Enterprise Healthcare Systems (2014–2025)

11+ Years of Architectural Leadership in Regulated Healthcare Environment

1. CONTEXT & CONTRIBUTION

Client: Healthcare management operator (serving corporate clients: IBM, Nestlé, Aon, Boticário, Modec, etc.)
Environment: Mission-critical healthcare operations (zero downtime tolerance)
Project Status: Ongoing enterprise systems with legacy evolution, complete platform rewrite, security hardening across systems, enterprise integrations, compliance (LGPD/GDPR, TISS, ANS, ISO 27001), and cloud migration

Contribution Scope

• Progressed into technical leadership with architectural ownership
• Led development team (4 developers) during portion of rewrite phase
• Maintained, refactored, and evolved legacy systems (ASP Classic, WebForms, WinForms, Delphi) with design patterns, SOLID principles, and OO practices
• Implemented new features and projects using modern architecture practices
• Led ISO 27001 certification process (as one of the heads, with DPO)
• Implemented LGPD/GDPR compliance (encryption in transit/rest, data anonymization, pentest-hardened)
• Participated in Azure migration pre-requirements (infrastructure migration by specialized partner)

2. DEVELOPMENT SCOPE

Legacy System Modernization

• Extended legacy ASP Classic lifespan through strategic refactoring (design patterns, SOLID principles, OO practices, security hardening)
• Evolved ASP.NET WebForms platform for complete healthcare management (beneficiaries, exams, hospitalizations, medical records) with refactoring to OO, design patterns, and SOLID principles
• Improved UI consistency and user experience through Bootstrap CSS framework adoption
• Enhanced UX by implementing AJAX/jQuery features consuming ASP.NET SOAP WebServices (eliminated full page postbacks/reloads)
• Modernized WinForms ETL applications with refactoring and new features using design patterns and SOLID principles

• Replaced fragile Delphi Forms data importer with .NET solution:
  - Integrated Office 365 email using Microsoft Graph .NET Client Library
  - Automated hospitalization and exam data ingestion from Amil

Platform Rewrite & Modern Architecture

• Established new platform foundation using DDD and Clean Architecture
• Built modular architecture with separation of concerns (API, Domain, Data, Infrastructure, Cross-cutting layers)
• Evolved presentation layer from ASP.NET MVC to fully decoupled ASP.NET WebAPI
• Enabled React frontend integration through collaborative development

Secure Enterprise Integrations

• Designed and led main data importers for beneficiaries and medical records
• Established secure WebAPI with modular architecture for standardized JSON data imports
• Implemented PGP encryption (4096-bit keys) using Cinchoo PGP for .NET library
• Enabled flexible data reception (encrypted and decrypted modes)
• Restricted API access to credentialed companies only (authentication required)
• Standardized data import process (operator-specific layouts → unified standard)
• Developed WinForms processor for imported data validation
• Delivered automated IBM integration:
  - Console Application (C#) for employee medical license data reception
  - Integrated company SFTP (hosted in Azure) with PGP encryption (4096-bit keys)
  - Automated email notifications to employees and managers
  - Collaborated with IBM international team for integration requirements

Compliance & Billing Systems

• Implemented TISS-XML invoice generation for electronic billing
• Ensured data consistency across exams, hospitalizations, and beneficiary records
• Supported billing accuracy for healthcare operator clients
• Implemented LGPD/GDPR compliance: encryption in transit, encryption at rest, data access controls
• Achieved ISO 27001 certification (organizational ISMS - Information Security Management System)
• Established PGP encryption standard for API (HTTPS) and SFTP data transmission
• Developed and executed security remediation action plan after initial pentest
• Achieved zero successful breaches in subsequent annual pentests
• Supported annual audits by enterprise clients (IBM, Nestlé, Aon, Boticário, Modec)

DevOps & Cloud Migration

• Participated in Azure migration pre-requirements and preparation
• SQL Server migration to Azure Virtual Machine executed by infrastructure team
• Established CI/CD pipelines using Azure DevOps (Pipelines, Repos, Boards)
• Migrated version control from Gitea to Azure DevOps Git
• Automated deployment workflows
• All systems protected by Azure WAF (Web Application Firewall)

3. ISO 27001 CERTIFICATION PROCESS

Certification Scope

• Organizational ISMS (Information Security Management System)
• Healthcare platform and associated processes
• Certified entity: Healthcare operator organization (not the software platform itself)

Timeline & Phases

• Gap analysis: Initial assessment of existing controls vs. ISO 27001 requirements
• Documentation: Policies, procedures, risk assessments, Statement of Applicability
• Implementation: Security controls, access management, incident response, monitoring
• Internal audit: Third-party consultancy conducted internal audit to verify readiness (external consultancy hired to guide certification process + perform pre-audit)
• External audit: Certification body assessment (Stage 1 + Stage 2)
• Certification: ISO 27001 certificate issued
• Surveillance: Annual audits to maintain certification

My Role (as one of the heads)

• Collaborated with DPO on security controls alignment (LGPD + ISO 27001)
• Implemented technical controls (encryption, access control, logging, monitoring, data anonymization)
• Documented security procedures and operational guidelines
• Participated in internal audits and remediation activities (with third-party consultancy guidance)
• Supported external certification audits (evidence collection, technical interviews)
• Maintained controls post-certification (continuous improvement)

Key Technical Controls Implemented (Development Team Responsibility)

• Access control: RBAC (Role-Based Access Control) + Policy-Based Access Control
  - Roles defined per job function (Nurse, Doctor, Administrator, etc.)
  - Each Role assigned N granular policies
  - UI sections rendered dynamically based on user's specific policies
  - Backend API enforcement: Policy validation on every request, not just UI hiding
  - Least-privilege principle: Users only accessed data necessary for their function
• Encryption at rest: TDE (Transparent Data Encryption) for full database encryption - database backups unusable without encryption keys/credentials
• Encryption in transit:
  - HTTPS/TLS for all API communications
  - PGP 4096-bit for SFTP file transfers
  - PGP 4096-bit for API importer (optional encrypted mode)
  - API importer supported TWO submission modes:
    a) Encrypted mode: Client encrypts data with PGP before sending (recommended)
    b) Decrypted mode: Client sends plaintext over HTTPS/TLS only (fallback)

  - Rationale: Not all clients had PGP capability; offered secure option for enterprise clients (IBM, Rede D'Or, Boticário,etc.) while maintaining accessibility for smaller operators
• Key management: Encryption keys stored separately from data, access restricted to authorized personnel only
• Logging & monitoring: Comprehensive audit trail system (application layer)
  - Login control with after-hours prevention:
    * Successful logins (timestamp, user) - logged to database
    * Failed login attempts: 3 attempts triggered temporary lockout (few minutes)
    * After-hours access PREVENTED: User unable to login outside designated work schedule (per HR records) - control enforced at authentication layer
    * Login attempts outside work hours: Logged as blocked/denied (not just logged as access)
  - Data operation logging: ALL database operations logged regardless of RBAC+Policies authorization
    * Logged: User ID + executed query
    * Reports: Available filtered by user (developed by another team member)
  - Audit logs stored in: Separate audit tables in database
  - Log retention: Audit logs retained for 5+ years (compliance requirement)
  - Log integrity: Audit logs write-only (no UPDATE/DELETE permissions)
  - Log access: Restricted to Dev Team, Infrastructure Team, and DPO
    * External auditors: No direct access; evidence provided upon request
• Data anonymization for non-production environments:
  - Process: Production database mirrored to Development and Staging environments
  - Anonymization applied BEFORE mirroring (automated script/procedure)
  - Fields anonymized:
    * Personal identifiers: Names, CPF, phone numbers, addresses, emails
    * Health information: Replaced with dummy text (non-sensitive placeholder data)
  - Rationale: Developers need realistic data structures for testing, but real PHI/PII should not be accessible in non-production environments
  - Compliance: Satisfied both ISO 27001 (access control) and LGPD (data minimization, purpose limitation)
  - Outcome: Development team could test with realistic data volumes without accessing sensitive personal/health information
• Security event monitoring & alerting: Zabbix integration (Infrastructure Team)
  - Infrastructure team deployed and managed Zabbix for real-time monitoring
  - Alert channels: Microsoft Teams (dedicated channel for development team)
  - Development team received pertinent alerts (application errors, API failures, deployment issues)
  - Infrastructure team managed: Server health, database performance, network issues, authentication failures
  - Alert response: Documented escalation matrix with SLA per severity
• Incident response: Security incident procedures, escalation matrix, post-incident reviews (collaborative with Infra + DPO)
  - Escalation path: On-call developer → Tech Lead → DPO → CTO → CEO
  - Post-incident review: Root cause analysis, corrective actions, documentation
• Vulnerability management: Annual third-party pentests, remediation action plans
  - Pentest scope: Web application, API, infrastructure, social engineering

  - Findings classification: Critical, High, Medium, Low, Informational
  - Remediation SLA: Critical (48 hours), High (1 week), Medium (2 weeks)
  - Zero successful breaches post-remediation (all annual pentests passed)
• Backup & recovery: Managed by Infrastructure Team
  - Encrypted backups (TDE-encrypted database backups)
  - Disaster recovery procedures in place
  - Details managed by Infrastructure Team (Azure storage, retention, testing)
• Security Hardening & Pentest Remediation:
  - Initial pentest findings (critical/high vulnerabilities):
    * Weak password policy: No complexity requirements, allowed brute-force
    * SQL Injection: User table exposed, passwords stored in plaintext
    * User impersonation: User ID passed via query string without session validation
    * Unrestricted file upload: Executables allowed in image/document upload fields
    * Missing CORS headers: Cross-origin requests not restricted
    * Cross-Site Scripting (XSS): User input not sanitized before rendering
  - Remediation actions implemented:
    * Authentication overhaul: Minimum 8 characters, uppercase + special character required, password hashing, 2FA implementation, 3-attempt lockout (temporary block for X minutes), mandatory password rotation (90 days), password history enforcement (cannot reuse previous passwords)
    * SQL injection prevention: Parameterized queries implemented incrementally (high-traffic features first, eventually 100% coverage)
    * Session validation: All user ID parameters validated against session variables on every request
    * File upload validation: File signature/magic number validation (not just extension), whitelist of allowed types (jpg, png, pdf, docx, xlsx)
    * CORS headers: Restricted to carelink.com.br origin only
    * XSS prevention: ASP.NET WebAPI DataAnnotations + output encoding
    * Custom error pages in production (no stack trace exposure)
    * All traffic enforced over HTTPS
    * Azure WAF protection for all systems
  - Outcome: Zero successful breaches in all subsequent annual pentests

Architectural Trade-Off: TDE vs. Column-Level Encryption

• Evaluated: Column-level encryption for sensitive fields (CPF, medical records, etc.)
• Decision: TDE only (full database encryption)
• Rationale: Column-level encryption would significantly impact read/write performance on high-volume healthcare operations (exams, hospitalizations, billing)
• Outcome: TDE satisfied ISO 27001 + LGPD requirements without performance degradation

Outcome

• ISO 27001 certification achieved (organizational ISMS)
• Zero successful breaches post-certification
• Annual surveillance audits passed successfully
• Enterprise client audits satisfied (IBM, Nestlé, Aon, Boticário, Modec)
• Database restoration impossible without encryption keys (enforced security)

• Login lockout (3 attempts) prevented brute-force attacks
• After-hours access BLOCKED at authentication layer (preventive control)
• Audit logs provided evidence for all certification audits
• Third-party consultancy guided certification process (internal audit + readiness)
• Data anonymization prevented developer access to PHI/PII in non-production
• Azure WAF provided additional layer of protection for all systems

4.TECHNICAL LEADERSHIP

Team Leadership

• Led development team (4 developers) during portion of platform rewrite phase
• Mentored team on SOLID principles, design patterns, and best practices
• Conducted code reviews to ensure architecture compliance
• Facilitated knowledge sharing on modern architecture patterns

Standards & Practices

• Established coding standards for new platform (naming, structure, documentation)
• Defined API design patterns for consistency across integrations
• Implemented CI/CD pipelines to automate deployment and reduce human error
• Promoted testability and maintainability through Clean Architecture adoption

Stakeholder Management

• Presented complete platform rewrite proposal to CEO with business value justification (all approved)
• Collaborated with enterprise client development teams (IBM international, Rede D'Or, Amil, Unimed, Boticário, Bradesco) for integration requirements
• Participated in security audits with enterprise clients (IBM, Nestlé, Aon, Modec, etc.)
• Supported ANS compliance requirements (healthcare regulatory body)

5. ARCHITECTURAL DECISIONS & TRADE-OFFS

Platform Architecture: DDD/Clean vs. Hexagonal vs. Microservices vs. Event-Driven

• Decision: Adopt DDD and Clean Architecture for new platform
• Alternatives Considered:
  - Hexagonal Architecture: Similar separation of concerns, but Clean was more established in .NET community at the time
  - Microservices: Evaluated but rejected due to team size (4 developers) and deployment complexity overhead
  - Event-Driven Architecture: Considered for integrations, but added complexity (message broker, event schema, retry logic) not justified for synchronous use cases at the time
  - Modular Monolith: Viable middle-ground, but Clean Architecture provided clearer

separation of concerns
• Trade-off: Higher initial development time vs. long-term maintainability and scalability
• Rationale:
  - DDD provided ubiquitous language for healthcare domain (beneficiaries, exams, hospitalizations, billing)
  - Clean Architecture enabled separation of concerns (API, Domain, Data, Infrastructure,Cross-cutting)
  - Team size and complexity justified modular approach over microservices
  - Event-driven patterns could be added later if needed
• Impact: New features became easier to implement, testing improved(integration and unit), team onboarding faster, architecture flexible enough for future evolution

API Architecture: WebAPI (REST) vs. gRPC vs. GraphQL vs. SOAP

• Decision: Adopt ASP.NET WebAPI with REST for new platform integrations
• Alternatives Considered:
  - gRPC: Evaluated for internal service communication, but REST was more universal for external integrations (health operators, enterprise clients)
  - GraphQL: Considered for frontend flexibility, but added complexity not justified for healthcare data structures
  - SOAP (only): Legacy systems used SOAP, but REST provided better web/mobile compatibility
• Trade-off: REST verbosity vs. universal compatibility and client simplicity
• Rationale:
  - REST/WebAPI enabled multiple frontend types (Web, mobile, 3rd-party)
  - Health operators and enterprise clients had broader REST tooling and expertise
  - SOAP maintained for legacy integrations (backward compatibility)
  - gRPC/GraphQL could be adopted later if specific needs emerged
• Impact: Frontend-backend separation enabled independent development cycles, React integration enabled, future mobile/3rd-party integrations simplified

Data Storage: Azure VM (SQL Server) vs. Azure SQL vs. Cosmos DB vs. On-Premises

• Decision: SQL Server to Azure Virtual Machine (executed by infrastructure team)
• Alternatives Considered:
  - Azure SQL (PaaS): Evaluated for reduced operational overhead, but required significant refactoring (compatibility issues, connection string changes)
  - Cosmos DB: Considered for scalability, but relational data model (beneficiaries, exams, billing) was better suited for SQL Server
  - On-Premises (continued): Would maintain infrastructure maintenance burden and scalability limitations
• Trade-off: Higher operational overhead (VM management) vs. minimal code changes and faster migration timeline
• Rationale:
  - Lift-and-shift to VM allowed migration without application refactoring
  - Relational data model (exams, hospitalizations, billing) required SQL Server
  - Azure SQL planned for future phase (not completed before departure)

- Cosmos DB not justified for structured healthcare data at the time
• Impact: Faster migration timeline, reduced infrastructure maintenance, Azure SQL planned for future phase

Presentation Layer: WebAPI vs. MVC

• Decision: Evolved from ASP.NET MVC to fully decoupled ASP.NET WebAPI
• Trade-off: Additional abstraction layer vs. flexibility for multiple frontends
• Rationale: React frontend required REST API; WebAPI enabled future mobile/3rd-party integrations
• Impact: Frontend-backend separation enabled independent development cycles

Data Import: Standardized API vs. Operator-Specific Layouts

• Decision: Created WebAPI with unified JSON schema
• Trade-off: Initial effort to map all operator layouts vs. ongoing manual errors
• Rationale: Each operator had different file formats, causing frequent import failures
• Impact: Reduced manual data entry errors, standardized validation rules

Security: Remediation Plan vs. Quick Fixes

• Decision: Developed comprehensive remediation action plan after initial pentest
• Trade-off: Longer remediation timeline vs. sustainable security posture
• Rationale: Quick fixes would leave gaps; systematic approach ensured zero subsequent breaches
• Impact: Zero successful breaches in subsequent annual pentests

Version Control: Gitea → Azure DevOps Git

• Decision: Migrated from self-hosted Gitea to Azure DevOps Git
• Trade-off: Migration effort vs. integrated CI/CD pipeline
• Rationale: Azure DevOps offered end-to-end DevOps (Boards, Repos, Pipelines)
• Impact: Unified toolchain, automated deployments, better traceability

Legacy Modernization: Gradual Evolution vs. Big Bang Rewrite

• Decision: Maintained legacy while building new platform (parallel development)
• Trade-off: Longer overall timeline vs. zero business disruption
• Rationale: Mission-critical system could not have downtime during migration
• Impact: Continuous operations throughout modernization; zero downtime during transition

Data Transmission Security: PGP Encryption vs. Transport-Layer Only

• Decision: Implemented PGP encryption for data in transit (API + SFTP)
• Trade-off: Additional complexity for clients (key management) vs. end-to-end security
• Rationale: Transport-layer encryption (HTTPS/TLS) protects data in transit only; PGP ensures encryption from client source to decryption point, including data at rest on SFTP
• Impact: Enterprise clients (IBM, etc.) could meet their internal security requirements; data protected even if storage compromised

PGP Key Size: 4096-bit vs. 2048-bit

• Decision: Implemented 4096-bit PGP keys
• Trade-off: Slightly higher computational overhead vs. stronger security margin
• Rationale: Healthcare data is highly sensitive; 4096-bit provides longer-term security against advancing computational power; enterprise clients expect maximum security for PHI
• Impact: Met enterprise security requirements; future-proof encryption for long-term data sensitivity

UI Framework: Bootstrap CSS vs. Custom CSS

• Decision: Adopted Bootstrap CSS framework for legacy UI modernization
• Trade-off: Additional library dependency vs. faster, consistent UI development
• Rationale: Custom CSS for each page was inconsistent and time-consuming; Bootstrap provided ready-made components and responsive design
• Impact: Improved UI consistency across platform, reduced CSS maintenance effort

UX Enhancement: AJAX/jQuery vs. Full Page Postbacks

• Decision: Implemented AJAX/jQuery for partial page updates
• Trade-off: Additional JavaScript complexity vs. significantly improved user experience
• Rationale: Full page postbacks were slow and disruptive; AJAX enabled seamless interactions without page reloads
• Impact: Eliminated full page postbacks, improved user experience, reduced server load

Data Importer: .NET + Microsoft Graph vs. Continuing Delphi

• Decision: Rewrote Delphi Forms importer with .NET and Microsoft Graph .NET Client Library
• Trade-off: Development effort for rewrite vs. long-term maintainability and modern integration
• Rationale: Delphi application was outdated, fragile, and couldn't integrate with Office 365
• Impact: Modern, maintainable solution with automated email ingestion from Amil

API Security: Credentialed Access vs. Open API

• Decision: Restricted API access to credentialed companies only (authentication + encryption)
• Trade-off: Additional authentication overhead vs. controlled, auditable access
• Rationale: Healthcare data requires strict access control; open API would pose security risks
• Impact: Unauthorized access prevented; audit trail of all API consumers

CI/CD: Azure DevOps Pipelines vs. Manual Deployment

• Decision: Established CI/CD pipelines using Azure DevOps
• Trade-off: Initial pipeline setup effort vs. automated, reliable deployments

• Rationale: Manual deployments were error-prone and time-consuming; automation reduced human error and deployment time
• Impact: Automated deployment workflows, reduced human error, faster release cycles

TISS-XML: Automated Generation vs. Manual Billing

• Decision: Implemented automated TISS-XML invoice generation
• Trade-off: Development complexity vs. billing accuracy and compliance
• Rationale: Manual billing was error-prone and didn't meet regulatory requirements
• Impact: Accurate electronic invoicing in TISS-XML format, compliance with healthcare regulations

6. PROBLEMS SOLVED & IMPACT

Security Vulnerabilities → Zero Breaches

• Problem: Initial pentest identified exploitable vulnerabilities in legacy system:
  - Weak password policy (no complexity, no lockout)
  - SQL Injection (plaintext passwords exposed in User table)
  - User impersonation (query string manipulation without session validation)
  - Unrestricted file upload (executables allowed)
  - Missing CORS headers
  - Cross-Site Scripting (XSS) - unsanitized user input
• Solution: Developed and executed comprehensive remediation action plan:
  - Authentication overhaul (8+ chars, uppercase, special char, hashing, 2FA, lockout, 90-day password rotation, password history enforcement)
  - Parameterized SQL queries (incremental, 100% coverage)
  - Session validation on all user ID parameters
  - File signature validation (magic number, whitelist approach)
  - CORS headers (restricted to carelink.com.br)
  - XSS prevention (ASP.NET WebAPI DataAnnotations + output encoding)
  - Custom error pages (production), HTTPS enforcement, Azure WAF
• Impact: Zero successful breaches in all subsequent annual pentests

Manual Data Imports → Standardized API

• Problem: Each health operator used different file layouts, causing import failures
• Solution: Created WebAPI with unified JSON schema and validation rules
• Impact: Reduced manual errors, eliminated layout-specific troubleshooting

Page Postbacks → AJAX/jQuery UX

• Problem: Full page reloads on every user action (slow, poor UX)
• Solution: Implemented AJAX/jQuery features consuming SOAP WebServices
• Impact: Eliminated full page postbacks, improved user experience

Delphi Legacy Importer → .NET + Microsoft Graph

• Problem: Delphi application reading Amil emails was outdated and fragile
• Solution: Rewrote in .NET with Microsoft Graph .NET Client Library for Office 365

• Impact: Modern, maintainable solution with automated email ingestion

SFTP Integration with IBM
• Problem: Manual upload of employee medical license data
• Solution: Developed Console Application (C#) integrated with company SFTP (Azure)
• Impact: Automated data reception, reduced manual intervention

Enterprise Audit Requirements
• Problem: Annual audits by IBM, Nestlé, Aon, Boticário, Modec required audit-ready systems
• Solution: Implemented LGPD compliance, encryption, access controls, documentation
• Impact: All audits passed successfully; system maintained compliance status

TISS-XML Invoicing Accuracy
• Problem: Inconsistent beneficiary data caused billing errors
• Solution: Ensured data consistency across exams, hospitalizations, beneficiary records
• Impact: Accurate electronic invoicing in TISS-XML format

Secure Data Transmission → PGP End-to-End Encryption
• Problem: Sensitive healthcare data transmitted over internet to SFTP/API
• Solution: Implemented PGP encryption (4096-bit keys) using Cinchoo PGP for .NET
• Impact: Data encrypted at source, protected in transit, secure at rest; met enterprise client security requirements (IBM, etc.)

API Access Control → Credentialed Companies Only
• Problem: Import API needed to be accessible only to authorized health operators
• Solution: Implemented authentication + credential-based access control
• Impact: Unauthorized access prevented; audit trail of all API consumers

Legacy ASP Classic → Refactored with Design Patterns
• Problem: ASP Classic codebase was procedural, difficult to maintain and extend
• Solution: Refactored with design patterns, SOLID principles, and OO practices
• Impact: Extended legacy system lifespan, improved maintainability, enabled new features

ASP.NET WebForms → OO + SOLID Refactoring
• Problem: WebForms platform had mixed concerns, difficult to test and modify
• Solution: Refactored to OO with design patterns and SOLID principles
• Impact: Improved code quality, easier testing, faster feature development

UI Inconsistency → Bootstrap CSS Standardization
• Problem: Each page had custom CSS, inconsistent look and feel across platform
• Solution: Adopted Bootstrap CSS framework for UI consistency

• Impact: Improved UI consistency, reduced CSS maintenance, faster UI development

WinForms ETL → Refactored with Design Patterns
• Problem: ETL applications had code duplication, difficult to add new operators
• Solution: Refactored with design patterns and SOLID principles
• Impact: Easier to add new operators, reduced code duplication, improved maintainability

Platform Architecture → DDD + Clean Architecture
• Problem: Legacy monolith was difficult to test, extend, and maintain
• Solution: Built new platform with DDD and Clean Architecture
• Impact: New features easier to implement, improved testability, faster team onboarding

MVC → WebAPI Decoupling
• Problem: MVC presentation layer was coupled with business logic
• Solution: Evolved to fully decoupled ASP.NET WebAPI
• Impact: Frontend-backend separation, independent development cycles, React integration enabled

SQL Server Migration → Azure VM
• Problem: On-premises SQL Server required infrastructure maintenance
• Solution: Infrastructure team migrated to Azure Virtual Machine
• Impact: Faster migration timeline, reduced infrastructure maintenance, Azure SQL planned for future

Version Control → Gitea to Azure DevOps Git
• Problem: Self-hosted Gitea wasn't integrated with CI/CD pipeline
• Solution: Migrated to Azure DevOps Git
• Impact: Unified toolchain, automated deployments, better traceability

Parallel Development → Zero Downtime Migration
• Problem: Mission-critical system couldn't have downtime during modernization
• Solution: Maintained legacy while building new platform (parallel development)
• Impact: Continuous operations throughout modernization, zero downtime during transition

Data Encryption → PGP 4096-bit Standard
• Problem: Transport-layer encryption (HTTPS/TLS) doesn't protect data at rest on SFTP
• Solution: Implemented PGP encryption (4096-bit keys) for end-to-end protection
• Impact: Data protected from source to decryption point, met enterprise security requirements

CI/CD → Automated Deployments

• Problem: Manual deployments were error-prone and time-consuming
• Solution: Established CI/CD pipelines using Azure DevOps
• Impact: Automated deployment workflows, reduced human error, faster release cycles

Billing → TISS-XML Automation

• Problem: Manual billing was error-prone and didn't meet regulatory requirements
• Solution: Implemented automated TISS-XML invoice generation
• Impact: Accurate electronic invoicing, compliance with healthcare regulations

7. KEY TAKEAWAYS

Technical Growth

• Evolved from maintenance developer to technical leader and architectural decision-maker
• Led complete platform rewrite with modern architecture patterns
• Gained deep expertise in regulated healthcare environments
• Mastered legacy modernization strategies (evolution, refactoring, new features,OO practices, design patterns, SOLID principles, security hardening, UI improvements)

Leadership & Influence

• Led development team (4 developers) during portion of rewrite phase
• Platform rewrite proposal presented with business value justification(all approved by CEO)
• Mentored team on SOLID principles, design patterns, and best practices
• Established coding standards and API design patterns for consistency

Business Understanding

• Healthcare data criticality (maternal-infant, oncology, hospitalizations, chronic disease monitoring)
• Beneficiary data consistency directly impacts patient care quality
• Human health management alongside data management (care follow-ups)
• TISS-XML compliance essential for electronic billing accuracy
• Enterprise client expectations (IBM, Nestlé, Aon, Boticário, Modec) require audit-ready systems

Compliance & Security

• LGPD/GDPR compliance is non-negotiable in healthcare (encryption, access controls)
• ISO 27001 certification requires organizational commitment (security management system)
• Initial pentest vulnerabilities → Remediation plan → Zero subsequent breaches
• Annual pentests + audits are standard for enterprise healthcare clients
• Security hardening of legacy systems is as important as new development
• PGP encryption (4096-bit) for enterprise-grade data protection

• Password policies: 90-day rotation, history enforcement, complexity requirements
• Azure WAF provides additional protection layer for all systems
• Data anonymization for non-production environments (ISO 27001 + LGPD)

This 11-year specialization prepared me for
• Founder (2025-Present): Boutique software & technical strategy consulting
• Offering deep enterprise expertise across healthcare and other regulated industries

8. TECHNICAL EXPOSURE

Technologies & Tools
• .NET Framework 4.8, .NET Core 3.1
• C#, ASP.NET WebForms, WebAPI, SOAP WebServices
• SQL Server, T-SQL, Stored Procedures
• Azure (Virtual Machines, SFTP, WAF)
• Azure DevOps (Boards, Repos, Pipelines)
• Microsoft Graph .NET Client Library
• Cinchoo PGP for .NET
• JavaScript, jQuery, AJAX
• React (collaborative development)
• Bootstrap CSS
• WinForms, Console Applications
• SFTP, REST, SOAP integrations
• Git (Gitea → Azure DevOps)
• PGP Encryption (4096-bit keys)

Architecture & Practices
• DDD (Domain-Driven Design)
• Clean Architecture principles
• Modular separation (API, Domain, Data, Infrastructure, Cross-cutting)
• API-first design (WebAPI for integrations)
• CI/CD pipelines (Azure DevOps)
• Legacy modernization (evolution, refactoring, new features, OO practices, design patterns, SOLID principles, UI improvements, security hardening)
• Encryption in transit and at rest (LGPD/GDPR compliance)
• PGP encryption for end-to-end data protection
• Pentest remediation and hardening (zero breaches after remediation)
• Data anonymization for non-production environments
• Azure WAF configuration and management
• OWASP Top 10 mitigation (SQL Injection, XSS, broken authentication, insecure file upload, missing security headers)
• ASP.NET WebAPI DataAnnotations for input validation (Application layer DTOs)
• Domain layer business validations (DDD separation of concerns)

Healthcare-Specific

• TISS-XML for electronic invoicing
• Beneficiary health management (exams, hospitalizations, medical records, care follow-ups)
• Maternal-infant sector (ages 0-7)
• Oncology sector (chronic disease monitoring)
• Hospitalizations sector
• Chronic disease monitoring sector
• Enterprise client audits (IBM, Nestlé, Aon, Boticário, Modec)