Application Context** is used to authorize the immediate caller of the application.
2. **Scalability and Reliability** – PayPal uses multiple redundant data centers both scale the system (horizontal scaling) and to improve reliability by failing over to another data center if one becomes unavailable.
3. **Observability** – PayPal uses Datadog and Splunk to provide observability into the system. Datadog is used for real-time monitoring and alerting while Splunk is used for log viewing and analysis.
4. **Usability** – Standards for user interface, both visual and API.
5. **Maintainability** – Standards for code quality. SonarQube for static quality analysis and unit test code coverage analysis.



Conclusion
What did we learn?

1. How clients call into PayPal
2. What the different kinds of applications are
3. How do we make sense of our applications
4. How do applications integrate
5. What other architecture concerns are there?

Reference

Where to learn more

1. PayPal Domain and Capability Model – The PayPal domain model
2. Architecture Building Codes – PayPal documentation on various architectural concepts as well as API authoring
3. Domain-Driven Design – Book on how domain modeling works

Thank you