Pacs Software Build Language and Explanation:

Software	Programming Language(s)	Architecture	Explanation
Dicoogle	Java	Modular, Plugin-based	The core system is built in Java, leveraging its cross-platform capabilities. It uses a plugin-based architecture, allowing developers to extend functionalities easily. Dicoogle replaces traditional databases with an indexing and retrieval mechanism, making it efficient for PACS solutions.
SPHERE	Python, JavaScript	Microservices, Web-based	The backend is primarily Python, which facilitates DICOM image processing and PACS functionality. JavaScript is used for the frontend to create an interactive web-based interface. The system follows a microservices architecture, making it scalable and adaptable for health research applications.
Orthanc	C++, Python, JavaScript	Client-Server, RESTful API	C++ is used for the core DICOM server to ensure high performance and efficiency. Python allows scriptability and automation for research purposes. JavaScript is utilized for web-based interfaces. Orthanc provides a RESTful API, enabling integration with various programming environments.
EasyPACS	Java (Spring Boot)	Monolithic, Web-based	Built entirely in Java, using Spring Boot for backend services and Thymeleaf for frontend rendering. It follows a monolithic architecture, simplifying deployment as a single JAR file. This structure reduces setup complexity while maintaining efficiency.
OsiriX	Objective-C, C++	macOS-based, Client-Server	Objective-C is used for GUI development, making OsiriX a native macOS application. C++ is employed for high-performance image processing. It follows a client-server model, supporting both local and remote DICOM image visualization.