

Microkernel Architecture

Software Architecture

Richard Thomas

March 9, 2026

So far...

Simplicity – Monolith, Pipeline

Modularlity – Layered, Pipeline

Definition 0. Extensibility

Features or extensions can be easily added to the software over its lifespan.

Question

How easy is it to extend *Monolith*, *Layered* or
Pipeline?

Question

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Answer

Monolith – Everything in one container



Question

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Answer

Monolith – Everything in one container



Layered – Typically all layers



Question

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Answer

Monolith – Everything in one container



Layered – Typically all layers



Pipeline – Create a new filter



Definition 0. Interoperability

Software can easily share information and exchange data with internal components and other systems.

Question

What about interoperability?

Question

What about interoperability?

Answer

Monolith – Everything in one container

- Internal  External 

Question

What about interoperability?

Answer

Monolith – Everything in one container

- Internal  External 

Layered – Nearest Neighbour

- Internal  External 

Question

What about interoperability?

Answer

Monolith – Everything in one container

- Internal External

Layered – Nearest Neighbour

- Internal External

Pipeline – Standard Interface

- Internal External

Question

What if I want simplicity, extensibility and interoperability?

Question

What if I want simplicity, extensibility and interoperability?

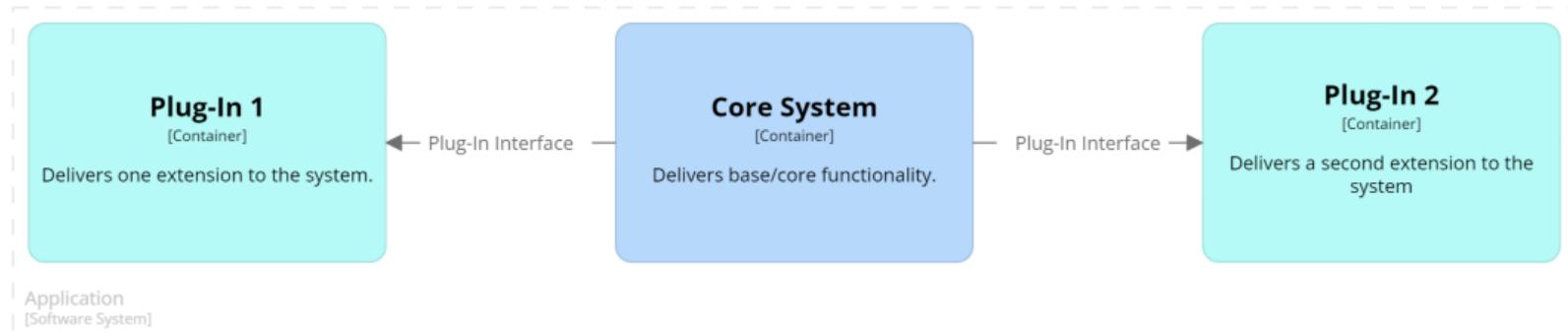
Answer

Consider *Microkernel Architecture*

§ Microkernel Architecture

Definition 0. Microkernel Architecture

Core system providing interfaces that allow plug-ins to extend its functionality.



Definition 0. Registry

Tracks which plug-ins are available to the core system and how to access them.

Loading Plug-ins

Static Loading when application starts

Dynamic Loading as needed at run-time

Registry designed for the selected strategy

Question

Can you think of a *microkernel architecture*?

Question

Can you think of a *microkernel architecture*?

Answer

Web Browser?

Definition 0. Independent Plug-in Principle

Plug-ins should be independent, with no dependencies on other plug-ins. The only dependency on the core system is through the plug-in interface.

Definition 0. Standard Interface Principle

There should be a single interface that defines how the core system uses plug-ins.

Question

Does a plug-in architecture equate to a microkernel architecture?

Question

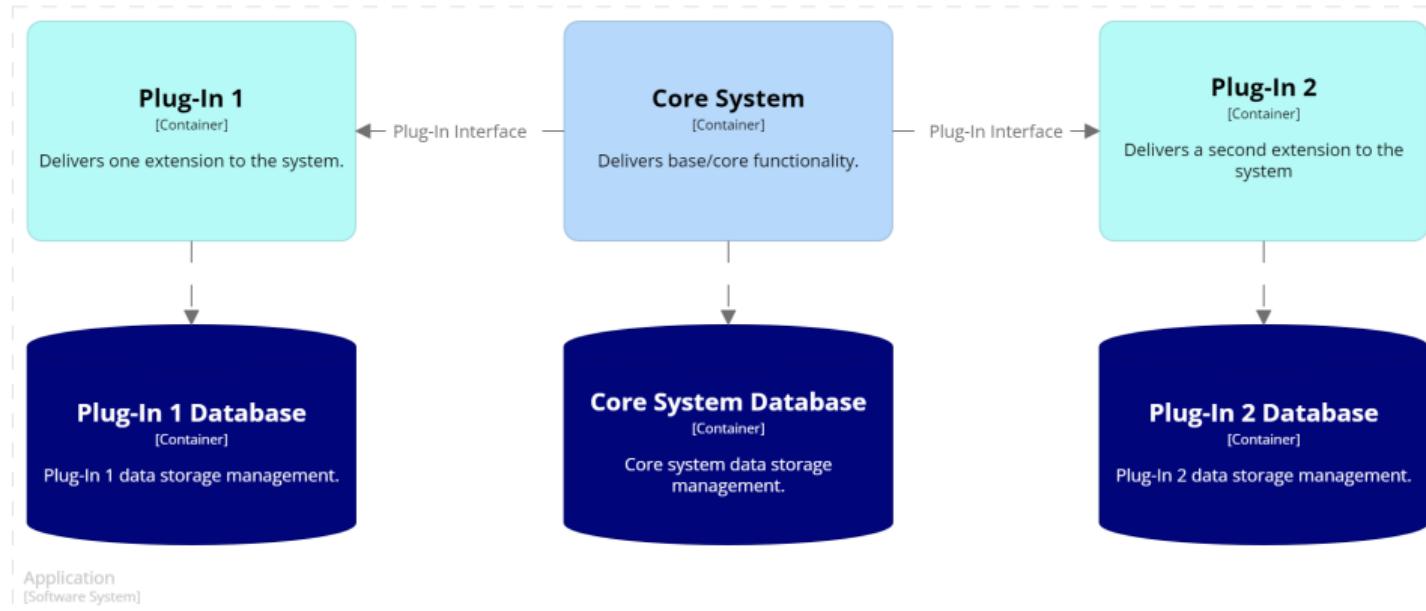
Does a plug-in architecture equate to a microkernel architecture?

Answer

What about *IntelliJ*?

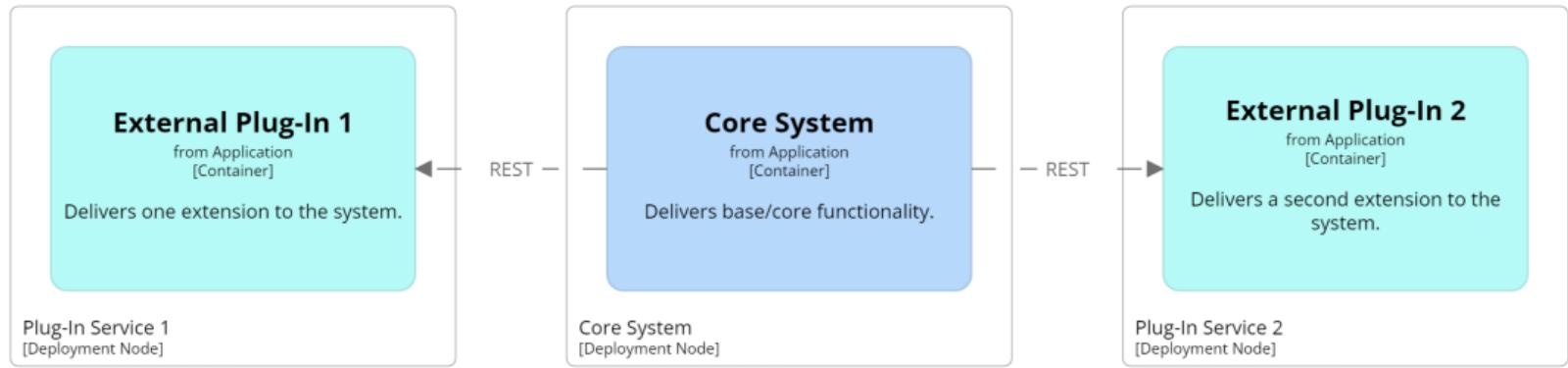
Plug-ins with Separate Databases

- Plug-ins cannot access core system data
 - Core system may pass data to the plug-in
- Plug-ins may have their own persistent data

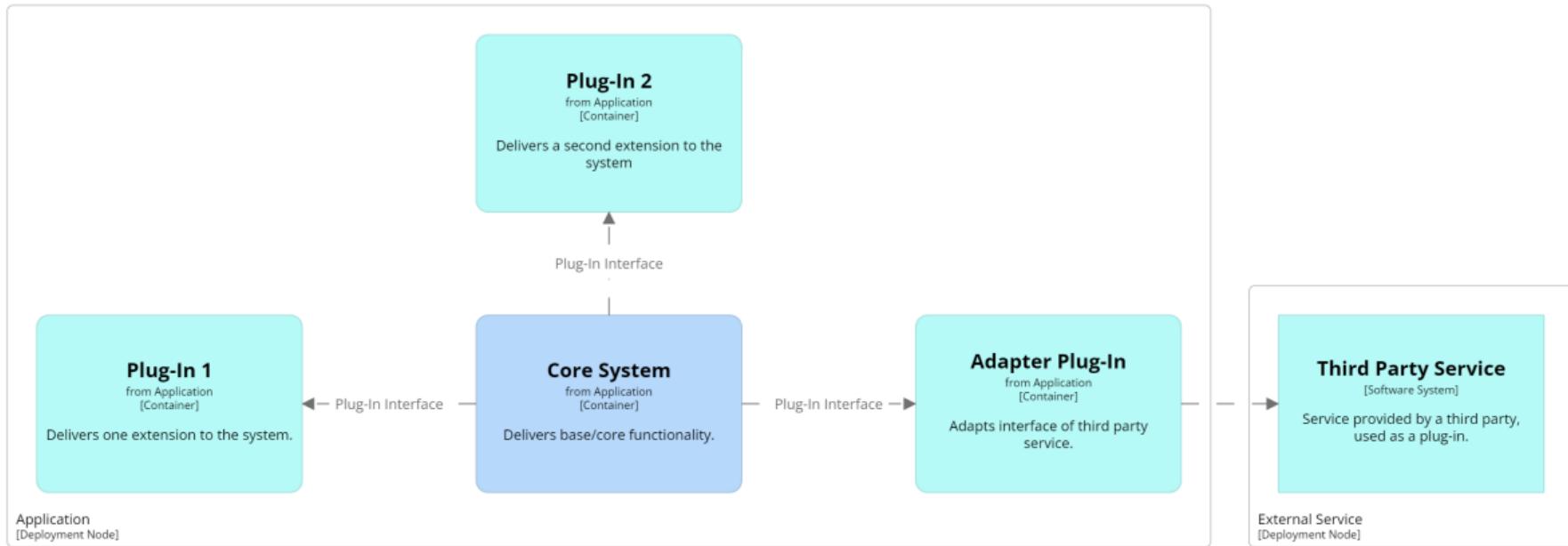


Plug-ins as External Services

- Need communication protocol
- Registry records communication contract
 - e.g. URL of the REST endpoint & data passed to it

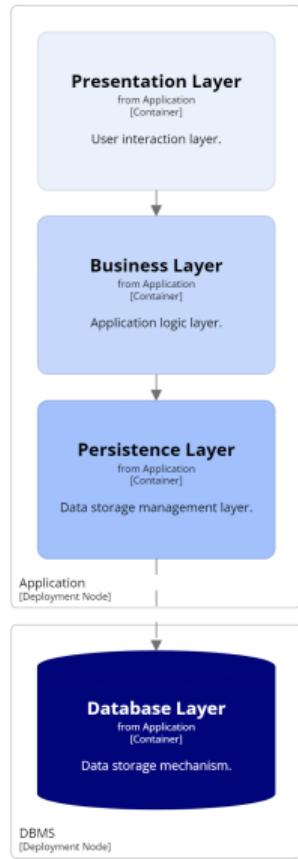


Adapting Non-Conforming Interfaces

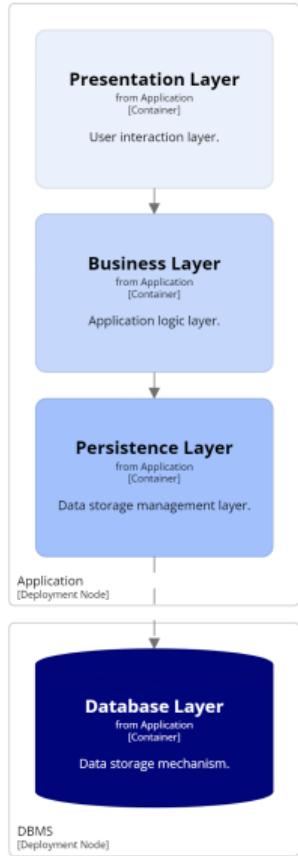


§ Technical & Domain Partitioning

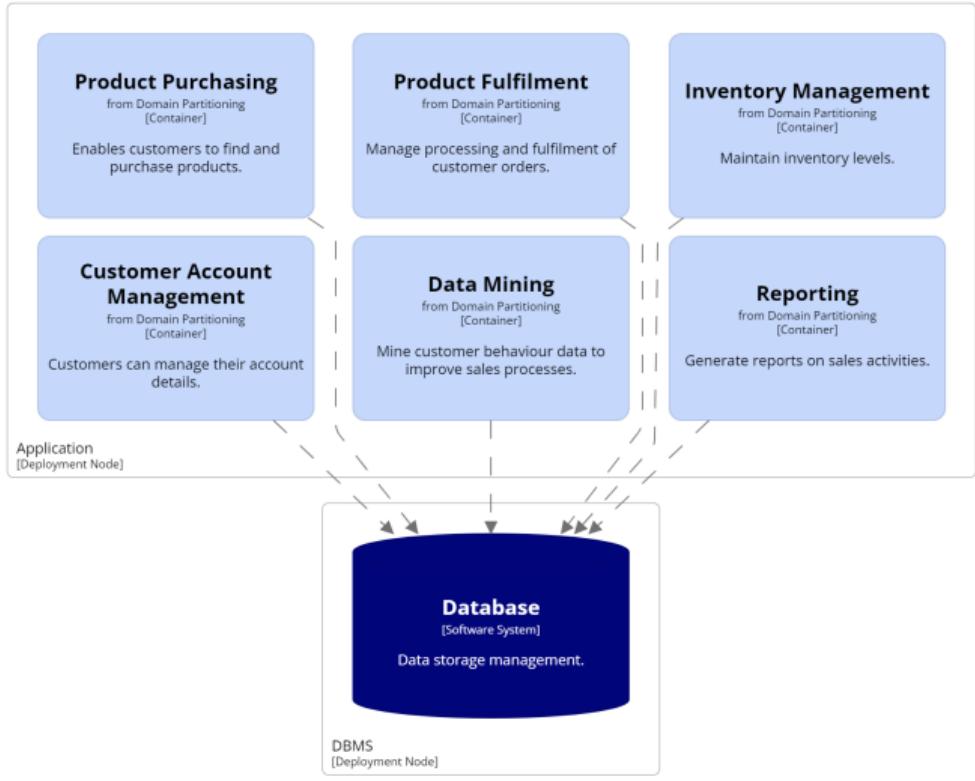
Technical Partitioning



Technical Partitioning



Domain Partitioning



Question

Is the microkernel architecture suited to
technical or *domain* partitioning?

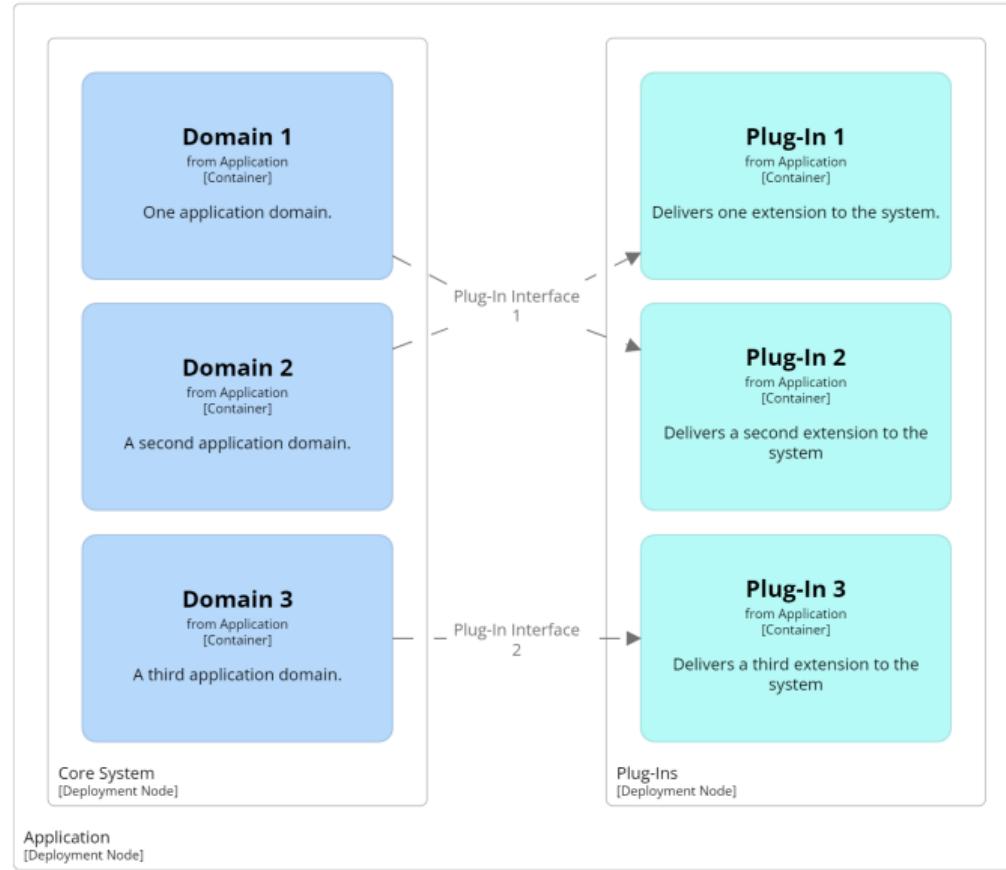
Question

Is the microkernel architecture suited to
technical or *domain* partitioning?

Answer

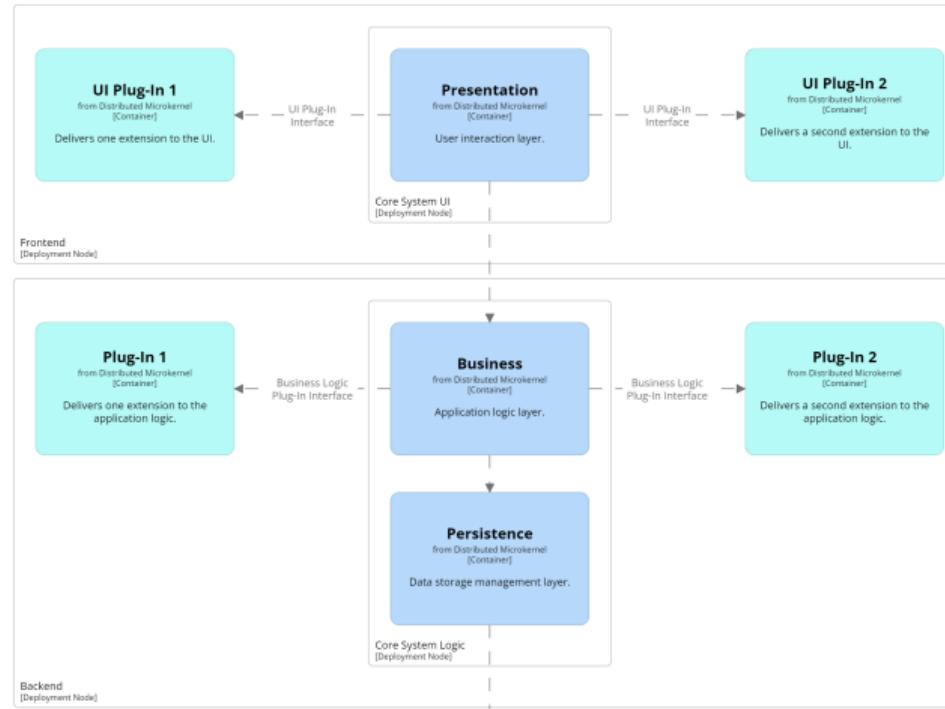
Core system can be partitioned either way.

Domain Standard Interfaces



Distributed Microkernel

- Partitions in the core system can be distributed
 - Technical or domain partitions
 - Plug-ins could also be distributed



§ Media Server Example

Question

What types of systems could use a *microkernel architecture*?

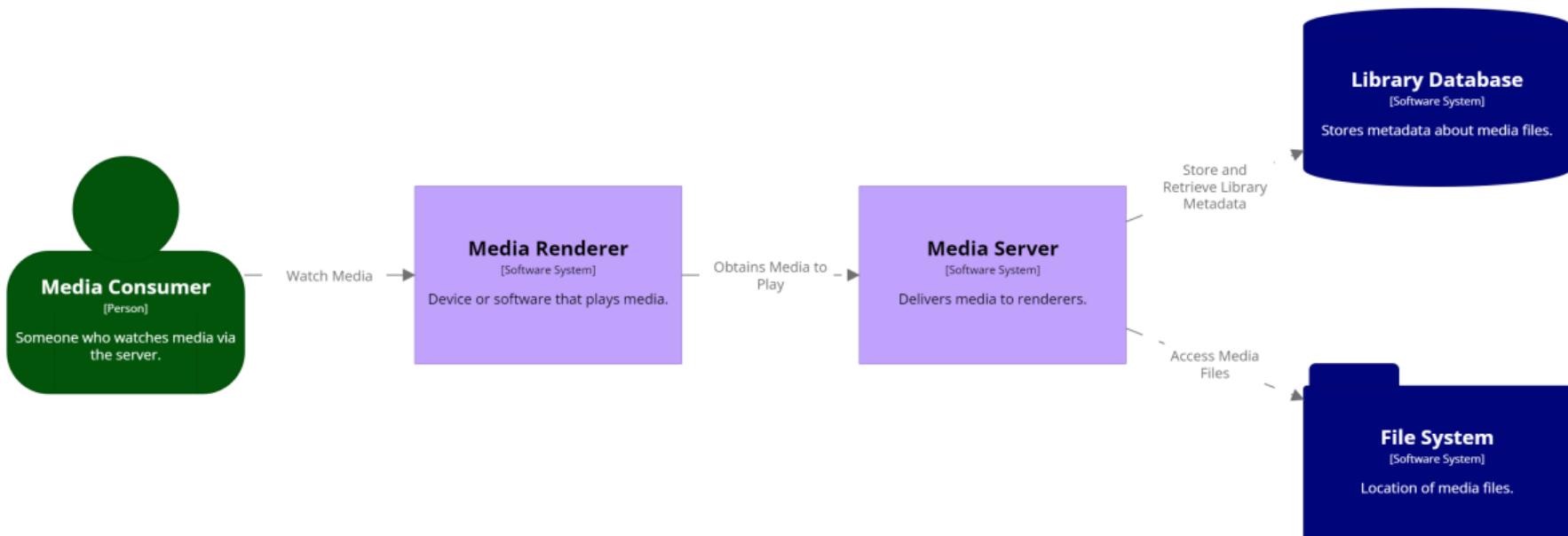
Question

What types of systems could use a *microkernel architecture*?

Answer

- Social media aggregator
- IoT management & processing
- Media server

Media Server & Renderer



Domain Colour Key

Container, audio

Container, core

Container, libmgt

Container, nav

Container, ui

Container, video

Element

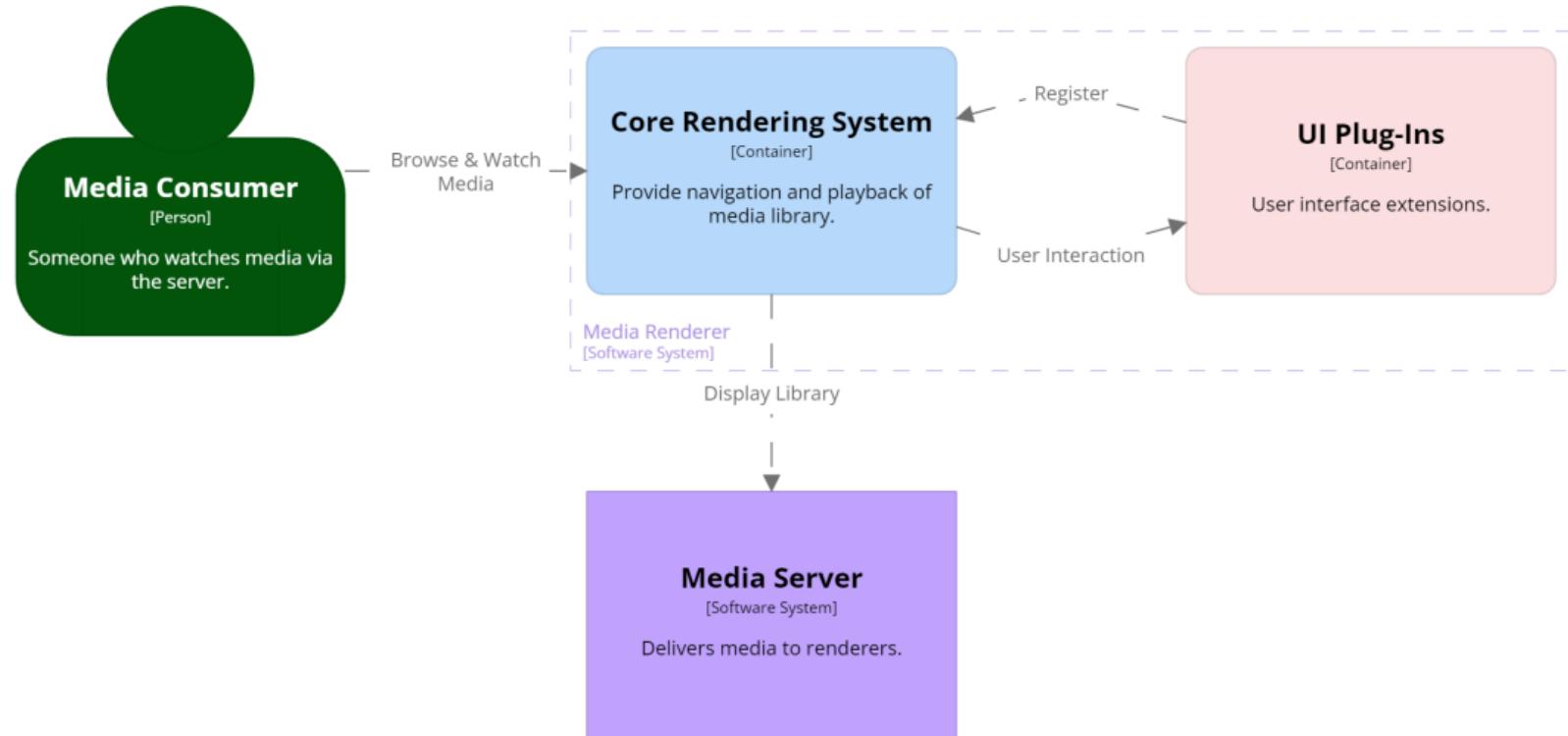
Software System, db

Software System, file

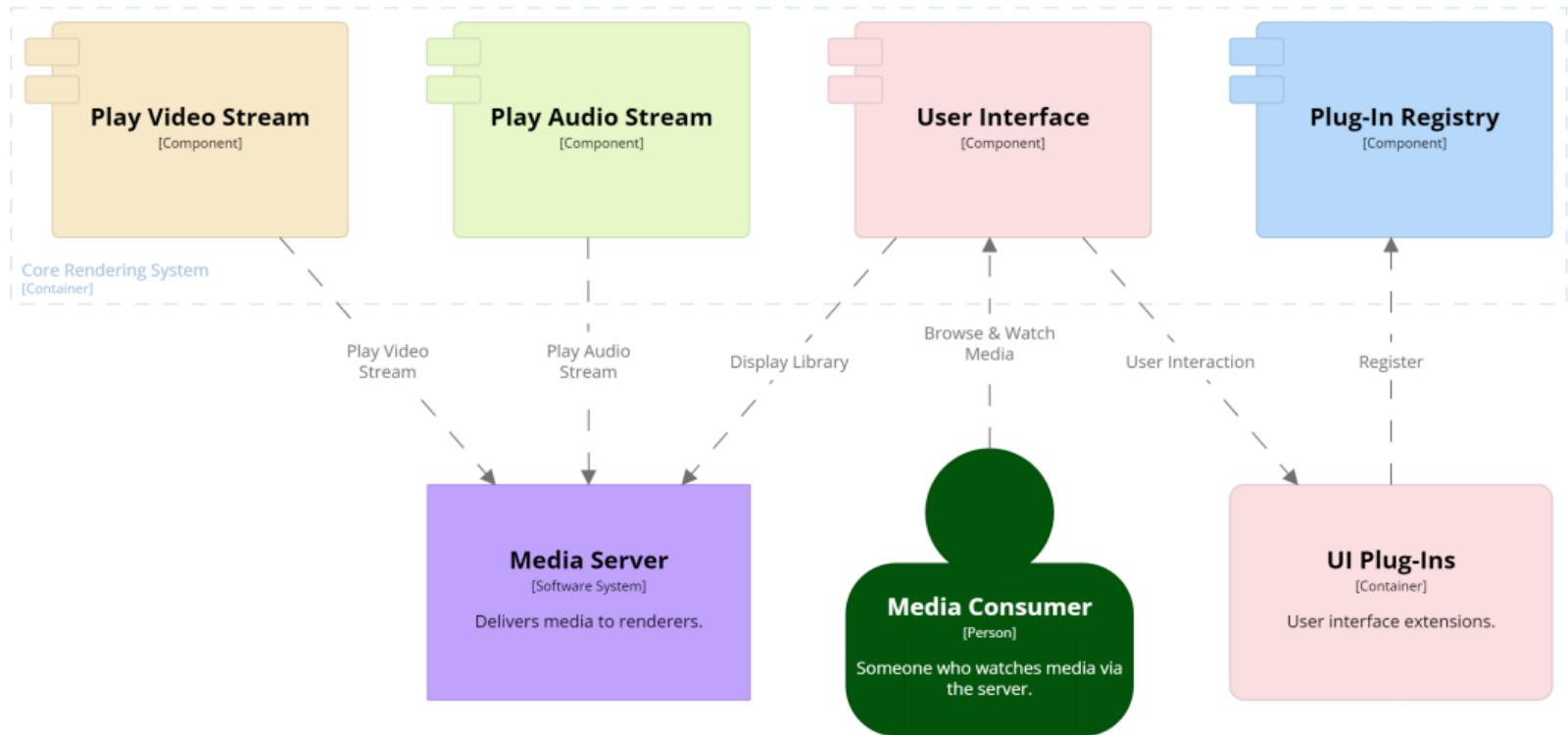
Relationship



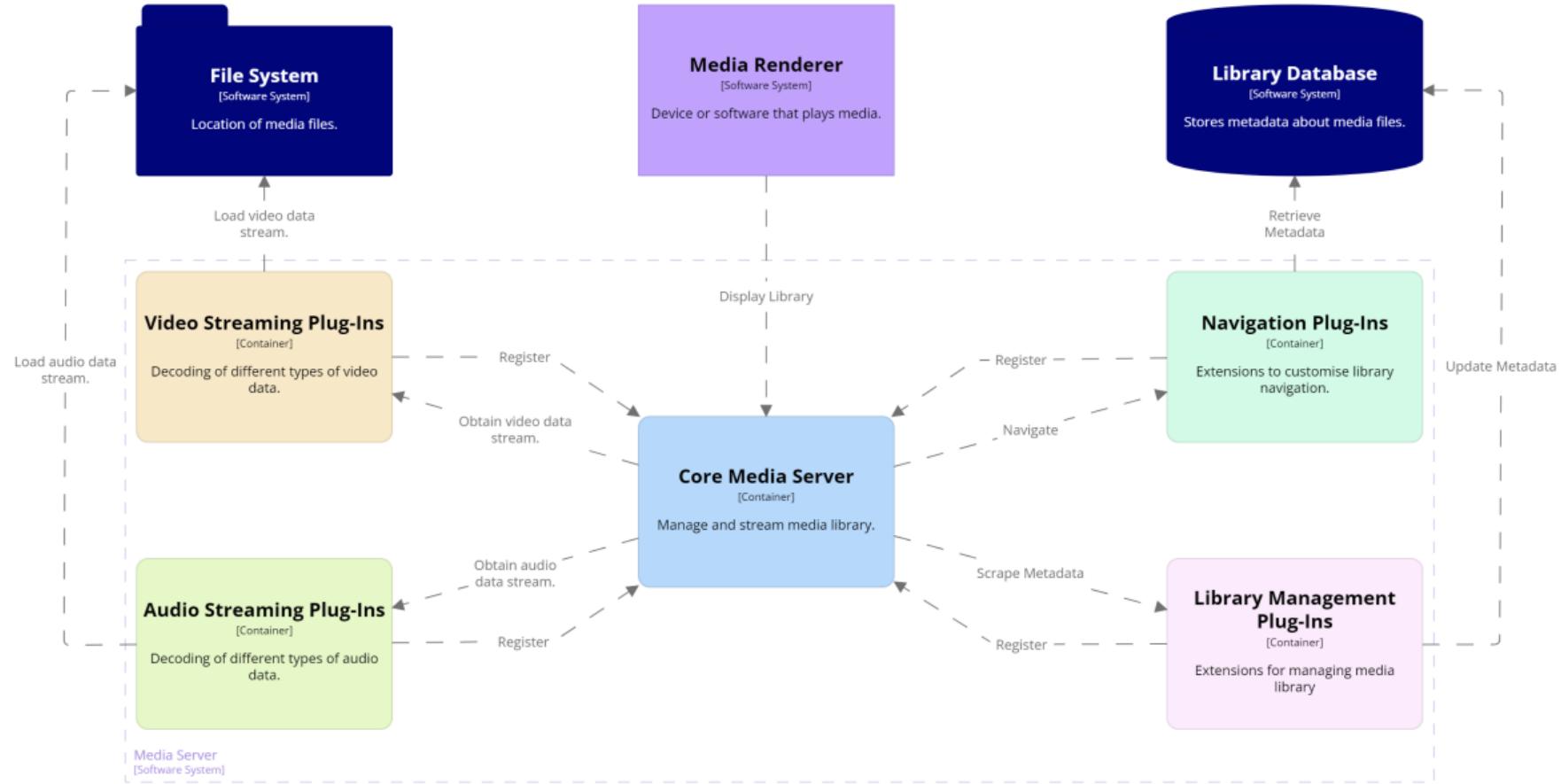
Media Renderer



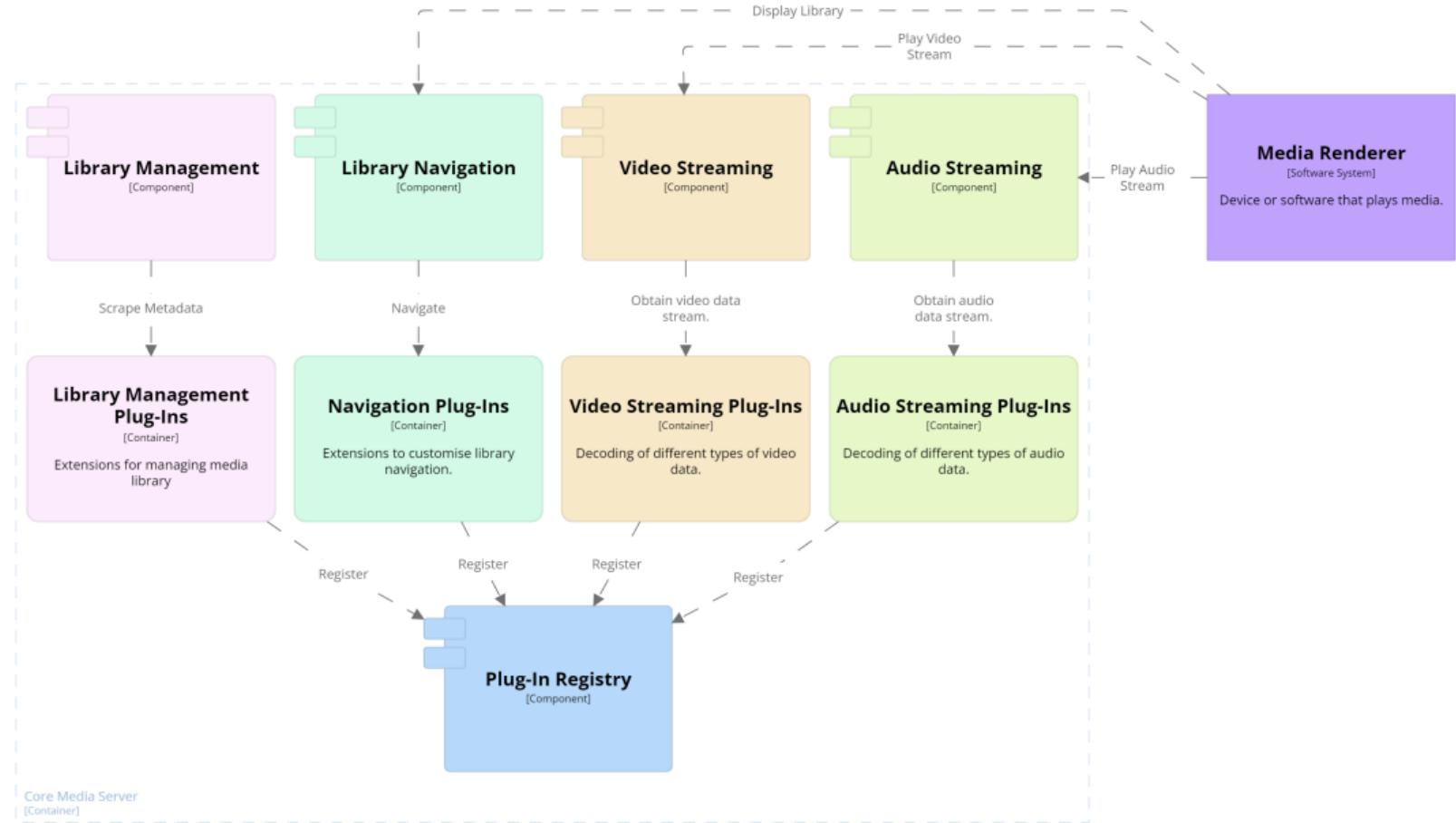
Core Rendering System Components



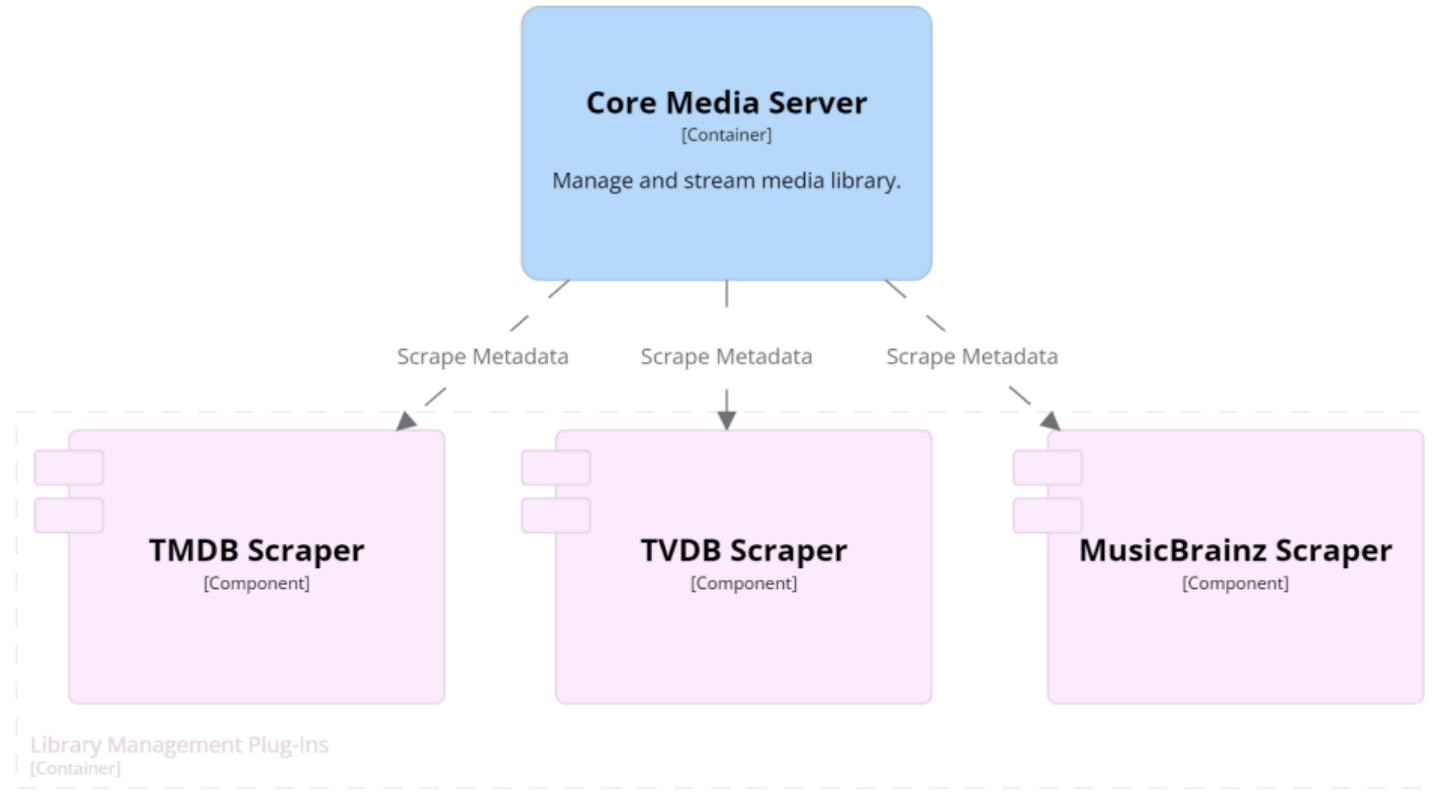
Media Server



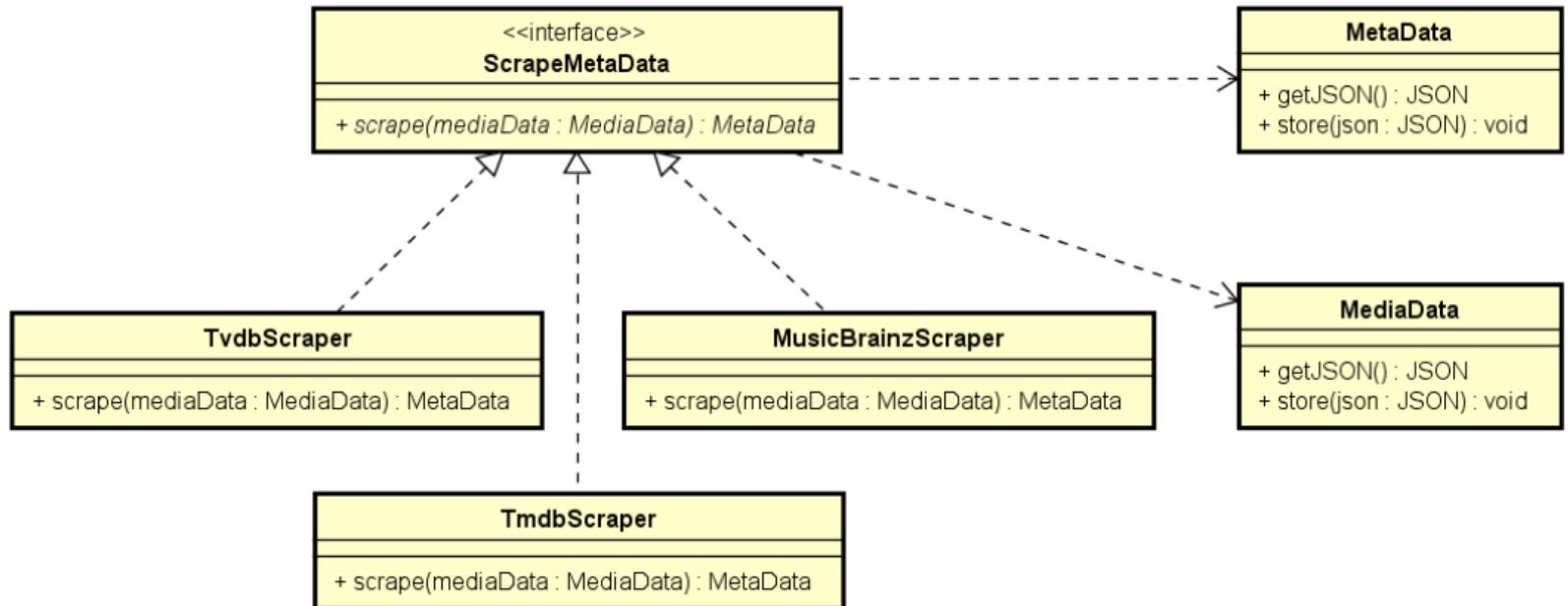
Core Media Server Components



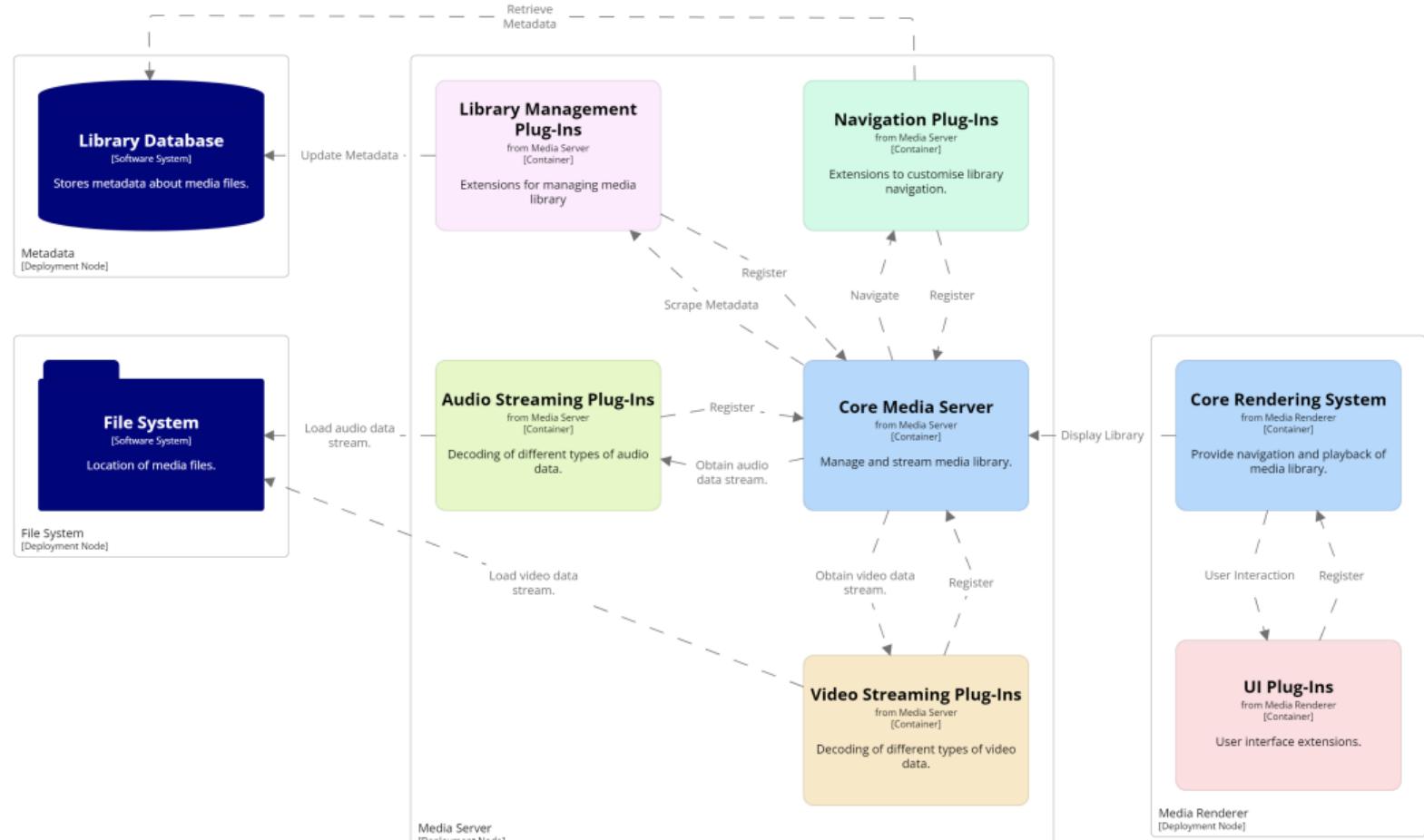
Media Library Management Components



Scrape Metadata Interface



System Deployment



§ Conclusion

Microkernel Pros & Cons

Simplicity Core system & Plug-in interface 

Extensibility Plug-ins 

Interoperability Plug-ins 

Scalability 

Reliability 