MARIO GUZMÁN MESÉN

Software Development Manager · CTO — Project Portfolio

mguzmanmesen.com · mguzmanmesen@hotmail.com · Heredia, Costa Rica

This document presents a curated selection of projects and initiatives across my career — from hands-on software engineering to leading full product and infrastructure transformations. Each entry outlines the context, challenges, decisions made, and outcomes delivered.

FEATURED PROJECT — BZPAY: Payment Platform Overhaul

Imic Solutions · 2023 – Present · Software Development Manager / CTO

Joined as Software Development Manager and effectively operating as CTO to lead a full rebuild of a fintech payment platform — from team structure and engineering culture to cloud architecture and security compliance. Since joining, I've grown the technology department from **9 to 20 people** while the platform's daily transaction volume scaled from **~6,000 to ~13,000 (more than doubled)**.

Context & Challenge

- No established development process, team structure, or deployment pipeline.
- System required PCI DSS Level 1 certification — the highest tier in payment card security.
- Legacy codebase with a .NET Framework 4.6 monolith in need of modernization.
- Need to integrate with multiple external providers (American Express, Mexican payment network) via ISO 8583.

What I Built & Led

- Designed and implemented a full Microsoft Azure infrastructure: Front Door (WAF), **6 App Services** (Gateway + Backend + Translators), Azure SQL Server PaaS, Key Vaults, and isolated dev environments with deployment slots — supporting growth to **~13,000 daily transactions**.
- Led migration from .NET Framework 4.6 monolith to .NET Core 8 using a clean layered architecture (API → Domain → Repository → Infrastructure), decomposing the backend into 4 bounded services: Transactions, Merchants, Providers, and Users.
- Introduced a NodeJS 22 Translator service inside a private VLAN that converts backend requests to ISO 8583 messages for provider communication.
- Achieved PCI DSS Level 1 certification — coordinating across Infrastructure, CISO, and development teams.
- Implemented DUKPT encryption for card data and coordinated online PIN generation with external providers.
- Built the team from scratch — growing from **9 to 20 people** — defining career paths, performance review processes, salary frameworks, and 1:1 coaching structure.
- Established full deployment cadence, PR conventions, bug severity classification, and CI/CD pipelines validated under PCI compliance.

Outcome

- PCI DSS Level 1 certified — fully operational and audited across all 12 requirement domains.
- Daily transaction volume more than doubled (~6,000 → ~13,000) on a modern, maintainable architecture serving live transactions with multiple providers.
- A structured engineering team of 20 — more than double the original 9 — with clear processes, from ideation to production.

.NET Core 8 · Azure · NodeJS 22 · ISO 8583 · PCI Level 1 · Clean Architecture · DUKPT · SQL Server · CI/CD · People Management

Initiative — Technology Team Build-Out & Agile Transformation

Team design · Agile framework adoption · Cross-area communication

Led the design and formation of the entire technology and software development department from scratch — defining team structures, creating new leadership roles, establishing agile frameworks, and building the communication channels that connect IT with the rest of the organization. Over the course of this initiative, the department grew from **9 to 20 people**.

Team Design & Leadership

- Defined the full structure of the technology department: development teams, QA, infrastructure, and support — scaling headcount from **9 to 20**.

- Created new leadership positions to distribute management of people, processes, and delivery across the department.
- Established 1:1 meeting cadence for all team members — covering performance, growth, and well-being.
- Coordinated with HR to define career paths, learning plans, salary frameworks, and performance improvement processes.
- Recruited and onboarded team members aligned with both technical requirements and cultural fit.

Agile Framework Establishment

- Implemented Scrum as the primary framework for all new product development: Sprint planning, Daily Standups, Sprint Reviews, and Retrospectives — with consistent ceremonies and team coaching throughout adoption.
- Implemented Kanban for support and maintenance tracks, enabling continuous flow and clear prioritization of incidents and requests without Sprint overhead.
- Defined the full process for requirements gathering: stakeholder meetings, Feature definition, User Story writing with Acceptance Criteria, and backlog refinement.
- Established Story and Feature templates so teams could write consistent, actionable work items.
- Led Scrum adoption coaching — accompanying teams through the learning curve and reinforcing good practices through retrospectives, metrics review, and ongoing mentoring.

MS Azure DevOps Configuration

- Configured Microsoft Azure DevOps as the central platform for the entire department — boards, backlogs, Sprints, and pipelines all aligned to the Scrum and Kanban frameworks.
- Set up team projects, area paths, iteration paths, and work item hierarchies (Epic → Feature → Story → Task).
- Created dashboards and reports for Sprint progress, velocity tracking, and delivery visibility for management.
- Integrated Azure DevOps pipelines with the deployment cadence validated under PCI Level 1 compliance.

Cross-Functional Communication Channels

- Created formal communication channels between the technology department and Operations, Commercial, Finance, and Customer Service areas.
- Established recurring sync meetings with each area to gather requirements, communicate delivery progress, and align on priorities.
- Designed a Sprint Review format open to business stakeholders — giving visibility into what is being built and creating a feedback loop between IT and the rest of the organization.
- Reduced ad-hoc interruptions to development teams by routing requests through structured intake processes.

Outcome

- A fully structured technology department — grown from 9 to 20 people, from individual contributors to team leads — built from zero.
- Scrum and Kanban running consistently across all product and support tracks.
- Azure DevOps fully configured and adopted as the single source of truth for all delivery work.
- Strong collaboration between IT and business areas, with clear channels, shared visibility, and reduced friction.

Scrum · Kanban · Azure DevOps · People Management · Team Design · Agile Coaching · Requirements · User Stories · Leadership · Cross-functional

Initiative — New Backend Architecture: .NET Core 8 Clean Architecture

Full platform re-architecture · 12-month migration, ongoing device rollout

Designed and led the implementation of a completely new backend architecture for all BZPAY services, replacing a legacy monolith with a modern, maintainable Clean Architecture in .NET Core 8 — a **12-month effort**, still in progress as physical payment devices migrate without disrupting clients already live on the platform.

Context & Challenge

- Legacy monolith on .NET Framework 4.6 — tightly coupled, hard to test, difficult to scale.
- Existing API endpoints were inconsistent, undocumented, and not designed for PCI Level 1 security demands.
- Physical payment terminals consuming old endpoints that needed deprecation without disrupting live operations.
- Required support for multiple bounded contexts (Transactions, Merchants, Providers, Users) independently.

Architecture Design

- Presentation Layer (Web API): RESTful endpoints with JWT validation, route definitions, and response formatting.
- Domain / Application Layer: all business logic, use cases, and entities. Manages Redis and singleton caching.
- Repository / Router Layer: abstracts data access — routes requests to the database or third-party providers.
- Infrastructure Layer: manages outbound connections to internal microservices and Azure service clients.
- Database Context Layer: ORM mappings for Azure SQL Server covering transactions, clients, providers, and rules.
- Shared Layer: cross-cutting models, enums, constants, and utilities shared across all layers.

Services Built on This Architecture

- Gateway (.NET Core 8): public-facing entry via Azure Front Door — handles routing, auth, and request forwarding.
- Backend (.NET Core 8, private VLAN): core engine split into four services — Transactions, Merchants, Providers, Users.
- Translator (NodeJS 22, private VLAN): converts backend requests to ISO 8583 messages for payment providers.

Device Migration

- Coordinating migration of physical payment terminals from legacy endpoints to the new API.
- Backward-compatible transition plan designed to avoid service interruption during device rollout for clients already in production.
- New endpoints follow RESTful conventions, full documentation, and PCI-compliant security contracts.

Outcome

- Fully operational new architecture serving live production traffic at ~13,000 transactions/day.
- Clean separation of concerns — each layer independently testable and deployable.
- Device migration actively in progress after 12 months — new API being adopted across the terminal fleet with zero disruption to existing clients.
- Architecture that enabled PCI DSS Level 1 certification across all backend services.

.NET Core 8 · Clean Architecture · JWT · Redis · REST API · Azure SQL · ISO 8583 · NodeJS 22 · PCI Level 1 · Microservices

Initiative — PCI DSS Level 1 Certification

Policy · Process · Technical controls · Certification obtained

Led the end-to-end program to achieve PCI DSS Level 1 certification — the highest tier of payment card industry compliance, covering **all 12 PCI DSS requirement domains**. This was not a purely technical effort: it required redefining business priorities, creating organizational processes, assigning ownership, implementing technical controls, and building an audit trail across the entire department.

Business Priority Framework

- Established a clear priority model for all IT work to align delivery with business and compliance needs: Business Continuity (P1) → Compliance & Regulatory (P2) → Support & Incidents (P3) → Product Improvements (P4) → New Products & Features (P5).
- Used this framework to justify PCI work items against competing product demands — giving compliance a defined lane in the backlog without blocking business delivery.
- Created a dedicated PCI Feature in Azure DevOps to track all compliance work: policies, technical controls, process definitions, and audit evidence — each as a traceable Story with acceptance criteria.

Policy & Process Definition

- Authored and formalized security policies required by PCI DSS: access control, data handling, incident response, change management, and vulnerability management.
- Defined and documented operational processes for each policy area, with assigned process owners responsible for execution and ongoing compliance.
- Created a change management log — every infrastructure or code change tracked with requester, approver, date, scope, and rollback plan.
- Established an infrastructure and code review register — periodic reviews documented and signed off by designated owners.
- Defined deployment ownership: each production deployment has a responsible party, a log entry, and traceability back to the approved change.

Technical Controls Implemented

- Azure Front Door configured as WAF — enforcing OWASP rules and blocking malicious HTTP/HTTPS traffic at the perimeter.
- Azure Key Vault adopted for all secrets management — passwords, connection strings, SSL certificates, email accounts, and API keys classified and stored by sensitivity tier.
- Always Encrypted columns in Azure SQL Server for all sensitive cardholder data fields — data encrypted at rest and in transit, inaccessible even to DBAs.
- VPN and private VLAN isolation for all backend services — internal services unreachable from the public internet.
- Enforced PCI-accepted cipher suites across all TLS configurations — deprecated weak protocols (TLS 1.0/1.1, RC4, 3DES) across all App Services and endpoints.
- Implemented DUKPT (Derived Unique Key Per Transaction) encryption for card data at point of interaction.
- Purged all sensitive data — PANs, CVVs, and authentication data — from logs, databases, and application memory in compliance with PCI DSS storage prohibition requirements.

Inventories & Audit Evidence

- Built and maintained a full asset inventory: all Azure services, App Services, databases, VMs, and network components documented with owner, environment, and classification.
- Maintained a data flow diagram showing the movement of cardholder data across all system components.
- Documented all third-party service providers (American Express, payment networks) with their respective PCI responsibility matrix (shared responsibility model).
- Produced and maintained all audit evidence required for the Level 1 assessment — organized by PCI DSS requirement number for QSA review.

Outcome

- PCI DSS Level 1 certification obtained — the highest tier, covering all 12 PCI DSS requirement domains.
- Full organizational buy-in: compliance embedded into backlog prioritization, not treated as a one-time project.
- Every technical control, policy, process, and audit artifact in place and maintained as living documentation.
- Deployment and change management processes now standard practice across the entire department.

PCI DSS Level 1 · Azure WAF · Key Vault · Always Encrypted · DUKPT · TLS Hardening · Change Management · VPN/VLAN · Audit Evidence · Policy Design

Initiative — MySQL → Azure SQL Server Migration

Live production migration · Zero downtime

A critical infrastructure initiative to migrate the entire production database — supporting ~13,000 daily transactions — from MySQL to Azure SQL Server PaaS, driven by security hardening requirements, PCI DSS Level 1 compliance, and the need for enterprise-grade redundancy.

Context & Challenge

- MySQL was the original database engine — not aligned with PCI Level 1 security controls or Azure-native redundancy capabilities.
- Migration had to happen on a live, transactional payment system with zero tolerance for downtime or data loss.
- Schema and syntax differences between MySQL and SQL Server required careful query and stored procedure adaptation.
- All connections, secrets, and access credentials had to remain secured in Azure Key Vault throughout the transition.

Approach

- Used Azure Database Migration Service (DMS) to orchestrate the migration, enabling continuous data sync between MySQL source and Azure SQL Server target during the cutover window.
- Performed schema conversion and query adaptation for SQL Server compatibility prior to cutover.
- Validated data integrity at every stage using automated comparison scripts before promoting the new database to production.
- Coordinated the cutover during a planned low-traffic window with rollback procedures in place — completed with zero downtime and no data loss.
- Post-migration: configured Azure SQL High Availability with geo-replication for business continuity.

Outcome

- Zero downtime — production system remained fully operational throughout the migration on a platform processing ~13,000 transactions/day.
- Azure SQL HA with geo-replication enabled — full redundancy and failover capability.
- PCI DSS Level 1 compliance satisfied for the database layer.
- Reduced operational costs by consolidating onto Azure SQL PaaS, eliminating self-managed MySQL infrastructure.

Azure DMS · Azure SQL Server · MySQL · PCI Level 1 · High Availability · Geo-Replication · Zero Downtime · Azure Key Vault

Maxar & Pulte DevOps Delivery — Lead Coordination

Gorilla Logic · 2019 – 2023 · Lead Scrum Master

Led Scrum teams across two distinct client engagements in complex DevOps environments — first for Maxar Technologies, a geospatial intelligence company, then transitioning to lead **2–3 concurrent Scrum teams for Pulte Group**, supporting the migration of a legacy application to .NET Core.

What I Did

- Led a Scrum team for Maxar Technologies, coordinating a DevOps team working across Java, Python, Ruby, Jenkins, Cloud Foundry, Kubernetes, and Terraform.
- Transitioned to Pulte Group, leading 2–3 concurrent Scrum teams supporting the migration of a legacy application to .NET Core.
- Facilitated quarterly planning, backlog prioritization, velocity tracking, and inter-team dependency resolution.
- Created sprint-level delivery reports for management visibility across features and commitments.
- Active contributor to Gorilla Labs Ideas — internal innovation initiative for new products and process improvements.

Scrum · Kubernetes · Jenkins · Terraform · Java · Python · .NET Core · Cloud Foundry · Jira

Sagent Financial Product — End-to-End Delivery

Fiserv Costa Rica · 2014 – 2019 · Scrum Master · BSA · Senior .NET Engineer

Wore multiple hats across a 5-year engagement building and maintaining a mortgage-related financial product for Sagent. Contributed as engineer, business analyst, and Scrum Master — often simultaneously — overseeing **two 5-person teams (10 people total)** and giving me a full-stack view of product delivery.

What I Did

- Oversaw two 5-person teams (10 total) building a new mortgage-related financial product using .NET MVC, .NET Core, and Scrum.
- As BSA: wrote User Stories and Acceptance Criteria, gathered requirements from Product Owners, and organized documentation across releases.
- As Scrum Master: facilitated ceremonies, removed blockers, coached teams, and produced delivery reports for management.
- As Senior .NET Engineer: supported platform development and handled classified tickets across financial services clients.

.NET Core · MVC · Scrum · BSA · Financial Services · SQL Server · Version One

Banking Integrations & SharePoint Platform

Equifax Costa Rica S.A. · 2012 – 2013 · Application Developer

- Developed Web Services integrations for banking requirements and analyzed client-specific business rules.
- Built custom SharePoint 2010 applications including Webparts, custom lists, and document libraries.
- Developed Windows applications integrated with Siebel OnDemand.
- Stack: C#, VB.NET, ASP.NET, JavaScript, jQuery, Silverlight 5, SQL Server 2008.

C# · ASP.NET · SharePoint 2010 · SQL Server · Web Services · Silverlight 5

Transaction System — Full Ownership

Unlimited Teaching S.A. · 2010 – 2012 · Development Supervisor

- Gathered requirements and built a Transaction System from scratch using ASP.NET (VS 2010, Framework 3.5/4.0) and SQL Server 2008.
- Managed all server administration under Windows Server 2008 and handled third-party API integrations.
- Built a Silverlight back-office application and handled SSL certificate documentation and installation.
- Provided direct customer service to key clients and coordinated with upper management on system upgrades.

ASP.NET · SQL Server · Windows Server · Silverlight · API Integration