applied sciences

MDPI

Article

Performance of API Design for Interoperability of Medical Information Systems

Leticia Dávila Nicanor [1,*], Abraham Banda Madrid [1], Jesús E. Martínez Hernández [2] and Irene Aguilar Juárez [2]

1 Laboratorio de Evaluación y Calidad de Software, Centro Universitario UAEMex Valle de México, Boulevard Universitario s/n. Predio de San Javier, km 11.5, Atizapán de Zaragoza 54500, Mexico; abandam@uaemex.mx

2 Centro Universitario Texcoco, Universidad Autónoma del Estado de México UAEMex, Av. Jardín Zumpango s/n. Fraccionamiento El Tejocote, Texcoco 56259, Mexico; jmartinezh016@alumno.uaemex.mx (J.E.M.H.); iaguilarj@uaemex.mx (I.A.J.)

* Correspondence: ldavilan@uaemex.mx

Abstract: After the experience of the COVID-19 pandemic, it has become evident that efficient and secure interoperability of medical information is crucial for effective diagnoses and medical treatments. However, a significant challenge arises concerning the heterogeneity of the systems storing patient information in medical centers or hospitals. Memory management becomes a pivotal element for the effective operation of the proposed API, as it must seamlessly execute across various devices, ranging from healthcare units, such as mobile phones, to servers in cloud computing. This proposal addresses these issues through techniques designed to enhance the performance of the software architecture in creating a medical interoperability API. This API has the capacity to be cloned and distributed to facilitate the exchange of data related to a patient's medical history. To tackle heterogeneity, efficient memory management was implemented by utilizing an object-oriented approach and leveraging design patterns like abstract factory and wrapper. Regarding the evaluation of the proposal, this study showed an estimated performance of 94.5 percent, which was indirectly demonstrated through the assessment of operation sequences. This result suggests a satisfactory level based on complexity and coupling.

Keywords: API; application programming interface; information medical interoperability; performance design; design patterns; HL7

check for updates

Citation: Nicanor, L.D.; Madrid, A.B.; Hernández, J.E.M.; Juárez, I.A. Performance of API Design for Interoperability of Medical Information Systems. *Appl. Sci.* **2024**, *14*, 3944. https://doi.org/10.3390/app14093944

Academic Editor: Hariton-Nicolae Costin

Received: 1 March 2024
Revised: 23 April 2024
Accepted: 28 April 2024
Published: 6 May 2024

1. Introduction

The transition from paper-based medical records to electronic health records (EHRs) holds significant potential benefits. One such advantage is the redundancy mitigation when sharing patient information across medical facilities. Additionally, the adoption of EHRs contributes to an enhancement in service quality. Through the sharing of patients' medical histories, doctors and specialists can make more informed and accurate decisions, ultimately improving the speed of their responses [1–3].

However, a substantial challenge arises when linking information systems due to the existing heterogeneity between data and processes related to health records (EHRs). Various standards have been proposed for the exchange of medical information, including openEHR [4], NANDA-I [5], ISO/IEEE 11073-PHD [6,7], and X73PHD [8] based on ISO/IEEE 11073, that are widely accepted. Notably, the HL7 standard has garnered attention; nowadays, there has been a global adoption of the HL7 protocol in the development of information systems applied to the clinical area [9–16]. The HL7 organization, which was established in 1987 for electronic data exchange in the health field (HL7, 2024), had these standards approved by the American National Standards Institute (ANSI) in 2003 [17]. Different versions of HL7, such as HL7 v2.x and HL7 v3, exhibit differences in structure and approach, although HL7 v2.x is more commonly used in current clinical practice. The latest version, HL7 Fast Healthcare Interoperability Resources (FHIR) [11], is built on versions

2.x and 3.x. FHIR boasts a simplified implementation, offering the advantage of using the HTTP protocol and RESTful principles for the development of efficient services. It utilizes HTML and cascading style sheets to generate graphical interfaces, providing the flexibility to integrate the user interface into a broader environment. Additionally, FHIR allows for the selection between JSON, XML, or RDF to represent detailed information in the resulting data.

Despite its strengths, FHIR has faced technical criticism, particularly in the development of API interoperability applications, where its implementation can become complex. In another critique, RIM version 3 is characterized as a standard with challenges in implementation, usability, scope, documentation, and marketing [18].

The proposed approach sought to address the following research questions, which are considered primary contributions:

1. Is an object-oriented-design-based API following the HL7 CDA standard a viable solution for addressing the heterogeneity of legacy medical systems?
2. How can design patterns enhance the architectural design performance of the API to provide interoperability on medical information systems?
3. Can valid measures of complexity and coupling be employed to assess the efficiency of the API's architectural design?

2. Related Work

After the experience of the COVID-19 pandemic, it was found that the efficient and secure interoperability of medical information is of great importance to provide a good medical diagnosis and treatment of patients [1]. There is a need to exchange clinical records to improve the knowledge base and efficiency of care in the health sector. Electronic health records (EHRs) are a collection of information about a person's health status [3]. This standard focuses on the distinction between the representation of health information and its technical implementation. This separation enables the development of more flexible and adaptable electronic health systems over time, which, in turn, significantly contributes to the continuous improvement of patient healthy [2,19]. This has a great impact on society's health since it is used to carry out assertive diagnoses in medical treatments. The information interoperability in the health sector from an international approach is intended to be resolved through interoperability standards and robust interoperability platforms [1,2,20]. The big problem when linking information systems are the coexisting heterogeneity between data and processes related to health records. Many countries have complex health systems; the use of standards and their automation must support the complexity that represents the interoperability of information from clinical records in the various systems that constitute the complex health systems of these regions.

2.1. Medical Information Interoperability

Shin [15] described the development of a mobile application for the transmission of biosignals in the health field through the HL7 protocol. The proposed solution is optimized for the home healthcare (uHealthcare) environment and uses ubiquitous computing technologies to monitor health at home. The main components of a ubiquitous medical system are identified, including detection, transmission, diagnosis, and prescribing. This paper describes a specific approach to developing an HL7 API using LabVIEW, emphasizing its role in the transformation of medical data and its capability to transmit this data to other systems via a radio frequency. In this application, data are transmitted from the SpO2 sensor's biosignal to a mobile phone using a serial approach. The mobile phone converts raw data to the HL7 standard and sends it to other PCs via Bluetooth or a WLAN. In the system development, the structure of the abstract message was defined as part of the message-modeling material. The research area of health monitoring systems has transitioned from the simple reasoning of wearable sensor readings to advanced data processing, with the aim to provide more valuable information to end users. In Banaee H.'s study [21], emphasis was placed on the significant role of data mining techniques in extracting valuable insights from data collected by wearable sensors in health monitoring systems. These techniques include anomaly detection, the prediction of future trends based on historical sensor data, making informed decisions about individual health status, and processing continuous time series measurements. However, these advancements also pose challenges and concerns, such as ensuring the reliability and accuracy of collected data, data processing efficiency, and validation in practical healthcare settings.

In Viangteeravat's work [16], a prototype for the HL7 mapping function in version v3-RIM was introduced to integrate distributed clinical data sources using R-MIM classes. This prototype was integrated as a module within the Clinical Data Management System (CDMS) clinical databases. During testing with the legacy CDMS, the authors argued for a significant improvement in the information delivery. The unique identifications of the message, interaction identifier, injection event, and receiver responsibilities is achieved through the root element. In the system architecture, the wrapper or decorator design pattern was employed to encapsulate information defined by the interaction code (IN). The IN displayed in the actual message consists of the injector event, message type, transmission container, decorator template of the control act, sender, and receiver. The initializing event and the control act decorator represent another layer of encapsulation around the actual message, providing information on the date and time the event occurred and the individuals responsible. A notable advantage is that HL7 v3 uses the XML data model, unifying data and metadata in a consistent format and operating as a real-time protocol. However, it is crucial to note that the transaction time in clinical database transactions continues to be a critical aspect in this context.

In Nicholson's work [20], the challenge of interoperability in population-based patient registries (PBPRs) was addressed, proposing the use of a federated semantic MDR framework. Given the complexity of healthcare infrastructures, health data systems, and the need to interact with legacy systems, it was emphasized that resolving all interoperability issues through a single health data standard is unlikely. According to the authors, this proposed framework can operate across standards, facilitate data linkage between disparate systems, and encourage data federation, eliminating the need for centralized data collection processes. While the framework's potential is highlighted, some obstacles are mentioned. Achieving interoperability between PBPR domains requires agreement on common data elements (CDEs) at the inter-domain level. Additionally, each metadata and data provider must establish a local semantic master data repository (MDR) and configure RESTful services, which may pose a challenge for smaller or underfunded registries. Another proposed framework is AIoTES [1], which offers semantic interoperability at the platform level, enabling different platforms, technologies, and standards to coexist and communicate effectively. This facilitates the development of applications and services that operate across multiple platforms, promoting the exchange and adoption of cross-platform services and applications for active and healthy aging (AHA). The AIoTES API serves as a single entry point to access the framework's functionality. It provides unified RESTful operations that allow access to internal components, such as the semantic interoperability layer, data lake, and data analytics. Additionally, it offers consolidated access to the user interfaces of the tools, ensuring security in transactions.

Cheng's proposal [22] outlines an architecture focused on the development of a RESTful API service. This API, upon access, verifies personal login rights in electronic health records (EHRs), REDCap login rights, REDCap project permissions, institutional review board (IRB) project approval status, and key study personnel status each time an end user logs into a REDCap project. The implemented standard is HL7 FHIR [11], which has been used to develop a new REDCap module that facilitates data extraction for case report form (CRF) studies, registries, and data repositories. This proposed module has been utilized in various high-profile studies at Vanderbilt University Medical Center and has been shared with institutions utilizing REDCap.

2.2. Application Programming Interface Challenges

An application programming interface (API) is a set of rules and tools that allows different software applications to communicate with each other [23]. APIs define the methods and data that developers can use to integrate different software components. From the viewpoint of the involved systems, the API is seen as a black box since it hides its internal structure. However, it provides access to a set of procedures and functions designed to establish interoperability between devices, platforms, and information systems. According to Biehl [24], a RESTful API is based on the architectural principles of representational state transfer (REST). These principles include representing resources through URLs using standard HTTP operations (GET, POST, PUT, DELETE) to interact with those resources, and transferring representations of resources, typically in the JSON or XML format.

In the work by Praschl [25], the "imaging framework" is presented as a solution to tackle the challenges of interoperability and extensibility in Java frameworks related to image processing tasks. The modular and extensible nature of Java, its platform independence, and broad support across various operating systems make it a versatile choice for image-related tasks. Additionally, Java's support for GPU-accelerated frameworks contributes to the efficient execution of complex image processing and computer vision algorithms. Despite Python's popularity in these fields, Java boasts a robust and extensive community. In the realm of API development with Java, Saleh [26] analyzed the features of various languages that interoperate with Java. The highlighted benefits include syntax similarities and agreements on object-oriented programming principles, facilitating interoperability. Saleh's main proposal focuses on solution development, particularly with the "wrapper" design pattern to execute Java code in Eolang. It was emphasized that the ability to support certain syntax is the fundamental mechanism for developing interoperability. Furthermore, expanding the atom set, including Java wrappers, is suggested as a promising perspective for Eolang in terms of interoperability.

Despite the existence of standards for data interoperability in the medical field, as mentioned earlier, persistent situations must be considered. Heterogeneity at the hardware and software levels implies that medical hubs can range from devices as small as a cell phone to cloud-integrated systems. Another challenge is the indeterminate number of transactions that may occur in each query, demanding efficient memory management. Therefore, APIs, which serve as the central core in interoperability frameworks, must be portable and capable of dynamically and efficiently handling an indefinite number of transactions. In this proposal, an API was designed under the semantics of the HL7-CDA substandard version 3. The proposed architecture adopted a dynamic memory approach using object-oriented design patterns to optimize the efficiency. The integration of wrapper and abstract factory design patterns for managing medical record information is crucial. This approach optimized the memory and time resources, enhancing the efficiency for object construction during execution in heterogeneous environments.

3. Design of the API Proposal

In the proposed API, the architecture was designed to exchange medical information from patients during their initial clinical contact, i.e., information recorded in the general medicine area. The API was based on HL7 standards for clinical documentation, and DICOM standards [27] for documentation from imaging studies were integrated into the API layers. As shown in the use case diagram in Figure 1, four main views were determined: healthcare professionals, healthcare institutions, patients, and governmental organizations.

(1) *Healthcare Professionals View:*

In this view, doctors, paramedics, nurses, and general healthcare staff in health centers or hospitals will have the ability to access medical information and update patient data. This includes medical orders, laboratory results, and electronic medical records. This view will have the primary involvement concerning the patient, where they are able to communicate with the imaging area and exchange patient results using unique identifiers for each study, activity, and message, following the DICOM imaging studies standard and its official CONF-DIR specifications. The physician will record the gathered information using a form provided through the API based on the CDA clinical document architecture of the HL7 standard. This form will consist of the header, which will contain the patient's general personal information. The body will be divided into sections, the number of which will depend on the studies needed and collected for the healthcare professional to reach an appropriate diagnosis. The document's body will also contain additional details that the physician deems necessary for the complete patient record. The physician is free to add as many sections as needed but must assign a unique identifier to each one.

(2) *Patient View:*

The patient will have a minority role in managing the API. They will not have a say in the manipulation of their medical information and can only access the system for inquiries and to download certificates added by the general practitioner, as well as to update personal data when necessary.

(3) *Healthcare Institutions View:*

This view will be responsible for admitting the personnel of the health center or hospital, allowing them to interact with the medical information system. This will facilitate the transfer of clinical information between different healthcare institutions, which is crucial for coordinated healthcare and the continuity of healthcare.

(4) *Governmental Organizations View:*

This contributes to collecting standardized data for public health and epidemiological research. Additionally, it will facilitate the implementation of health policies and healthcare quality monitoring at a national and international level.

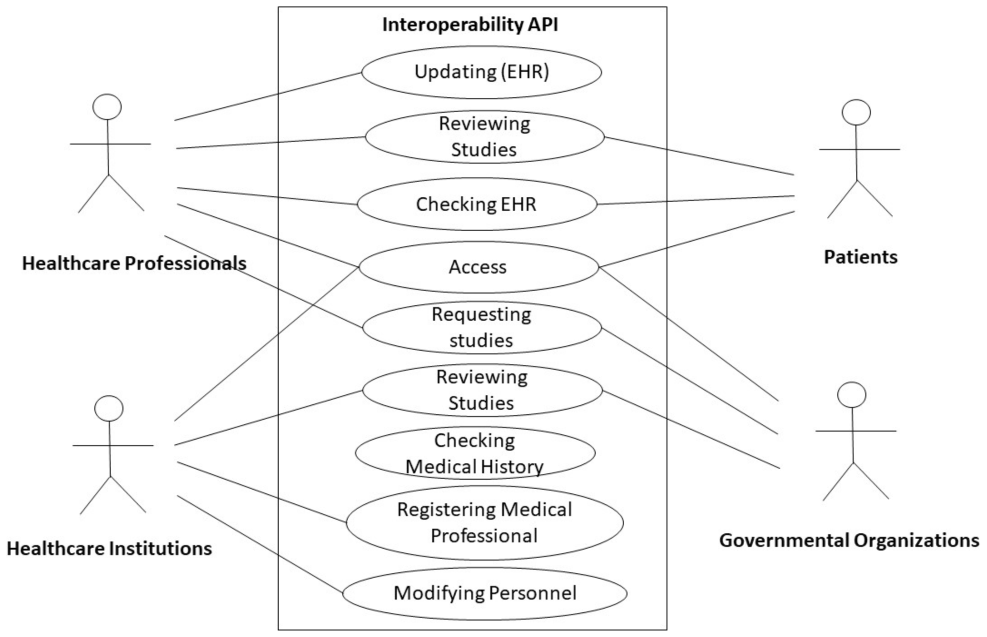

Figure 1. Interoperability API's use case.

Concerning functional requirements, Table 1 is presented, which describes the specifications following the views.

Table 1. API's functional requirements descriptions.

Id	Description
FR1	The system allows access to authorized users through authentication with their username and password
FR2	The system is required to consider profiles based on roles and organizational structure. The system includes four views: healthcare professionals, healthcare institutions, patients, and governmental organizations.
FR3	Healthcare professionals: healthcare staff in health centers or hospitals will have the ability to access medical information and update patient data, such as updating electronic health records (EHRs), reviewing studies, checking EHRs, and requesting studies.
FR4	Patient: this user will only be able to perform query operations and download certificates and studies, such as checking EHRs and reviewing studies.
FR5	Healthcare institutions: this user will be able to update information about the personnel of the health center or medical unit in charge, such as registering medical professionals, modifying personnel, and reviewing studies.

API's Architectural Design

A clonable and distributable architecture has been proposed for the medical interoperability API, whose instances establish connections with medical centers or hospitals to obtain patient medical history information. In Figure 2, the proposed architecture is shown and was designed using abstract factory and wrapper design patterns. When a patient arrives and their medical record is required, the API can clone itself through an abstract factory (*Factory_HCE*) that generates objects of type HCE at runtime according to the number and characteristics of medical centers or hospitals it needs to contact to obtain the patient's medical information. Once contact is established with the medical centers, the *Wrapper Medical History* class retrieves patient information by encapsulating details such as personal data controlled by the *Patient Identification Data* class, references to clinical document records, and details of events related to the ailment that triggered the search, as controlled by the *CDA* class, where medical records are concentrated according to the HL7 CDA protocol format in the contacting medical centers or hospitals. Methods were established to obtain information about a list of previous illnesses or conditions, records of performed medical procedures, reports from imaging studies (X-rays, MRIs, etc.), vaccination history, vaccines received with corresponding dates, clinical notes, records of medical visits and consultations, observations and comments from medical staff, prescriptions, treatments, prescribed medications, details of specific treatments, insurance information and billing, details about the patient's health insurance, billing and financial records, emergency contact information, and details of individuals to contact in case of emergency. Finally, the Laboratory Results and Studies class is part of the encapsulation of references to download information related to imaging studies.

According to the use case diagram, the healthcare professional view can request to initialize the API operation. Figure 3 depicts the state diagram of the HCE class corresponding to the medical history. In this figure, it is possible to observe three states in the API operation: data validation, data processed, and record updated. The method get_Patient_Identification_Data establishes the validation data to reach the validation state. The method get_List_Of_Previous_Illnesses_Or_Conditions locates the information related to the validated user in the medical system. Once the patient's data are located, the wrapper updates and encapsulates the records according to the HL7 format using the get_Record_Of_Performed_Medical_Procedures methods, getter, and setter. For studies containing images, only the links' addresses from the imaging area are taken into account. Once this process is complete, the complete record is sent as text. In this way, each instance is generated through the abstract factory according to the medical centers or hospitals where the patient has information.

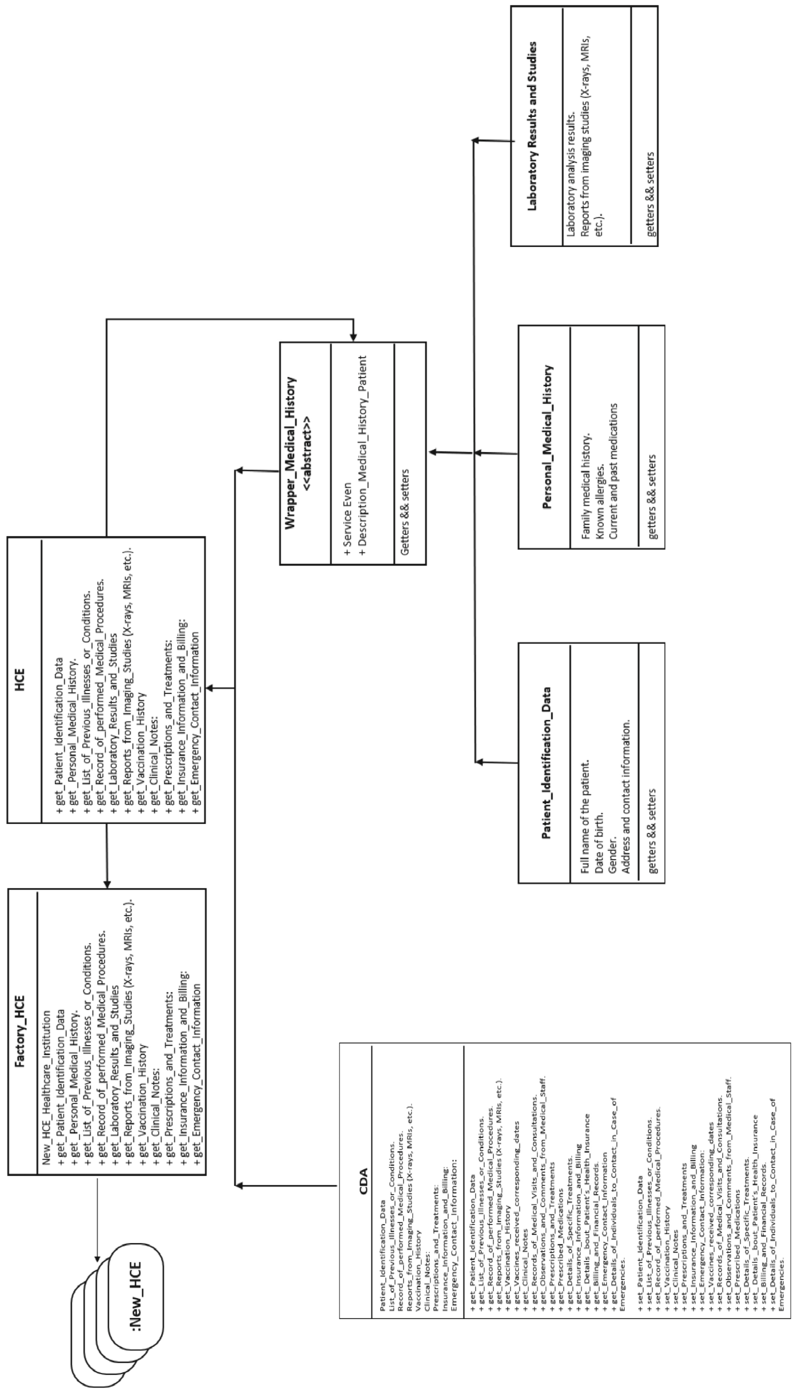
Laboratory Results and Studies
Laboratory analysis results.
Reports from imaging studies (X-rays, MRIs, etc.).
getters && setters
Wrapper_Medical_History
<<abstract>>
+ Service Even
+ Description_Medical_History_Patient
Getters && setters
Personal_Medical_History
Family medical history.
Known allergies.
Current and past medications
getters && setters
Patient_Identification_Data
Full name of the patient.
Date of birth.
Gender.
Address and contact information.
getters && setters
HCE
+ get_Patient_Identification_Data
+ get_Personal_Medical_History.
+ get_List_of_Previous_Illnesses_or_Conditions.
+ get_Record_of_performed_Medical_Procedures.
+ get_Laboratory_Results_and_Studies
+ get_Reports_from_Imaging_Studies (X-rays, MRIs, etc.).
+ get_Vaccination_History
+ get_Clinical_Notes:
+ get_Prescriptions_and_Treatments:
+ get_Insurance_Information_and_Billing:
+ get_Emergency_Contact_Information
Factory_HCE
New_HCE_Healthcare_Institution
+ get_Patient_Identification_Data
+ get_Personal_Medical_History.
+ get_List_of_Previous_Illnesses_or_Conditions.
+ get_Record_of_performed_Medical_Procedures.
+ get_Laboratory_Results_and_Studies
+ get_Reports_from_Imaging_Studies (X-rays, MRIs, etc.).
+ get_Vaccination_History
+ get_Clinical_Notes:
+ get_Prescriptions_and_Treatments:
+ get_Insurance_Information_and_Billing:
+ get_Emergency_Contact_Information
:New_HCE
CDA

Figure 2. Interoperability APIs architectural design.

As shown in the interoperability operation in Figure 4, the information obtained through the instances at each medical center returns to the requesting medical center and consolidates the information for the requisition.

When the *Healthcare Professionals View* requests information for a patient with the following medical history: mild hypoventilation with wheezing, heart rate of 95, respiratory rate of 28, inguinal lymph node with abnormal behavior, and periscapular inguinal lymph node with normal signs, an XML document is generated with the format as shown in Table 2.

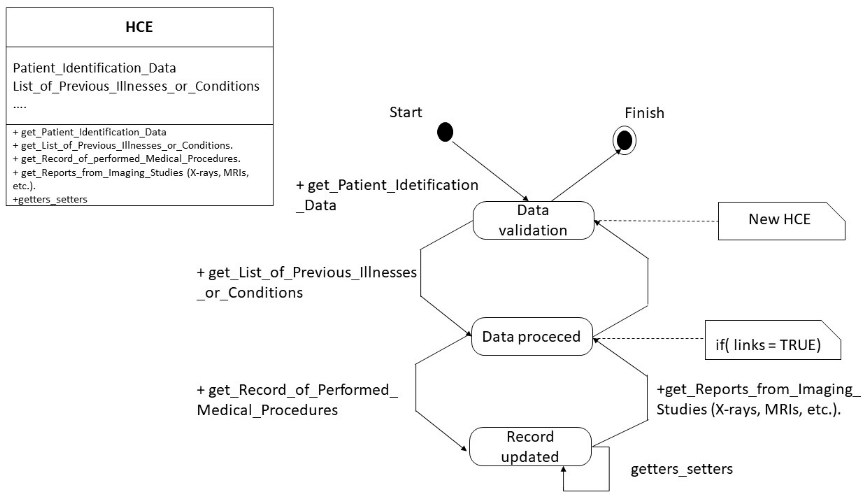
HCE
Patient_Identification_Data
List_of_Previous_Illnesses_or_Conditions
....
+ get_Patient_Identification_Data
+ get_List_of_Previous_Illnesses_or_Conditions.
+ get_Record_of_performed_Medical_Procedures.
+ get_Reports_from_Imaging_Studies (X-rays, MRIs, etc.).
+getters_setters
Start
Finish
+ get_Patient_Idetification _Data
Data validation
New HCE
+ get_List_of_Previous_Illnesses _or_Conditions
Data proceced
if(links = TRUE)
+ get_Record_of_Performed_ Medical_Procedures
+get_Reports_from_Imaging_ Studies (X-rays, MRIs, etc.).
Record updated
getters_setters

Figure 3. HCE state diagram design.

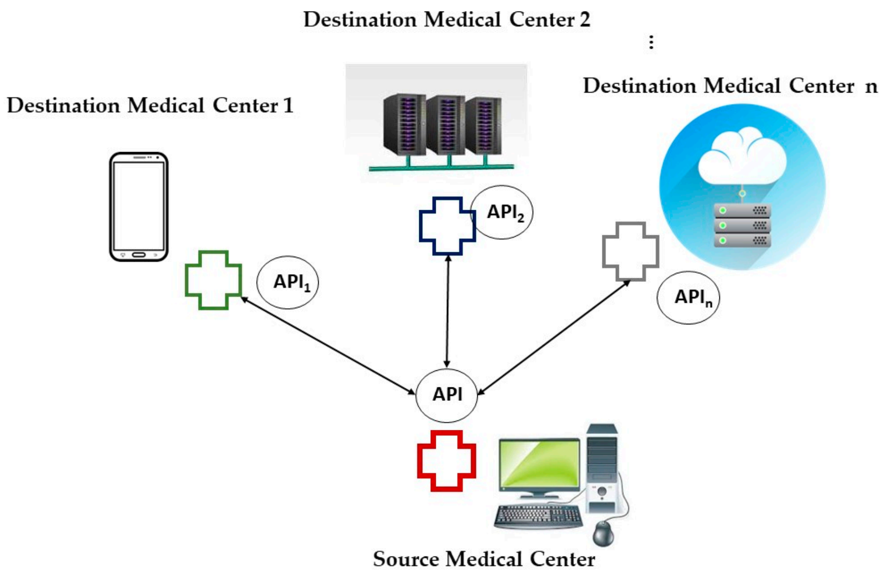
Destination Medical Center 2
Destination Medical Center 1
Destination Medical Center n
API$_2$
API$_1$
API$_n$
API
Source Medical Center

Figure 4. Interoperability APIs operation.

Table 2. XML response of the patient's objective description.

Output XML Document
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalDocument xmlns="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<!-- Document Header -->
<typeId root="2.16.840.1.113883.1.3" extension="POCD_HD000040"/>
<templateId root="2.16.840.1.113883.10.20.22.1.1"/>
<!-- Patient Information -->
<recordTarget>
<patientRole>
<!-- Patient Details -->
<patient>
<name>Patient Name</name>
<administrativeGenderCode code="F" codeSystem="2.16.840.1.113883.5.1"/>
<birthTime value="Date of Birth"/>
</patient>

Table 2. *Cont.*

Output XML Document
<!-- Signs and Symptoms Details -->
<observation classCode="OBS" moodCode="EVN">
<!-- Hypoventilation -->
<code code="267036007" codeSystem="2.16.840.1.113883.6.96"
displayName="Hypoventilation"/>
<value xsi:type="ST">Mild</value>
</observation>
<observation classCode="OBS" moodCode="EVN">
<!-- Heart Rate -->
<code code="8867-4" codeSystem="2.16.840.1.113883.6.1" displayName="Heart rate"/>
<value xsi:type="PQ" value="95" unit="/min"/>
</observation>
<observation classCode="OBS" moodCode="EVN">
<!-- Respiratory Rate -->
<code code="9279-1" codeSystem="2.16.840.1.113883.6.1" displayName="Respiratory rate"/>
<value xsi:type="PQ" value="28" unit="/min"/>
</observation>
<!-- Lymph Nodes Details -->
<observation classCode="OBS" moodCode="EVN">
<code code="102942000" codeSystem="2.16.840.1.113883.6.96" displayName="Lymph node"/>
<value xsi:type="ST">Inguinal with Abnormal Behavior</value>
</observation>
<observation classCode="OBS" moodCode="EVN">
<code code="102942000" codeSystem="2.16.840.1.113883.6.96" displayName="Lymph node"/>
<value xsi:type="ST">Preescapular with Normal Signs</value>
</observation>
</patientRole>
</recordTarget>
</ClinicalDocument>

4. Methodology for Evaluating Architectural Design of API's Architecture System Using Predictive Modeling and Graph Theory-Based Metrics

We employed the proposal in [28] to evaluate the performance of the architectural design in the proposed system. This involved using a predictive model to establish a quality framework for assessing object-oriented systems during the design stage. The assessment incorporated software architecture analysis through graph theory and software metrics, with a specific focus on complexity and coupling. The following steps were tailored for a system consisting of all classes: Factory_HCE, HCE, Wrapper_Medical_History, CDA, Patient_Identification_Data, Laboratory_Results, and Personal_Medical_History.

4.1. Methodology Applied

I. *Creation of the ACG (architectural complexity graph)*: The architectural design is visualized in Figures 2 and 3, representing the six classes in the system. UML diagrams served as the input to construct the ACG.

II. *Assignment of the complex attribute on the ACG*: CCM metrics were used as weights in the ACG (Refer to Figure 5). The Floyd–Warshall algorithm was applied to estimate critical paths and eighteen critical paths were identified.

III. *Assignment of the CBO metric on localized paths*: Following the proposal, performance architectural factors for critical paths were determined by calculating the Spearman correlation coefficient between the CBO and CCM data groups in each localized path. Table 3 presents some path results, featuring the identification path in the first column, CCM and CBO metrics in the next column, and the quality factor results in the last column. Spearman's correlation evaluates the relationship between CCM and CBO as ordinal variables, providing values on a spectrum between −1 to +1. A value closer to −1 indicates a weak relationship (low performance), while a value closer to +1 suggests a stronger relationship and higher performance.

IV. *Inference of performance parameter*: Table 3 displays six critical paths (1, 2, 3, 4, 5, and 18) as examples of the results, including the ID of the critical paths, component sequences with CMM and CBO metrics, and quality parameters. All results served as input data for the scatter dispersion plot in Figure 6.
The Factory_HCE class was included in the study. Being an abstract class, metrics of 1 in CCM and 0 in CBO were assigned. Eighteen critical paths were identified, in seventeen of which, the efficiency evaluation parameter had values close to +1, while only one path had a value of zero according to the scatter plot. Furthermore, these paths ultimately led to only four active and dynamic nodes. In summary, this adapted methodology provides insights into the architectural design quality of a system with six components. It utilizes metrics and critical paths to infer the system's architectural performance, focusing on the attributes of complexity and coupling.

Table 3. Set the critical paths evaluated through performance parameters on API architecture.

ID Path	Sequence				Performance Parameter
1	v0	v1			0.9428
CBO	2	3			
CCM	1	1			
2	v0	v1	v2		0.7745
CBO	2	3	1		
CCM	1	1	1		
3	v0	v1	v2	v3	0.0
CBO	2	3	1	2	
CCM	1	1	1	1	
4	v0	v1	v2	v4	0.7745
CBO	2	3	1		
CMM	1	1	1		
5	v0	v1	v2	v5	0.7745
CBO	2	3	1		
CMM	1	1	1		
⋮		⋮			⋮
18	V2	V6			1.0
CBO	1	0			
CMM	1	0			

4.2. Discussion

In this proposal, an object-oriented approach was selected, and design patterns were incorporated to enhance the performance of the software architecture of the medical interoperability API. In this case, the primary concern was addressing the context of inherited medical interoperability that involved medical centers or hospitals with devices that have limited memory capacity for patient data. Emphasis was placed on optimizing the design through the use of dynamic objects in memory generated at runtime. This approach differs from others where all system functions are loaded into memory, potentially negatively impacting the API's operational dynamics due to potential memory saturation on the executing devices.

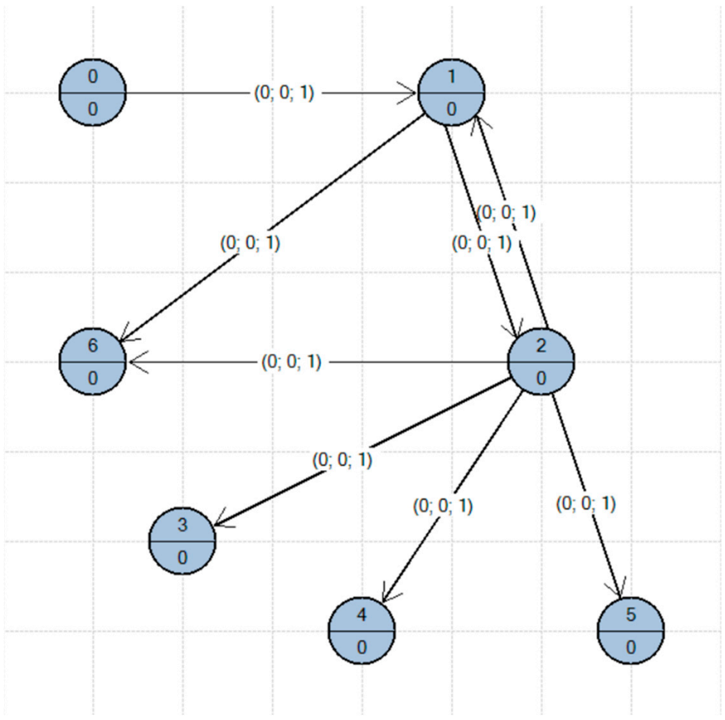

Figure 5. Interoperability API's ACG.

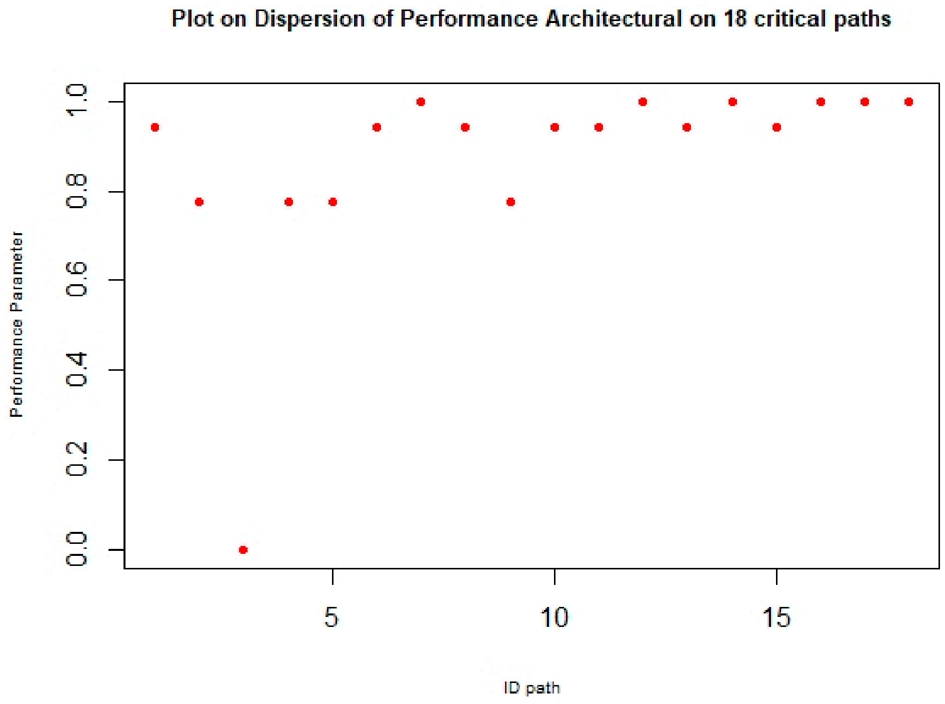

Figure 6. Dispersion performance of API architectural design.

To validate and quantify the proposed design approach, a software architecture evaluation method was integrated based on design quality attributes, such as complexity and coupling. The selected proposal for analysis emphasized a complexity approach strengthened with coupling, with the understanding that this relationship is linked to performance and is a desired attribute for application performance. The design evaluation was static, as the source code was not available during this phase. The evaluation was conducted through UML diagrams, algorithms, and other design artifacts. Evaluating the design was advantageous due to cost reduction, defect localization and correction before code implementation, and early test prioritization based on components identified with a higher probability of failure.

The architectural design evaluation applied in this research established a predictive model and a formal method to assess the quality in object-oriented systems. The architectural complexity graph (ACG) was introduced for a deterministic analysis, serving as a basis to establish the performance parameter. Component sequences with a parameter close to +1 were considered to have a stronger relationship between complexity and coupling attributes, indicating a good level of performance. As shown in the plot of the dispersion performance of the API architectural design in Figure 6, implementing abstract factory and wrapper design patterns in the proposed API architecture led to 94.5 percent of the evaluated sequences being close to +1, suggesting a good level based on complexity and coupling. We considered the approach logically valid for indirectly studying the performance of software design architectures.

4.3. Research Limitations and Future Trends

This proposal addresses the pressing need for interoperability in medical information, which is particularly underscored in the post-COVID-19 era, where its pivotal role in facilitating accurate diagnoses and effective treatments for patients is widely acknowledged. The primary objective was to enhance the architecture of the new medical interoperability API software through the utilization of an object-oriented approach and the seamless integration of design patterns. The emphasis was placed on refining the design to accommodate devices with limited memory capacity. Furthermore, this initiative laid down a robust framework for early evaluation and enhancement of the software architectural design, thus mitigating costs in the API development lifecycle. However, a significant challenge arose during the design phase that stemmed from the difficulty in assessing the practical effectiveness of the proposed architecture in real-world scenarios. The evaluation predominantly relied on static methods using predictive models and formal techniques. Additionally, practical implementation may face hurdles due to resource constraints, compatibility issues with existing platforms, and the intricate nature of integration with established healthcare systems.

Among the programming languages where the proposed architecture was suggested to be implemented, Java stood out as a strong candidate. It was designed from its inception to be interoperable, promoting portability and integration across different systems and services through standards, abstractions, and a common virtual machine. During the testing phase, the analysis model utilizing the proposed GCA acted as a guiding framework. This model, coupled with functional analysis weighted with design metrics, delineated the operational sequences of the API, offering early insights for crafting the test matrix.

We worked with the HL7 CDA standard [13], which is based on specific clinical documents. Its data model is structured around segments and fields, and the standard specifies how they should be organized and transmitted in messages. However, CDA's rigid structure limits its adaptability to different contexts and data requirements, particularly when handling large volumes of information. Other standards are also widely utilized and offer complementary features in managing health information, such as OpenEHR. This standard focuses on creating and managing electronic health records (EHRs) and semantically representing clinical information. OpenEHR utilizes a data model based on archetypes, which are semantic constructs that define the structure and meaning of clinical information. This enables a more robust and structured representation of health data. In this scenario, the API must integrate with other standards to function effectively.

5. Conclusions

At a global level, there is a significant need for medical interoperability. However, one of the major challenges is the heterogeneity of the systems that store patient information in medical centers or hospitals. Memory management becomes a key challenge in the operation of the proposed API, as it needs to seamlessly execute on any device, ranging from healthcare units like mobile phones to servers in cloud computing. In this proposal, an object-oriented approach was chosen, and design patterns were integrated to enhance the performance of the software architecture of the medical interoperability API. The implementation of design patterns, such as abstract factory and wrapper, in the proposed API architecture provided the capability to clone itself, generating instances at runtime and distributing each of these instances to medical centers where patient information is required. Once contact is established with the medical centers, the API retrieves patient information by encapsulating details such as controlled personal data, like patient identification data, a list of previous illnesses or conditions, records of performed medical procedures, and reports from imaging studies (X-rays, MRIs, etc.). The information obtained through the instances at each medical center returns to the requesting medical center and consolidates the information for the requisition. During the architectural design evaluation, this study reported a 94.5 percent performance, which was indirectly demonstrated through the assessment of operation sequences, suggesting a good level based on complexity and coupling. Based on the results obtained, we can conclude that this approach optimizes memory and time resources, improving the efficiency for object construction during execution in heterogeneous environments. This approach increases the efficiency of memory usage and response time in the exchange of electronic records, which impacts the improvement of efficiency in hospital care responses. Therefore, we concluded that HL7 is a suitable standard for managing interoperability. However, it needs to be complemented to enhance and facilitate the management of the semantic representation of electronic health records (EHRs).

In future work, the validation of the proposed design in real-world contexts is imperative to ensure its practical utility. The proposed architecture will be implemented in the Java language, and other protocols will be included in its format, such as OpenEHR, which is widely used in the development of medical interoperability APIs. This pivotal aspect involves equipping the architecture with mechanisms to adapt to diverse existing medical interoperability standards. The builder design pattern emerged as a promising solution to address this requirement.

Author Contributions: Conceptualization, L.D.N. and J.E.M.H. Methodology, L.D.N. and I.A.J. Design, L.D.N., J.E.M.H. and A.B.M. Validation, A.B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to privacy.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Valero, C.I.; Medrano Gil, A.M.; Gonzalez-Usach, R.; Julian, M.; Fico, G.; Arredondo, M.T.; Stavropoulos, T.G.; Strantsalis, D.; Voulgaridis, A.; Roca, F.; et al. AIoTES: Setting the principles for semantic interoperable and modern IoT-enabled reference architecture for Active and Healthy Ageing ecosystems. *Comput. Commun.* **2021**, *177*, 96–111. [CrossRef]
2. Khan, W.A.; Khattak, M.S.; Magbool, L.H.; Bilial, A. Achieving interoperability among healthcare standards: Building semantic mappings at models level. In Proceedings of the 6th International Conference on Ubiquitous Information Management and Communications ICUIMC'12, Kuala Lumpur, Malaysia, 20–22 February 2012.
3. Benson, T. *Principles of Health Interoperability HL7 and SNOMED*, 1st ed.; Springer: London, UK, 2010; pp. 1–263.
4. OpenEHR. Available online: https://www.openehr.org (accessed on 28 February 2024).
5. Lu, D.F.; Eichmann, D.; Konicek, D.; Park, H.T.; Ucharattana, P.; Delaney, C. Standardized nursing language in the systematized nomenclature of medicine clinical terms: A cross-mapping validation method. *Comput. Inform. Nurs.* **2006**, *24*, 288–296. [CrossRef] [PubMed]

6. Trigo, J.D.; Chiarugi, F.; Iglesias, A. Transmisión de ECGs en tiempo real basada en estándares: Armonización de ISO/IEEE 11073-PHD y SCP-ECG. In Proceedings of the XXVII Congreso Anual de la Sociedad Espanola de Ingenieria Biomedica, Cadiz, Spain, 15–16 January 2010.
7. Yuksel, M.; Dogac, A. Interoperability of Medical Device Information and the Clinical Application: An HL7 RMIM based on the ISO/IEEE 11073 DIM. *IEEE Trans. Inf. Technol. Biomed.* **2011**, *15*, 557–566. [CrossRef] [PubMed]
8. Martinez-Espronceda1, M.; Martinez, S.; Led, S. INTENSA: Sistema de Monitorización de Pacientes con Insuficiencia Cardíaca basado en el Estándar ISO/IEEE11073. Available online: http://diec.unizar.es/intranet/articulos/uploads/INTENSA:%20Sistema%20de%20monitorizacion%20de%20pacientes%20con%20insuficiencia%20cardiaca%20basado%20en%20el%20estandar%20ISO-IEEE11073.pdf (accessed on 28 February 2024).
9. Koncar, M. Implementing the HL7 v3 standard in the Croatian primary Healthcare domain. *Stud. Health Technol. Inform.* **2004**, *105*, 325–336. [PubMed]
10. Newsletter, HL7 Europe. Available online: http://www.hl7.eu (accessed on 28 February 2024).
11. Joune, A. Development of an Interoperable Exchange, Aggregation, and Analysis Platform for Health and Environmental Data. Master's Thesis, University of Applied Sciences Technikum Wien, Wien, Austria, 2017.
12. Aguilar, R.A.; López, D.M. *HL7 Implementation Guide for Public Health Reporting Systems in Colombia*; Technical Report; Grupo de Ingenieria Telematica, Universidad del Cauca: Cauca, Colombia, 2009.
13. Fuentes, I.; Guevara, D.; García, M. Toma de decisions inteligente a partir de registros medicos alamacenados en CDA-HL7. *Rev. Cuba. Informática Médica* **2016**, *8*, 109–124.
14. Velázquez, M.; Vázquez, M.; Nieto, J.; Sanchez, J. Modelo de interoperabilidad dela historia clinica electrónica utilizando HL7-CDA basado en computación en la nube. *Res. Comput. Sci.* **2015**, *108*, 37–44. [CrossRef]
15. Shin, D.I.; Pak, P.J.; Huh, S.J. The Mobile Implementation of HL7 API for u-Healthcare Devices. In *World Congress on Medical Physics and Biomedical Engineering*, 1st ed.; Shin, D.I., Hug, S.J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 1, pp. 110–111.
16. Viangteeravat, T.; Anyanwu, M.N.; Nagisetty, V.R. Clinical data integration of distributed data sources using Health Level Seven (HL7) v3-RIM mapping. *J. Clin. Bioinform.* **2011**, *1*, 32. [CrossRef] [PubMed]
17. Schadow, G.; Mead, N.; Walker, M. The HL7 Reference Information Model Uner Scrutiny. *Stud. Health Technol. Inform.* **2006**, *124*, 151. [PubMed]
18. Smith, B.; Ceusters, W. HL7 RIM: An Incoherent Standard. In Proceedings of the MIE 2006, Studies in Health Technology and Informatics 124, Maastricht, The Netherlands, 27–30 August 2006; pp. 133–138.
19. Beale, T.; Frade, S.; Leslie, H. Archetypes: Constraint-based Domain Models for Future-proof Information Systems. Openehr Foundation: London, UK, 2008.
20. Nicholson, N.; Perego, A. Interoperability of population-based patient registries. *J. Biomed. Inform.* **2020**, *112*, 100074. [CrossRef] [PubMed]
21. Banaee, H.; Ahmed, M.U.; Loutfi, A. Data mining for wearable sensors in health monitoring systems: A review of recent trends and challenges. *Sensors* **2013**, *13*, 17472–17500. [CrossRef]
22. Cheng, A.C.; Duda, S.N.; Taylor, R.; Delacqua, F.; Lewis, A.A.; Bosler, T.; Johnson, K.B.; Harris, P.A. REDCap on FHIR: Clinical Data Interoperability Services. *J. Biomed. Inform.* **2021**, *121*, 103871. [CrossRef]
23. Jacobson, D.; Brail, G.; Woods, D. *APIs: A Strategy Guide*; Oreilly & Associates: Sebastopol, CA, USA, 2012; pp. 1–134.
24. Biehl, M. *RESTful API Design: Best Practices in API Design with REST*, 1st ed.; Volume 3 of the API-Univerity Series; API-University Press: Rotkreuz, Switzerland, 2018; pp. 1–327.
25. Praschl, C.; Pointner, A.; Baumgartner, D.; Zwettler, G.A. Imaging framework: An interoperable and extendable connector for image-related Java frameworks. *SoftwareX* **2021**, *16*, 100863. [CrossRef]
26. Saleh, H.; Attakorah, J.A.; Zykov, S.; Legalov, A. Exploring the Eolang-Java Integration and Interoperability. *Procedia Comput. Sci.* **2021**, *192*, 4560–4569. [CrossRef]
27. Health Level Seven International (HL7). *Implementation Guide for CDA Release 2: Imaging Integration Levels 1, 2, and 3: Basic Imaging Reports in CDA and DICOM*; Health Level Seven International (HL7): Ann Arbor, MI, USA, 2015.
28. Davila-Nicanor, L.; Aguilar, I.; Ayala, J.; Banda-Madrid, A.; Cruz, S. Performance on Software Architecture Design to Serious Games for Mobile Devices. In *Software Engineering for Games in Serious Contexts*, 1st ed.; Cooper, K., Bucchiarone, A., Eds.; Springer Nature: Cham, Switzerland, 2023; Volume 1, pp. 63–84.