TECNOLOGIA SETÚBAL

POLITÉCNICO SETÚBAL
SCHOOL · POLYTECHNIC UNIVERSITY

ANDREI
FLORIN BALCAN

DESIGNING AND DEVELOPING HUMAN-CENTERED INFORMATION SYSTEMS WITH NOMIS

Project work submitted as a partial requirement for obtaining a Master's degree in Software Engineering

JURY

President: Professor Nuno Miguel Vicente de Pina Gonçalves, ESTSetúbal/IPS
Supervisor: Professor José António Moinhos Cordeiro, ESTSetúbal/IPS
Arguer: Professor Patrícia Alexandra Pires Macedo, ESTSetúbal/ IPS

December 2025

Abstract

This project investigates the practical applicability of the NOMIS (Normative Modelling of Information Systems) methodology by designing and developing Vortex Combat, a functional information system for managing training activities and student progression in a Jiu-Jitsu gym. NOMIS places observable human actions at the core of system modelling, and this work evaluates how these concepts translate into concrete architectural, methodological, and implementation decisions. The system was modelled using NOMIS artefacts—such as Human Action Tables and Existential Dependency Diagrams—which provided the conceptual foundation for the software. The development followed a hybrid Design Science Research (DSR) and Scrum approach, enabling iterative refinement through technical Sprint Reviews and conceptual Design Science Research reflections. The resulting application was implemented using Angular 19 with PrimeNG on the front-end and ASP.NET Core 9 with Entity Framework Core on the back-end, structured according to Clean Architecture and complemented by design patterns including Repository, Strategy, and Specification. These patterns offered a direct and explicit mapping between NOMIS concepts—particularly environmental states and human observable actions—and executable code. The project demonstrates that NOMIS can effectively guide the design and development of human-centered information systems, providing conceptual clarity and improving architectural coherence. It also identifies challenges and opportunities for refinement, particularly in the formalization of modelling artefacts and the generalization of the methodology to other domains. The results confirm NOMIS as a viable methodological foundation and establish Vortex Combat as the first complete system built fully from NOMIS principles, offering insights for further methodology evolution and future practical applications.

Keywords: NOMIS, Design Science Research, Scrum, Clean Architecture, Human-Centered Systems.

Content

Abstract ii

Content iii

List of Figures vi

List of Tables vii

List of Acronyms viii

Chapter 1: Introduction 1

1.1. Background 2

1.2. Motivation 3

1.3. Problem Definition 3

1.4. Work Goals 3

1.4.1. Specific Objectives 4

1.4.2. Research Questions 4

1.5. Document Structure 4

Chapter 2: Theoretical Review 5

2.1. NOMIS – Theoretical Foundations 6

2.1.1. Philosophical Stance: Human Relativism 6

2.1.2. Organizational Semiotics 6

2.1.3. Theory of Organized Activity 9

2.1.4. Enterprise Ontology 10

2.1.5. Theories Comparison 12

2.2. NOMIS Vision 14

2.2.1. Interaction View 15

2.2.2. State View 15

2.2.3. Physical View 15

2.2.4. Information View 15

2.2.5. Norms and Information Fields 16

2.3. Related Methodologies 16

2.3.1. Soft and Hard Approaches to Information Systems Development 16

2.3.2. Socio-Technical Systems 17

2.4. Critical Synthesis and Outlook 18

Chapter 3: System Design 20

3.1. Methodology 21
3.1.1. Design Science Research (DSR) 21
3.1.2. SCRUM 22
3.1.3. DSR with Scrum Combination 22
3.2. NOMIS Representation 23
3.2.1. Integrated Modelling Perspective 23
3.2.2. NOMIS Modelling Artifacts 24
3.2.3. Artefacts Used in the Project 26
3.3. Architecture 26
3.3.1. Conceptual Architecture 26
3.3.2. Application Architecture 28
3.4. Summary 31
Chapter 4: Solution 33
4.1. Case Context 34
4.2. Case Description 34
4.3. System Overview and Structure 35
4.4. Continuity with Previous NOMIS Prototypes 35
4.5. Technological Overview 36
4.6. Applying NOMIS Concepts 37
4.7. Summary 38
Chapter 5: Implementation 39
5.1. Methodology in Practice 40
5.2. Architecture 40
5.3. Data Model 41
5.4. Sprints 43
5.5. Sprint Evolution 46
5.5.1. Sprint 1 46
5.5.2. Sprint 2 46
5.5.3. Sprint 3 47
5.5.4. Sprint 4 48
5.5.5. Sprint 5 49
5.5.6. Sprint 6 49
5.5.7. Sprint 7 50
5.5.8. Sprint 8 51

5.5.9. Sprint 9 51
5.6. Integration and Consolidation 52
5.6.1. User Flows 53
5.6.2. Architectural Coherence 55
5.6.3. Scope 55
5.7. Summary 56
Chapter 6: Results 57
6.1. Evaluation 58
6.2. Results and Findings 58
6.2.1. Vortex Combat 58
6.2.2. NOMIS Applicability 65
6.2.3. Methodology 66
6.2.4. Analysis 66
6.3. Lessons Learned 68
6.4. Limitations and Future Work 68
Chapter 7: Conclusion 70
7.1. Summary of the Work 71
7.2. Main Contributions 71
7.3. Validation and Limitations 71
7.4. Future Work 73
7.5. Final Remarks 73
References 74

List of Figures

Figure 2.1 - NOMIS Vision of Human Centered Information System....14
Figure 3.1 - DSR with Scrum....23
Figure 3.2 - NOMIS Elements Metamodel....25
Figure 3.3 - The Interface System Component....27
Figure 3.4 - Nomis Application System....28
Figure 3.5 - Clean Architecture Diagram....29
Figure 3.6 - Repository Pattern Diagram....30
Figure 3.7 - Specification Pattern Diagram....31
Figure 3.8 - Strategy Pattern Diagram....31
Figure 5.1 - Initial EDD for workout attendance registering....42
Figure 5.2 - Initial Entity-Relationship Diagram....42
Figure 5.3 - Sprint Chronology....44
Figure 5.4 - Final Entity-Relationship Diagram....52
Figure 5.5 - Final EDD for workout attendance registering....54
Figure 6.1 - Authentication Screen....59
Figure 6.2 - Registration Screen....60
Figure 6.3 - Exercise Creation Screen....61
Figure 6.4 - Exercise List Screen....61
Figure 6.5 - Workout Scheduling Screen....62
Figure 6.6 - Workout Enrollment Screen....63
Figure 6.7 - Attendance Registering Screen....64
Figure 6.8 - Progress Screen....65

List of Tables

Table 2.1 - Comparison of OS, TOA, and EO Key Concepts 13
Table 3.1 - NOMIS Diagram modelling artifacts 24
Table 3.2 - NOMIS Table modelling artifacts 25
Table 4.1 - Human Actions Table for Vortex Combat 37
Table 5.1 - Sprints Overview 44

List of Acronyms

ABD	Action Body Diagram
ASD	Action Sequence Diagram
AVD	Action View Diagram
BSD	Body State Diagram
DDD	Domain Driven Design
DEMO	Dynamic Essential Modelling
DSL	Domain Specific Language
DSR	Design Science Research
EDD	Existential Dependencies Diagram
EO	Enterprise Ontology
ETHICS	Effective Technical and Human Implementation of Computer-Based Systems
HADT	Human Actions Dependency Table
HAT	Human Action Table
HID	Human Interaction Diagram
HR	Human Relativism
ICD	Information Connection Diagram
IF	Information Field
IIT	Information Items Table
IS	Information System
ISD	Information System Development
JWT	JSON Web Tokens
LAP	Language Action Perspective
MDSM	Model-Driven Systems Development
NAM	Norm Analysis Method
NOMIS	Normative Modeling of Information Systems
OE	Organizational Entity
OS	Organizational Semiotics
PAM	Problem Articulation Method
POC	Proof of Concept
SAM	Semantic Analysis Method
SSM	Soft Systems Methodology
TOA	Theory of Organized Activity
UML	Unified Modeling Language
VC	Vortex Combat

Chapter 1
Introduction

This chapter establishes the foundational context for this research, which is situated at the intersection of human-centered information systems and practical software engineering. It begins by presenting the background of the NOMIS (Normative Modeling of Information Systems) framework, a methodology that places observable human actions at the core of system design. The discussion then outlines the primary motivation for this work: the recognized gap between NOMIS's robust theoretical potential and its demonstrated practical application. This gap directly informs the central problem definition, which translates to the need for a comprehensive, real-world validation of the framework through the development of a fully functional application. Finally, the chapter concludes by articulating the clear work goals and the structured methodology that will guide this implementation and evaluation effort, setting the stage for the subsequent design and development chapters.

1.1. Background

NOMIS (Normative Modeling of Information Systems) is a human-centered approach to the modeling and development of information systems, introduced by Dr. José Cordeiro in his doctoral study A Normative Approach to Information Systems Modelling [1]. Unlike traditional methodologies that primarily emphasize technical structures and functional specifications, NOMIS places the human being at the center of system analysis and design. It focuses on observable human actions—those that can be objectively identified in real-world interactions—rather than on abstract intentions or subjective interpretations.

NOMIS was conceived to address a persistent challenge in the development of information systems: the difficulty of defining precise and consistent requirements when human and social elements are involved. Traditional approaches often struggle to capture the meaning and implications of human activity in organizational contexts, leading to ambiguity between what is intended and what can be observed. To overcome this, NOMIS adopts the philosophical stance of Human Relativism, which acknowledges that reality is interpreted through the observer's perspective, while maintaining that certain phenomena exist independently of interpretation. This distinction allows the framework to combine human-centered understanding with the objectivity needed for scientific modeling.

From this perspective, NOMIS recognizes information systems as human activity systems, which may or may not involve computer technologies. These systems are inherently complex because they integrate human, organizational, social, and technical dimensions. NOMIS addresses this complexity by distinguishing between observable actions and interactions, which can be modeled and validated empirically, and non-observable elements, such as intentions or beliefs, which remain outside the formal model. This focus on observability provides a bridge between human meaning and technical precision.

Building upon these foundations, NOMIS provides both a conceptual and representational framework for modeling information systems. Its modeling language combines diagrams and tables that depict human actions, actors, physical elements, and information items, all formalized through a metamodel that ensures internal consistency and traceability. This dual nature—conceptual and practical—enables NOMIS to serve as a bridge between theoretical understanding and real-world implementation, a topic explored further in the following chapters.

1.2. Motivation

Despite the theoretical and conceptual strengths of NOMIS, the framework has not yet undergone comprehensive practical validation. Existing prototypes remain partial and exploratory, lacking the completeness and stability required to fully demonstrate its potential. This gap offers an opportunity for a new implementation effort aimed at testing and advancing the framework through practical application.

This project seeks to bridge the gap between NOMIS's theoretical foundations and its practical realization. Its main motivation lies in exploring how the framework's concepts can be transformed into a functioning information system, providing insights into both its strengths and limitations.

In addition to its academic contribution, the project carries practical relevance by allowing the selection and development of a concrete application with real-world value. This approach not only grounds the work in a meaningful context but also demonstrates the adaptability and versatility of NOMIS in addressing diverse information system scenarios.

1.3. Problem Definition

The central problem addressed by this project derives from the lack of comprehensive practical validation of the NOMIS framework. Although NOMIS provides a coherent theoretical foundation, its application in real-world information systems remains largely unexplored. This absence of empirical evidence limits understanding of how NOMIS concepts, such as observable actions, information, and environmental states can be operationalized in a functioning system. It also raises questions about the type of architecture and methodological process required to support such implementation.

Accordingly, this project investigates these open issues through the practical construction of an information system based on NOMIS, providing evidence of its applicability and potential refinement in practice.

1.4. Work Goals

The main goal of this project is to operate the NOMIS framework by designing, developing, and evaluating a functional information system. The work aims to demonstrate how the framework can be applied in practice, to identify the challenges encountered during implementation, and to generate insights for its further improvement and evolution.

1.4.1. Specific Objectives

To achieve the main purpose, this project is guided by the following objectives:

- **O1** – Develop a fully functional application that demonstrates how NOMIS concepts can be implemented in a real-world scenario.
- **O2** – Define and evolve a software architecture consistent with NOMIS's socio-technical foundations, ensuring consistency between conceptual models, system design and technical implementation.
- **O3** – Apply and document a methodological approach that integrates reflective and iterative development practices, allowing adaptation throughout the process.
- **O4** – Evaluate the practical outcomes of applying NOMIS, identifying strengths, limitations, and lessons learned for future research and development.

1.4.2. Research Questions

To guide the investigation and validate the project's objectives, the following research questions were defined:

- **RQ1** – How does NOMIS influence the design and development of an information system when applied as the main conceptual framework?
- **RQ2** – How can software architecture be defined and adapted to align with NOMIS's socio-technical principles during implementation?
- **RQ3** –What challenges and limitations arise when translating NOMIS's theoretical concepts into practice?

These objectives and research questions provide the basis for the methodological and architectural choices presented in Chapter 3, and for the analysis of results and reflection discussed in later chapters.

1.5. Document Structure

The remainder of this document is structured into five main chapters:

- **Chapter 2** – provides the theoretical foundation with a review of the NOMIS model and related sociotechnical methodologies.
- **Chapter 3** – details the proposed system design, including the chosen methodology, modeling approach and architecture.
- **Chapter 4** – presents the practical solution, documenting the case study, technologies, development plan, and the results obtained from applying the methodology.
- **Chapter 5** – dedicated to the evaluation of the process and the validation of NOMIS within the case study.
- Finally, **Chapter 6** presents the final system and its achievements, and **Chapter 7** concludes the document by summarizing the findings and presenting final considerations.

Chapter 2

Theoretical Review

This chapter establishes the theoretical foundation for this study. It begins by introducing the NOMIS methodology, detailing its core principles and the integrated theoretical vision that guides it. The chapter then situates NOMIS within the broader landscape of information systems development by exploring and critiquing established soft and sociotechnical approaches. This comparative analysis highlights the distinct position of NOMIS and frames a final critical synthesis that evaluates its overall contributions and considerations.

2.1. NOMIS – Theoretical Foundations

The NOMIS framework is grounded on a combination of philosophical and theoretical foundations that define its human-centered perspective. This section introduces these foundations, beginning with its philosophical stance—Human Relativism—and followed by three complementary theories: Organizational Semiotics, the Theory of Organized Activity, and Enterprise Ontology. Each of these theories contributes distinct, yet compatible perspectives that together establish the conceptual basis of NOMIS. The section concludes with a comparative synthesis linking these theories to the views that compose the NOMIS Vision presented in Section 2.2.

2.1.1. Philosophical Stance: Human Relativism

The philosophical foundation of NOMIS lies in Human Relativism [2], which acknowledges human-centeredness in all human activities while recognizing that objective reality is always relative to human perception. In this view, reality consists of what can be perceived through the senses, independent of individual interpretation. Human Relativism distinguishes between perception and interpretation: only information involves interpretation, while other elements of human reality—such as actions and physical objects—are simply perceived. It also introduces the notion of observability, referring to what a human being can perceive or acquire through the senses, excluding the interpretative process itself.

From this perspective, only the observable part of human behavior can be treated objectively, forming the basis for rigorous description and technical development. NOMIS adopts this stance by taking observable human actions as the fundamental elements of an information system. These actions represent the driving force of organizational activity, while information systems, in turn, serve to support and extend human action by creating the necessary artifacts and representations that connect human reality with information technology.

2.1.2. Organizational Semiotics

Organizational Semiotics (OS) applies semiotic theory—the study of signs and meaning—to the analysis and design of organizations and their information systems [3]. It provides a socio-technical framework, connecting the physical, technical, semantic, pragmatic, and social dimensions of information. According to Ronald Stamper's Semiotic Ladder [4], meaning in organizations can be understood through six interrelated levels:

- **Physical world** – the material existence of signs, such as printed marks, sound waves, or electronic signals.
- **Empirics** – the statistical and measurable properties of signs, concerning the accuracy and reliability of data transmission and storage.
- **Syntactics** – the formal rules governing how signs combine into valid structures, including grammar, logic, or programming syntax.

- **Semantics** – the relationship between signs and what they denote, linking information to the concepts and phenomena it represents.
- **Pragmatics** – the effects of signs on their users, focusing on intentions, actions, and practical consequences in context.
- **Social world** – the highest level, where norms, conventions, and cultural values determine how communities interpret and regulate meaning.

These levels distinguish clearly between:

- **Observable phenomena** (physical, empirical, and aspects of the syntactic level), which correspond to what can be directly perceived through human senses.
- **Interpretative phenomena** (semantic level), involving the meanings attributed to signs.
- **Intentional and consequence-oriented aspects** (pragmatic level), involving intentions, motivations, and practical effects on human behavior.
- **Socially regulated structures** (social world), consisting of norms, conventions, and shared cultural expectations.

For NOMIS, this hierarchy is essential. Since Human Relativism restricts modelling to what is observable, NOMIS does not associate observable human actions with the pragmatic level—because pragmatics involve intentionality and interpretation, which are not directly perceivable. Observable actions correspond instead to the physical and empirical levels, while pragmatic and semantic aspects appear indirectly through norms and socially regulated behavior, situated at the social level.

Affordances and Ontological Dependency

Among the most influential concepts in OS are affordances—states of affairs in the environment that enable or constrain certain behaviors. An affordance represents a condition that makes an action possible; for instance, “a completed form” affords “registering an order.”

In NOMIS, affordances inspired the definition of Environmental States—stable, observable conditions that enable specific human actions. Similarly, OS emphasizes ontological dependencies between affordances: certain states cannot exist unless others are present. NOMIS adopts this logic when modelling dependencies between Environmental States, ensuring a grounded connection between actions and the conditions that make them possible.

Norms and Information Fields

Another central element of OS is the relationship between norms and information fields. Norms are formal or informal rules that prescribe, permit, or prohibit behaviors within an organization.

A norm can be represented formally as:

```
whenever <condition>
    if <state>
        then <agent> is <deontic operator> to do <action>
```

This structure captures the logic of responsibility, condition, and action—a pattern that also appears in NOMIS, where environmental states and human actions are related by clear, verifiable conditions.

Information Fields are conceptual communities in which norms operate. They define shared terminology, expectations, and meaning, enabling consistent communication within an organizational context.

NOMIS does **not** explicitly model information fields. Instead, it incorporates their role indirectly by ensuring **semantic clarity through observability**:

- Each action, actor, body, and information item is uniquely identified.
- Each model element corresponds to a perceivable aspect of reality.
- Stakeholders can verify correctness through observation rather than interpretation.

Thus, NOMIS reproduces the coherence function of an information field without requiring its formal representation.

MEASUR Methods

Organizational Semiotics also proposes a family of practical methods—MEASUR (Method for Eliciting, Analyzing and Specifying User Requirements) [5]—to support requirements analysis:

- **PAM (Problem Articulation Method)** – focuses on identifying and clarifying the organizational problem.
- **SAM (Semantic Analysis Method)** – leads to the identification of affordances and ontological dependencies, typically resulting in ontology charts where these relationships are represented.
- **NAM (Norm Analysis Method)** – used to identify and formalize norms that regulate agents' behavior.

Although these methods are not applied within NOMIS, they remain relevant as part of the broader theoretical landscape of OS: they illustrate how organizations can identify their affordances, clarify the semantics of their concepts, and formalize the norms that regulate behavior. Their presence highlights the comprehensive nature of OS as a socio-technical framework and situates NOMIS within a wider methodological context, even if NOMIS adopts a distinct and observability-driven approach.

2.1.3. Theory of Organized Activity

The Theory of Organized Activity (TOA), proposed by Anatol W. Holt [6], provides a foundational view of organizations as systems of coordinated human actions. It conceives every organization as an Organized Activity (OA), a universal human phenomenon comparable to language—wherever people collaborate, organized activity exists. Holt defines it as:

"Organized activity exists wherever and whenever people exist. It will be found in social groups of dozens or millions—in the jungle or in New York City, in every culture and at every stage of technological history. It is manifest in every form of enterprise, whether catching game, coping with a fire, or running a modern corporation." [6]

This theory provides a human-centered and normative foundation for understanding organizations and, by extension, information systems. TOA is supported by an associated meta-theory—the Theory of Units (TU)—which explains how communities identify and maintain the elements that make their activities intelligible.

Core Elements of TOA

TOA defines three essential dimensions of any organized activity:

- **Actions** – the fundamental units of human effort. Every action involves some change or effect in the material world.
- **Bodies** – the material or physical entities involved in actions; they occupy space, whereas actions extend in time.
- **Units** – actions, objects, or relations that a community recognizes as meaningful according to shared criteria. A unit is what members of a group collectively identify and maintain through experience and use.

According to Holt's formal statements, every action involves at least one body, and every existing body is involved in at least one action. The theatre of an action is the set of bodies it involves, while the life of a body is the set of actions in which it participates. This duality reveals that organizations can be analyzed as interdependent networks of actions and bodies.

Action Performers and Responsibility

A distinctive feature of TOA is its strong emphasis on human agency. Every action is doubly performed: personally, by an individual, and organizationally, by the Organizational Entity (OE) that the person represents (e.g., a department, a company, or a committee). TOA therefore distinguishes two kinds of performers—persons and organizational entities—and links them through the concept of responsibility.

Actions are driven by the interests of their performers: individuals have personal interests, while OEs have organizational interests. Achieving alignment between these interests is the key to

organizational effectiveness. Machines or software systems cannot perform actions because they cannot bear responsibility or possess interests; they can only support human actions.

Information and States

TOA introduces a distinct interpretation of information. Information is always tied to human decision-making and exists only when it influences subsequent actions. It is carried by bodies in discrete lumps, whose content depends on the context and on the actors who interpret them.

In TOA, states describe the condition of bodies within specific domains of action. This concept differs from the purely technical notion of “system state”: a TOA state is meaningful only in the context of the human activities that define it.

Relation to NOMIS

The Theory of Organized Activity has a direct and foundational influence on NOMIS. Its central assertion—that organizations can be understood as networks of human actions involving material elements—served as the main inspiration for adopting human observable actions as the core modelling element in NOMIS. The TOA distinction between actions (which unfold in time) and bodies (which occupy space), as well as the idea that every action involves one or more bodies, is reflected in the way NOMIS models the physical dimension of an information system.

NOMIS adapts these concepts by retaining only their observable components, in accordance with its Human Relativism stance. While TOA acknowledges that actions are motivated by intentions, interests or decision-making processes, such internal aspects are not observable and therefore are not included in NOMIS models. Instead, NOMIS builds upon TOA’s structural clarity—actions, bodies, and the environmental conditions in which they occur—to provide a precise representation of human activity that can be directly verified through perception.

2.1.4. Enterprise Ontology

Enterprise Ontology (EO) [7] offers a formal and rigorous perspective for understanding organizations as networks of communicative and production acts. Developed within the Dynamic Essential Modelling (DEMO) methodology, EO is grounded on a small set of ontological axioms—the Operation, Transaction, Composition, and Distinction axioms—and the Organization Theorem. Together, these principles define an organization as a coherent structure of actor roles engaging in transactions that create new facts and commitments within the enterprise.

At the core of EO is the notion of the transaction, a standardized pattern of coordination between two actor roles: the initiator and the executor. A transaction normally consists of three stages—request, promise, execute, followed by state and accept—representing how actors negotiate commitments, perform actions, and reach shared agreement about the resulting *state of affairs*. This focus on

commitments rather than behavior highlights the communicative dimension of organizational work and the way authority, responsibility, and accountability are distributed within an enterprise.

EO also distinguishes clearly between different types of acts:

- **Communicative acts** (performative and informative) – establish commitments and shared understanding.
- **Production acts** – change the material world and create new facts.

This separation enables a precise modelling of organizational dynamics: communicative acts create the conditions for action, while production acts realize the changes negotiated between actors. EO thus provides a formal ontology for representing the essential structure of organizational cooperation, independent of implementation or technology.

A further contribution of EO is the modelling of actor roles—abstract responsibilities that can be fulfilled by different individuals over time. Actor roles define what kinds of commitments can be made or accepted, thereby giving a clear structure to organizational authority. This abstraction supports a stable representation of an enterprise even when personnel, procedures, or technologies evolve.

Relation to NOMIS

Enterprise Ontology has a direct influence on the Interaction View of NOMIS. The EO transaction pattern is interpreted in NOMIS as one possible interaction pattern among human actors, and NOMIS retains the fundamental insight that communication is central to coordination in organizations. However, NOMIS extends this perspective in several important ways.

First, whereas EO focuses exclusively on communicative acts as the drivers of organizational coordination, NOMIS models all forms of human interaction, communicative or otherwise, as long as they are observable. Human interactions in real organizational contexts often involve actions that do not fit into the strict structure of a transaction (e.g., informal exchanges, unilateral actions, observation-based interactions, or non-verbal acts). NOMIS accommodates this wider spectrum by treating interaction patterns as reusable structures of observable human actions, not limited to the EO transaction cycle.

Second, NOMIS is not restricted to bilateral interactions between two actor roles. Although EO's transaction pattern structure involves at least two actors, NOMIS allows interaction patterns with two or more participants, including complex multi-actor coordination situations. EO transactions are therefore seen in NOMIS as a special case of a broader class of interaction patterns.

Third, NOMIS removes the dependency on internal commitments or abstract states of agreement that exist in EO. While EO makes commitments explicit within the transaction structure, NOMIS models

only the observable manifestations of communication—language actions and other interaction acts—consistent with its Human Relativism stance.

2.1.5. Theories Comparison

The development of NOMIS is grounded in a detailed analysis of the foundational socio-technical theories introduced in the previous sections: **Organizational Semiotics (OS)**, **Theory of Organized Activity (TOA)**, and **Enterprise Ontology (EO)**.

Although each theory originates from a distinct disciplinary background, they share a common goal—to understand information systems as networks of human actions and communications occurring within a social and organizational context.

Across these theories, human action and responsibility are central. Each perspective examines these notions differently:

- **OS** – emphasizes the norms and affordances that establish the conditions under which actions occur.
- **TOA** – focuses on the interests of human performers and the sequences of their actions, introducing the concept of units as socially recognized elements of activity.
- **EO** – highlights authority and commitments derived from communicative acts, showing how organizational roles coordinate through transactions.

All three acknowledge the importance of context. For **TOA**, context is defined by the activity itself; **OS** expresses it through information fields; **EO** situates it within an organizational or system context.

Each theory treats meaning and information as context-dependent: **OS** views meaning as socially constructed through signs; **TOA** connects information with human decision-making; **EO** derives it from communicative exchanges between actors.

The role of technology is also aligned across these perspectives. In **TOA**, technology indirectly supports human actions; in **OS**, it enables norms and affordances; in **EO**, it facilitates communication and coordination. In every case, technology is subordinate to human activity—a means to support and extend, not replace, human action.

The complementarity of these theories provides a holistic foundation for information systems development. Together, they inspired NOMIS by combining three essential dimensions:

- From **OS** – the normative and contextual layer that defines meaning and constraint.
- From **TOA** – the behavioral and structural layer that captures human activity and material interaction.

- From **EO** – the communicative and organizational layer that governs coordination and responsibility.

This integrated perspective forms the conceptual bridge on which NOMIS was built—a framework that unifies these viewpoints through the principle of observable human action as the common denominator linking norms, actions, and communication.

Table 2.1 summarizes the key concepts of **OS**, **TOA**, and **EO**, highlighting their complementarities and how they collectively inform the development of NOMIS.

Table 2.1 - Comparison of OS, TOA, and EO Key Concepts

Aspect	**Organizational Semiotics**	**Theory of Organized Activity**	**Enterprise Ontology**	**NOMIS**
Core Elements	Signs, norms and affordances	Human actions, units and activities	Communicative actions and transactions	Observable human actions, environmental states, and information items
Context	Information fields	Activities and their contexts	Organizational or system Context	Normative context connecting actions, states, and information
Meaning and Information	Socially constructed through signs	Used for decision-making, mediated by tools	Derived from communicative acts	Represented through observable phenomena linked to actions
Technology	Supports activities through norms and affordances	Supports human actions indirectly through computers	Facilitates communicative actions	Implements and extends human actions through technical artefacts
Human Action	Central to norms and affordances	Core of activities and decision making	Central to communicative transactions	Fundamental unit of analysis; basis of modelling and implementation
Responsibility and Authority	Emphasizes responsibility within norms	Focuses on interests and actions	Highlights authority and commitments	Centers accountability on human agents; systems support but do not act

These theories form a consistent intellectual foundation for NOMIS. **OS** explains the conditions and norms that make human actions possible, **TOA** defines how those actions and materials interact, and **EO** shows how commitments and responsibilities link actors through communication. NOMIS integrates these insights into a single modelling framework that treats information systems as networks of observable human actions governed by norms and enabled by technology.

2.2. NOMIS Vision

The NOMIS Vision of Information Systems [8] builds on the integration of core concepts from the **TOA**, **EO**, and **OS**, centering around the concept of human observable action as the driving force within information systems. Each of these foundational theories provides a distinct perspective on human actions and their contexts, which are central to NOMIS's approach.

Figure 2.1 presents the NOMIS Vision [1], showcasing the central element of observable human actions and the four distinct views: **Interaction**, **State**, **Physical**, and **Information**. These views are interconnected through norms and information fields, which regulate human actions and ensure consistency across the system.

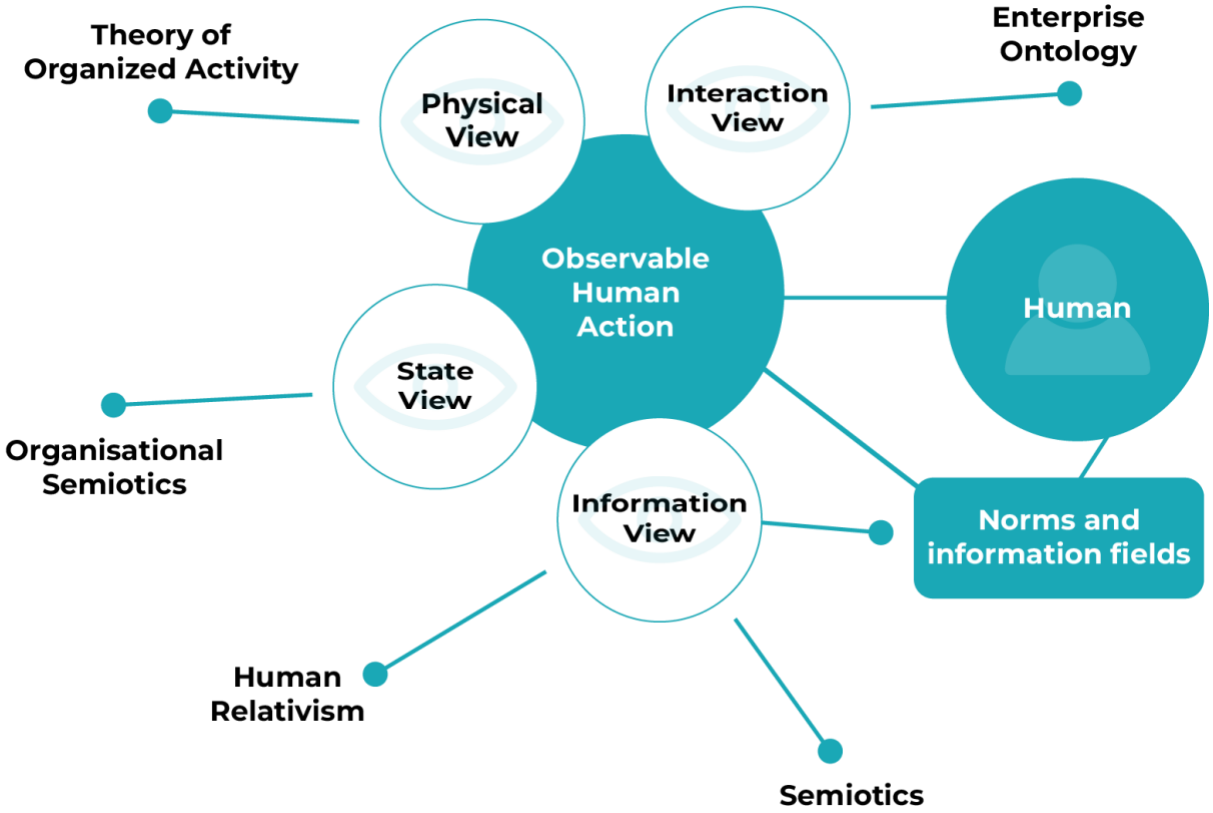

Figure 2.1 - NOMIS Vision of Human Centered Information System

2.2.1. Interaction View

The Interaction View in NOMIS focuses on the communicational and interactional dimensions of human activity. While it draws inspiration from Enterprise Ontology (EO), which models organizations through communicative transactions, NOMIS extends this perspective by allowing a broader range of interaction patterns. Instead of relying on a single transaction structure, NOMIS can represent different forms of observable interactions between actors, including language actions and other types of coordinated behavior.

This view identifies who interacts, how interactions occur, and through which channels or media they take place. By generalizing interactions beyond formal transactions, the Interaction View provides a flexible representation of how people coordinate and communicate within an information system, always grounded in observable human actions.

2.2.2. State View

The State View in NOMIS examines the environmental conditions and dependencies that enable and facilitate human actions, closely aligning with the principles of Organizational Semiotics (OS). These conditions are termed Environmental States, similar to the concept of affordances in OS. An Environmental State represents a stable and observable state that enables and supports human actions. By focusing on these states and their dependencies, NOMIS ensures a stable and well-grounded foundation for modelling information systems, emphasizing the contextual factors that underpin and sustain human activities.

2.2.3. Physical View

The Physical View focuses on the material and observable aspects related to human actions, reflecting the principles of the Theory of Organized Activity (TOA). This view examines the relationships between actions and the physical elements involved, such as tools, instruments, and materials. It models business processes in terms of human actions, their sequence, and their physical context, adhering to expected behaviors regulated by norms.

2.2.4. Information View

The Information View integrates insights from Human Relativism (HR) and semiotics. It acknowledges that information, as an essential aspect of human actions, depends on material support and human interpretation. Semiotics, the study of signs and meaning, plays a crucial role here. By understanding how signs are used to represent information, this view identifies and models the information required, produced, and consumed by human actions. The Information View ensures that

the relevant information is provided to support human actions, aligning with the semiotic principles that govern meaning-making processes in organizations.

2.2.5. Norms and Information Fields

Norms, inherited from OS, play a vital role in regulating human actions within NOMIS. They provide a framework for expected behaviors, ensuring that human actions align with organizational goals. Information Fields (IFs) represent systems of norms shared by a community, defining the common terminology and facilitating consistent communication and understanding within the organization.

2.3. Related Methodologies

This chapter reviews established literature on Information Systems Development (ISD) methodologies, with a particular focus on soft and socio-technical approaches. It examines their historical evolution, core principles, and inherent limitations. This analysis establishes the context for the subsequent introduction of the NOMIS paradigm, highlighting the necessity for its development within the field.

2.3.1. Soft and Hard Approaches to Information Systems Development

In the field of Information Systems Development during the late 20th and early 21st centuries, there was a common distinction between soft and hard approaches.

Hard approaches were understood to be technically oriented, often neglecting the inherently human nature of information systems. hard approaches tend to treat people in mechanistic terms, viewing them as processing devices that can undertake tasks that could equally be done by machines. Hard approaches were associated with characteristics such as mechanistic, rational, technical, scientific, predictable, formal, ordered, objective, quantitative, deterministic, realist, positivist, functionalist and more. These terms were frequently used by proponents of soft approaches to criticize hard approaches, highlighting the contrasting qualities of human and organizational context.

In contrast, soft approaches recognize the centrality of human activity in information systems, seeking to understand the complex interplay between technology and its social context. One of the pioneering soft approaches is ETHICS (Effective Technical and Human Implementation of Computer-based Systems), developed by Enid Mumford [9]. ETHICS emphasizes the alignment between technological and social systems within organizations, treating ISD as a change process involving both technical and social elements. It focuses on employee job satisfaction and quality of work life through participative design processes. Key aspects of ETHICS include actively involving stakeholders in the design process to ensure their needs and expectations are met, simultaneously optimizing both social and technical systems, and empowering workers by considering their roles and satisfaction in IS development.

However, some have argued that ETHICS does not fully integrate the social and technical aspects, instead treating them as distinct elements that need to be "aligned". This separation can lead to difficulties in fully optimizing the interplay between the technological and human factors throughout the development process.

Another significant soft approach is the Soft Systems Methodology (SSM), developed by Peter Checkland and Scholes [10]. SSM is grounded in systems theory, particularly the concept of human activity systems. It views organizations as complex, adaptive systems where human activities and interactions play a central role. This methodology emphasizes understanding the rich context of human activities and the purposeful actions of individuals and groups within organizations. By focusing on human activity systems, SSM aims to uncover the underlying patterns and structures that influence organizational behavior and outcomes, enabling more effective and sustainable improvements. SSM is particularly effective in addressing ill structured or ambiguous problems typically encountered in organizations. It uses a seven-stage process to transition from an unstructured problem situation to actionable improvements.

These stages involve understanding the problem from multiple stakeholder perspectives, defining the core issues and systemic nature of the problem, creating conceptual models based on root definitions to explore potential changes, and comparing these models with the real-world situation to identify feasible improvements.

There are multiple other soft approaches, each offering unique insights and perspectives on the development of information systems. These approaches recognize the centrality of human activity and social context within information systems, in contrast to the more technical, positivist paradigms that have historically dominated the field.

The underlying commonality among these soft approaches is their focus on understanding the complex interplay between technology and its social, organizational, and human factors. By emphasizing the alignment of technological and social systems, these approaches aim to optimize both the technical and human elements for effective and sustainable information systems design and implementation.

2.3.2. Socio-Technical Systems

Building on the foundations laid by soft approaches, socio-technical systems theory further integrates the social and technical dimensions of ISD. Central to the socio-technical perspective is the idea that information systems are inherently socio-technical in nature, as they involve the interaction between people, technology, and the wider organizational and social environment.

One such theory is Activity Theory, which uses activities as the basic unit of analysis. According to Activity Theory, an activity is directed towards an objective or motive, known as the “object of the activity”. These activities are carried out by subjects through purposeful, goal-oriented actions that can only be understood in the context of the overall activity. Actions are composed of well-defined operations and routines and are always mediated by tools and artifacts.

Another significant perspective is the Language Action Perspective (LAP) [11], which focuses on how language is used to perform actions and create commitments, an essential element for understanding the communicative aspects of information systems. This approach emphasizes the role of communicative actions in organizational settings, aiming to design systems that facilitate effective communication and coordination.

Organizational Semiotics (OS) is one more important perspective that helps understand how signs and symbols are used within organizations to create, communicate, and interpret information. OS is grounded in a radical subjectivist philosophical stance, proposing that reality is socially constructed by agents through their actions. According to this view, an agent accesses reality only in the present, with the past and future accessible solely through signs. This concept of a sign is borrowed from semiotics, the foundational theory of OS.

In conclusion, socio-technical systems theory provides a comprehensive framework for understanding and designing information systems that integrate both social and technical dimensions. While Activity Theory, the Language Action Perspective, and Organizational Semiotics offer detailed examinations of human activities, communication, and meaning, the broader spectrum of socio-technical theories. These approaches collectively highlight the necessity of a holistic view that addresses the multifaceted nature of information systems, ensuring that both human and technological factors are effectively integrated into the design and implementation processes.

2.4. Critical Synthesis and Outlook

This synthesis of NOMIS's theoretical foundations reveals its distinct position within the socio-technical landscape. By integrating the nuanced understanding of norms from Organizational Semiotics, the action-centric perspective of the Theory of Organized Activity, and the formal transactional structure of Enterprise Ontology, NOMIS offers a comprehensive framework for human-centered design. Its primary strength lies in this cohesive integration, which enables a holistic analysis of information systems that equally prioritize human action, organizational context, and technical implementation.

However, this very comprehensiveness presents a significant consideration: the potential complexity of its application. The methodology demands a high level of expertise in its constituent theories and may require a substantial investment of time to implement fully, potentially limiting its

adoption in fast-paced or resource-constrained environments. Furthermore, the practical validation of NOMIS through extensive real-world case studies remains an area for future development. Ultimately, NOMIS represents a robust theoretical proposition for bridging the gap between social and technical systems, yet its practical efficacy and scalability are key factors that will determine its broader impact and adoption.

Chapter 3

System Design

The development of the system followed an iterative and exploratory process in which modelling, methodology and architecture evolved together. For clarity, however, this chapter describes the work in an analytical order that reflects the conceptual relationship between its components rather than their temporal sequence. It first introduces the methodological approach that guided the development cycles, then presents the NOMIS-based modelling artefacts used to represent the system conceptually and finally discusses the architecture that emerged because of this process.

3.1. Methodology

This section outlines the dual-perspective methodology adopted for this project, which integrates Design Science Research (DSR) as the overarching research paradigm and Scrum as the specific agile framework for development. This synergistic approach is chosen to rigorously construct and evaluate the NOMIS-based application while ensuring practical, iterative progress.

3.1.1. Design Science Research (DSR)

Design Science Research (DSR) is defined as the “research that invents a new purposeful artefact to address a generalized type of problem and evaluates its utility for solving problems of that type” [12]. Its focus is on solutions, which means that research in design science is oriented toward solving problems. In natural and social sciences, the research goal is to “explore, describe, explain, and predict” [13]. DSR aims to generate knowledge of how things can and should be constructed or arranged (i.e., designed), usually by a human agency, to achieve the desired set of goals.

DSR is composed of six core components that allow simple and easy communication of the scope [14].

- **Problem Description** – What is the problem for which a DSR approach must identify possible solutions?
- **Input Knowledge** – What prior knowledge will be used in the DSR project? Descriptive explanatory, predictive or prescriptive?
- **Research Process** – What are the essential activities planned to make the intended contribution?
- **Key Concept** – What are the most important concepts used in research performed in the DSR approach? A clear definition of the key concepts is particularly important to ensure a rigorous execution of the evaluation activities.
- **Solution Description** – What is the solution to the problem being investigated by DSR?
- **Output Knowledge** – What knowledge is produced in the DSR phase?

DSR provides the macro-level framework, allowing for a problem-solving paradigm that seeks to create and evaluate IT artifacts intended to solve identified organizational problems. In this project, the primary artifact is the functional application developed using NOMIS principles. The DSR process is inherently iterative, cycling through phases of building the artifact and evaluating it rigorously. This aligns with the goal to not only develop an application but also reflect upon and evolve the understanding of how NOMIS translates into a functional information system. Each development cycle produces empirical evidence that feeds back into the refinement of both the artifact and the theoretical knowledge base.

3.1.2. SCRUM

Scrum [15] is the most popular framework for implementing Agile Approach (defined as the ability to move quickly and easily). It is a lightweight process which is iterative in nature, used to manage complex software development. It allows the teams to ship software or products at a regular pace, while following a set of roles, responsibilities and meetings that never change.

In contrast to DSR, Scrum provides the micro-level, operational process. This agile framework structures development work into fixed-length iterations, known as Sprints, which typically last between one to four weeks. The objective of each Sprint is to deliver some concrete, potentially shippable increment of the product. However, in this specific project, the standard Scrum model will be adapted. Rather than adhering to a rigid sprint duration, the cycle length will be dynamically adjusted based on the reflective insights and evaluation findings generated by the DSR process. This modification ensures that the development is directly responsive to the research and learning cycle.

3.1.3. DSR with Scrum Combination

The combination of DSR with Scrum is highly effective for this research, as they complement each other by operating at different but interconnected levels:

- **Design Science Research (DSR)** – Guides the **why** and **what**: It provides a rigorous, scientific context. The building activities in DSR are executed through Scrum sprints. The evaluation activities are formally integrated into the sprint reviews. The developed increment is assessed against both functional requirements and the theoretical propositions of NOMIS.
- **Scrum** – Provides the structured mechanism to manage the complexity of the development. It grants a steady, measurable pace and delivers tangible evidence for the DSR evaluation phase.

This integrated approach ensures that the system's architecture, detailed in the following section, is not statically designed upfront. Instead, it emerges and evolves based on the empirical findings and knowledge generated through each DSR-guided Scrum Sprint. This approach also contributes to the evolution of NOMIS models such as environmental states and human actions, which are reconsidered each DSR review cycle, leading to an evolution of EDD diagrams. This ensures the final application is both theoretically grounded while also adapting practical findings.

Figure 3.1 shows a diagram containing the full workflow using the combination of these methodologies.

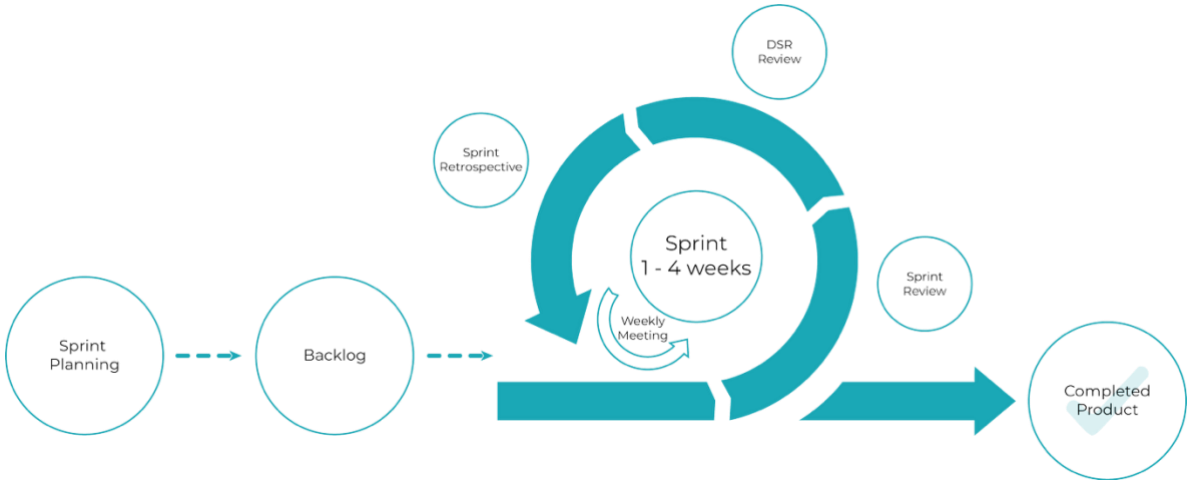

Figure 3.1 - DSR with Scrum

3.2. NOMIS Representation

NOMIS provides a coherent set of modelling artefacts that collectively describe an information system from the perspective of observable human actions [8]. These artefacts—tables, diagrams, and their underlying metamodel—are not independent elements; rather, they work together to represent the Interaction, State, Physical, and Information Views in an integrated and consistent way. This section presents these elements as a unified modelling environment and clarifies which of them were effectively used in the development of the project system.

3.2.1. Integrated Modelling Perspective

At the core of NOMIS is its metamodel (in Figure 3.2), which defines the key concepts of the approach—human actions, actors, bodies, environmental states, and information items—and the relationships that connect them. The metamodel establishes the semantics that make the different diagrams compatible with one another: each diagram expresses a partial perspective (interaction, state, physical, information), but all of them instantiate the same underlying concepts.

This integration ensures that the representations produced by different diagrams can be combined into a consistent and complete system model. The views used in NOMIS are:

- **Interaction View** – how actors interact through observable actions.
- **State View** – the environmental conditions enabling or constraining actions.
- **Physical View** – the material elements involved in actions.
- **Information View** – the information items associated with actions and bodies.

Each view uses specific diagrams and tables, but all are coordinated through the shared concepts defined in the metamodel.

3.2.2. NOMIS Modelling Artifacts

NOMIS employs a set of diagrams and tables that provide structured representations of the elements defined in the metamodel and articulated across the four views of the approach. These artefacts are not independent but complementary, offering different perspectives—interactional, physical, state-based, and informational—over the same underlying concepts.

The diagrams used in NOMIS are summarized in Table 3.1, which identifies each diagram, the view in which it is applied, and its modelling purpose. Similarly, the tables used in NOMIS are presented in Table 3.2, detailing how human actions, dependencies, and information items are organized and recorded. Together, these artefacts support a coherent modelling process: diagrams capture structural and behavioral aspects visually, while tables consolidate the elements involved and ensure completeness and consistency across views.

Table 3.1 - NOMIS Diagram modelling artifacts

Diagram	Content	Usage	Observations
HID (Human Interacion Diagram)	Actors and their interactions	Interaction View	Like a Construction Model used in Enterprise Ontology
ASD (Action Sequence Diagram)	Sequences of actions	Interaction View and Physical View	Like UML Activity Diagrams
BSD (Body State Diagram)	The different states of a body	State View	Like UML State Diagrams
EDD (Existential Dependencies Diagram)	Environmental States and their existential dependencies	State View	Like Ontology Charts as in Organizational Semiotics
AVD (Action View Diagram)	Details each environmental state showing its elements	Physical View	Shows Information Items, Bodies, and Actors related to an action
ABD (Action Body Diagram)	Sequences of actions related to body state changes	Physical View	Like Diplan as used in Theory of Organized Activity
ICD (Information Connection Diagram)	Information Items related to bodies and actors	Information View	Shows information interpreters and information supporting bodies.

Table 3.2 - NOMIS Table modelling artifacts

Table	Content	Usage	Observations
HAT (Human Action Table)	Human actions, actors, bodies and information items	Interaction View and Physical View	Collects human actions, actors, bodies and information items
HADT (Human Actions Dependency Table)	Human actions, their dependencies on bodies, information and context	State View	Collects dependencies for human actions
IIT (Information Items Table)	Information Items	Information View	Collects details of information items

NOMIS elements and their relationships are formalized in a metamodel [8] (shown in Figure 3.2). In this metamodel, besides the key elements previously described, there are also some simple composite elements such as **Activity**, **CompositeInformationItem**, **CompositeBody**, and **CompositeActor** to group similar elements, and a complex composite element, the **EnvironmentalState** grouping different key elements. There also some specializations of elements, such as **Interactions** and **LanguageActions**, for actions **Context**, for bodies and **BodyState** for **States**.

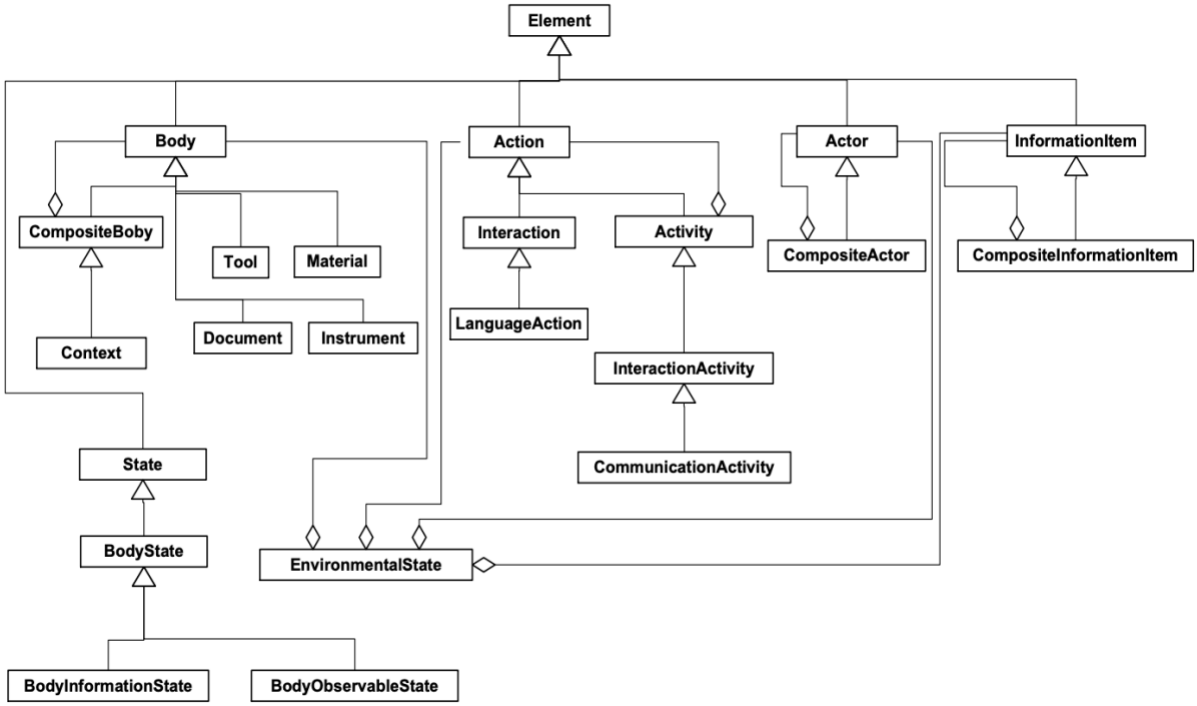

Figure 3.2 - NOMIS Elements Metamodel

3.2.3. Artefacts Used in the Project

Although NOMIS provides a wide set of artefacts, their use depends on the needs and scope of each system. In this project, the following artefacts were used systematically:

- **Human Action Table (HAT)** – served as the primary tool for identifying and categorizing observable human actions within the information system.
- **Existential Dependency Diagram (EDD)** – served as a great first step into identifying and understanding the environmental states and their relationships, revealing the logical sequence and preconditions necessary for actions to occur; multiple versions of EDD were drawn until a final version was achieved.

3.3. Architecture

This section presents the two-layer architectural design that bridges the conceptual foundations of NOMIS with its concrete software implementation.

The first level—the Conceptual Architecture—establishes how NOMIS principles guide the structuring of an information system.

The second level—the Application Architecture—translates these concepts into an operational system built upon modern architectural patterns.

3.3.1. Conceptual Architecture

The conceptual architecture adopted in this project also draws inspiration from the study in [8]. That work proposed a Model-Driven Systems Development (MDSD) strategy, using a Domain-Specific Language (DSL) to represent NOMIS models.

The DSL is defined through the NOMIS metamodel (abstract syntax) and the NOMIS graphical notation [16] (concrete syntax). It also introduces:

- A code structure to be used by a computerized information system.
- A database schema to store business data.
- A rule-based system to store business rules.

The following Figure 3.3 presents the Interface System Component of NOMIS:

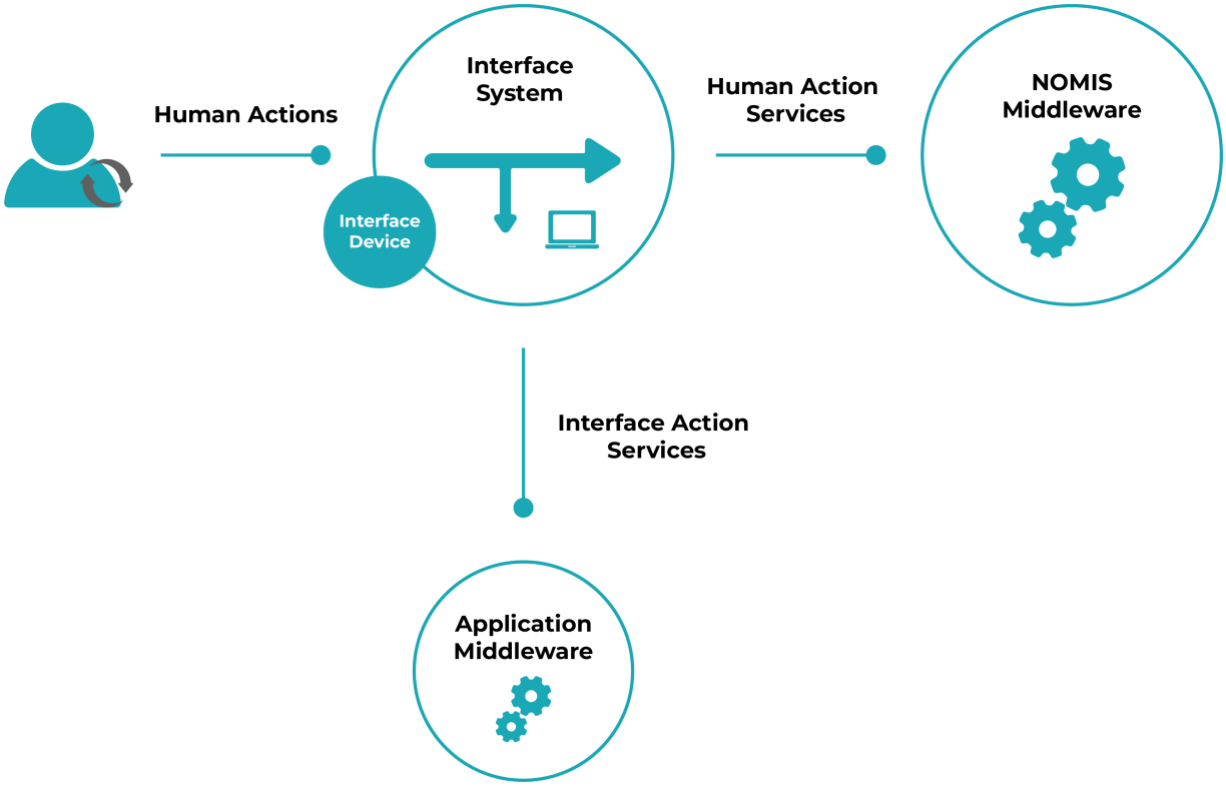

Figure 3.3 - The Interface System Component

System Interface and Middleware

In the conceptual model, human actions are the primary entry points into the system. The Interface System Component receives these human actions and translates them into corresponding services. Some human actions require direct interaction with the core business domain, and these are handled by NOMIS Middleware. Other actions, particularly those related to interface-specific logic or data presentation, are managed separately by the application or interface system.

For example, requesting a list of products would engage NOMIS Middleware to return business data, while sorting that list by price would be handled by the interface system, as it involves presentation logic rather than core business rules.

NOMIS Application System Overview

The Application Middleware exposes system functionalities that are triggered by human actions through the interface. The application middleware handles technical coordination, while NOMIS Middleware manages core business operations and state validation. This layered design ensures a clean separation between the interaction layer (focused on human–system communication) and the core domain layer (focused on organizational rules and state transitions). Such modularity allows the system to evolve with minimal coupling between user interfaces and the NOMIS-based domain model.

As can be observed in Figure 3.4, available functionalities, triggered by human actions, are exposed by the application middleware that includes the Interface System Component. The application

middleware does not deal with any business directly related information, this is the job of NOMIS Middleware, that's accessible through system services.

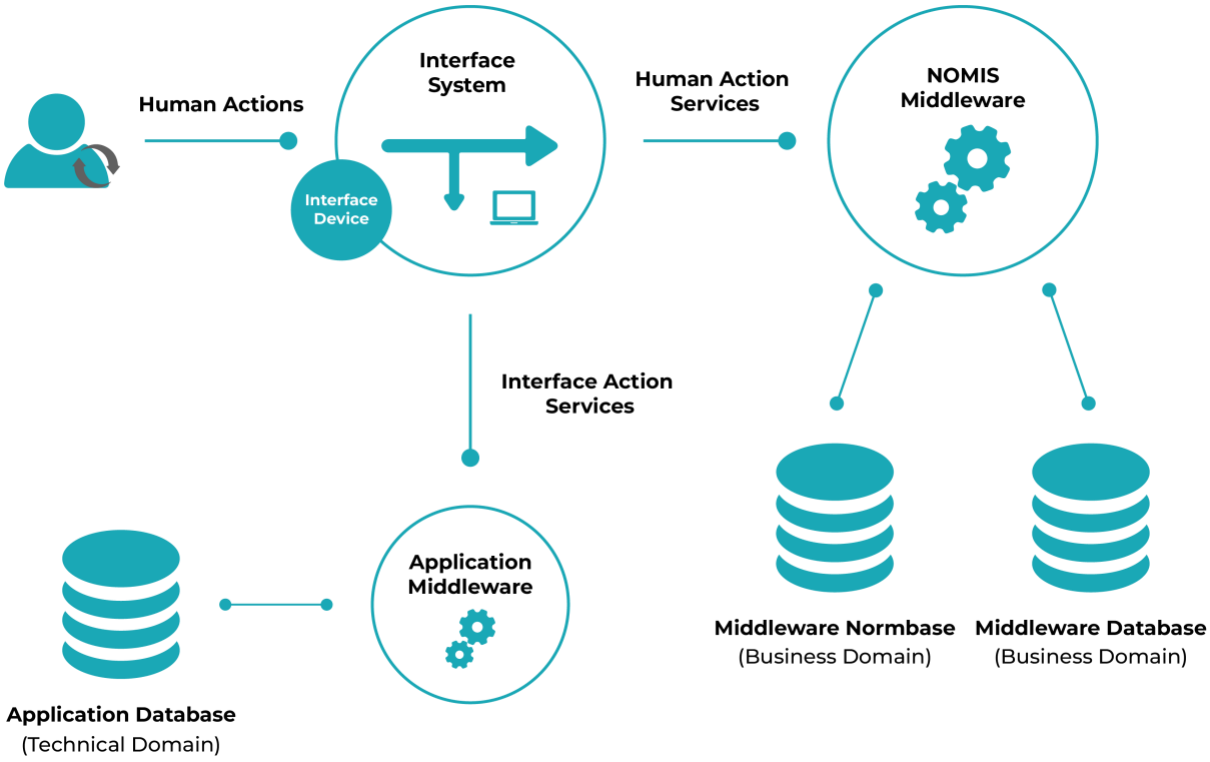

Figure 3.4 - Nomis Application System

3.3.2. Application Architecture

Building upon the conceptual model, the project's Application Architecture was developed and refined throughout the project's Scrum sprints, enhanced by DSR reflection cycle—creating a cohesive hybrid approach for iterative development.

Clean Architecture [17] is based on ideas from several existing software architecture approaches, widely discussed in the software engineering literature Although these architectures differ in their specific implementations, they share the same main goal: separating concerns within a software system. This is usually achieved by organizing the system into layers. Typically, one layer contains the core business logic, while other layers handle user interaction and communication with external systems.

As a result, systems designed using these principles are independent of frameworks, user interfaces, databases, and other external components. This separation also improves testability, since the business logic does not depend on technical details. A key principle of Clean Architecture is the dependency rule, which states that dependencies in the source code must always point inward, toward the core business logic [17].

The choice of Clean Architecture was a natural fit for this project, as its principles align closely with Domain-Driven Design (DDD). This connection is relevant given that the earlier proposal for a NOMIS DSL essentially follows a DDD approach—first modeling the domain concepts and their relationships, then building the application around this core model. This architectural style ensures the system remains centered on the business domain, directly supporting NOMIS's focus on human actions and organizational context.

Clean Architecture organizes code into concentric layers, each with a single, clear responsibility. The core principle is that dependencies point inward: inner layers contain the most fundamental business rules and have no knowledge of outer layers, which handle technical details like databases and interfaces. This inward dependency creates a stable core that remains unaffected by changes in external frameworks or delivery mechanisms, resulting in systems that are easier to test, maintain, and evolve over time. Figure 3.5 illustrates the separation of Clean Architecture into multiple layers.

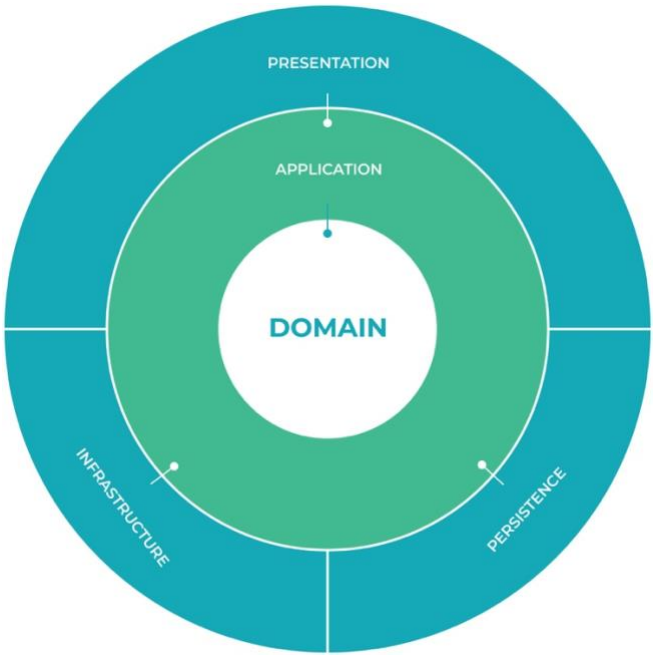

Figure 3.5 - Clean Architecture Diagram

Clean Architecture Alignment

Clean Architecture's concentric structure organizes the system into independent layers:

- **Domain Layer** – encapsulates the core business rules derived from NOMIS concepts, focusing on human actions, environmental states, and normative rules.
- **Application Layer** – coordinates use cases, enforcing the correct sequencing and validation of human actions.
- **Infrastructure Layer** – implements persistence and external services, remaining independent from the domain logic.

- **Presentation Layer** – provides user interfaces that trigger the human actions modelled in NOMIS.

This organization mirrors the NOMIS separation between observable actions and supporting conditions, ensuring that the code structure preserves conceptual integrity.

Design Patterns Supporting NOMIS Concepts

The transition from NOMIS models to functional code required more than just conventional programming. The methodology's focus on human actions and environmental states demanded specific architectural solutions that could preserve these concepts throughout the implementation. The chosen design patterns served as crucial translation mechanisms between the conceptual models and the final application.

Repository Pattern: Abstracts data persistence and isolates the domain model from database concerns. Repositories present a simple model for obtaining persistent objects and managing their lifecycle. They are also responsible for communicating design decisions about object access, and allow an easy substitution of a dummy implementation, for use in testing (typically using an in-memory collection) [18]. This reflects NOMIS's distinction between core human actions and their technical realization. The Repository Pattern will abstract data access, ensuring that the domain logic remains agnostic to the underlying persistence technology, which aligns with the NOMIS principle of separating core human actions from technical implementation details. Figure 3.6 illustrates a generic use of a Repository Pattern.

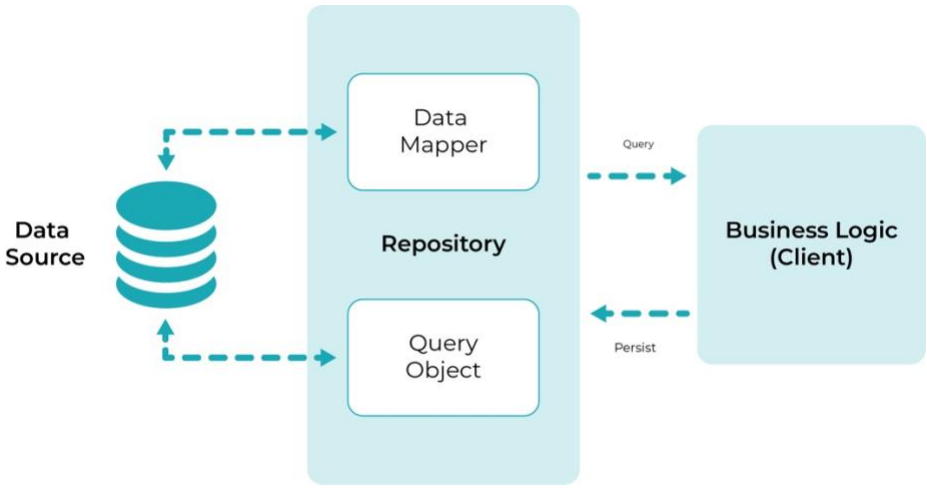

Figure 3.6 - Repository Pattern Diagram

Specification Pattern: A specification states a constraint on the state of another object, which may or may not be present. It has multiple uses, but one that conveys the most basic concept is that a specification can test any object to see if it satisfies the specified criteria. The specification keeps the rule in the domain layer. Because the rule is a full-fledged object, the design can be a more explicit reflection of the model [18]. It encapsulates business rules and environmental conditions that in NOMIS

define the stable, observable conditions that enable human actions. The Specification Pattern allows these states to be expressed declaratively and verified programmatically, ensuring that preconditions for each action are satisfied before execution. Figure 3.7 illustrates a simple implementation of this pattern.

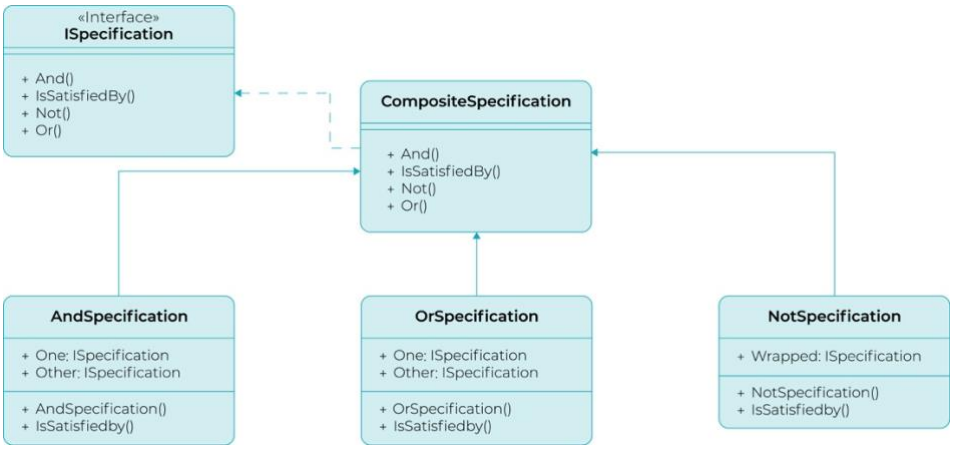

Figure 3.7 - Specification Pattern Diagram

Strategy Pattern: This pattern defines a family of algorithms, encapsulate each one, and make them interchangeable. It defines classes that encapsulate different line-breaking algorithms. An algorithm that's encapsulated in this way is called a strategy [19]. Since NOMIS accommodates multiple interaction patterns. It allows each algorithm to vary independently while sharing a consistent interface. This approach captures the flexibility of human action while maintaining architectural coherence. Figure 3.8 presents a generic implementation of this pattern.

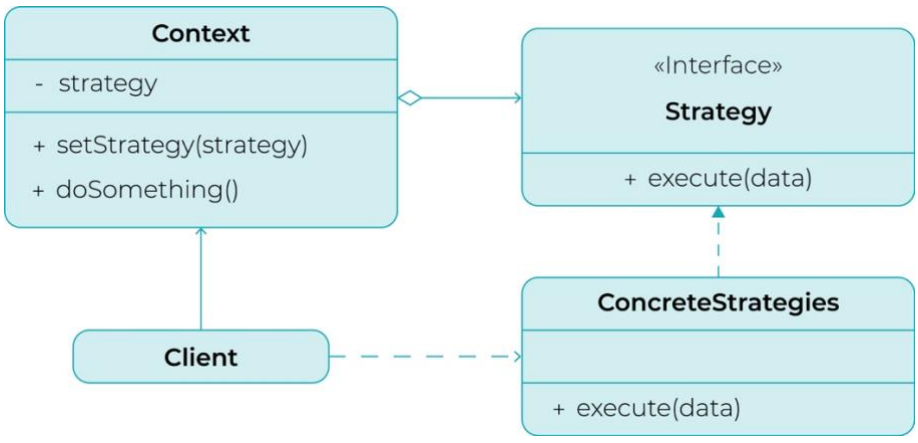

Figure 3.8 - Strategy Pattern Diagram

3.4. Summary

The combined use of Design Science Research and Scrum supports both structured inquiry and iterative refinement, enabling the project to evolve alongside its understanding of NOMIS implementation.

The Clean Architecture structure and associated design patterns provide a concrete and maintainable translation of NOMIS concepts into executable software, ensuring conceptual alignment between model, code, and human activity.

Chapter 4

Solution

This chapter details the practical implementation of the proposed solution, following a structured progression from case selection to technical execution. It begins by establishing the case context and description, followed by the NOMIS modeling representation that connects theory to practice. The chapter then presents the functional requirements derived from this analysis, outlines the technological implementation, and concludes with a preview of the methodological approach guiding the development process.

4.1. Case Context

The case study selected for this project needed to satisfy specific conditions to effectively demonstrate the practical application of the NOMIS approach. The chosen domain had to represent a complete information system involving real users, business data, and clear, observable human actions, allowing NOMIS modelling to be applied consistently across its views.

Among several candidate domains—academic management, e-learning systems, medical scheduling, e-commerce, and social networking—the selected case was one that combined practical relevance, operational simplicity, and close familiarity with its real context: the management processes of a Jiu-Jitsu training gym. This context provides a naturally human-centered environment, where interactions, responsibilities, and routines are well defined and easily observable, making it a suitable scenario for modelling under NOMIS principles.

The **Vortex Combat** gym was therefore identified as an ideal setting in which to design and implement an information system aligned with NOMIS. The choice was further supported by the student personal experience in martial arts, which enabled accurate understanding of the workflow, roles, and implicit rules governing the gym's operations. This familiarity ensured reliable identification of human actions, actors, and environmental states, offering a realistic and sufficiently rich domain to evaluate NOMIS in practice.

4.2. Case Description

The Vortex Combat Gym is a martial arts facility dedicated to Jiu-Jitsu training. Its internal operations involve three main roles: reception staff, the master instructor, and students. The gym includes a reception area for handling general inquiries, a training mat where classes and practice sessions occur, and an office used by the instructor to manage student records.

The master instructor is responsible for teaching classes and monitoring student progression. Progress is currently tracked manually on paper, recording the number of classes attended and the techniques completed at each belt level. These records serve as the basis for determining when a student is eligible for belt promotion.

Students who wish to know their progress—such as how many classes they have attended or how close they are to their next belt level—must ask the instructor directly. Some students are reluctant to do so, as they fear their inquiry may be interpreted as undue pressure for promotion. As a result, students are often unaware of their standing, which may lead to frustration or reduced motivation.

The existing system therefore relies heavily on manual record-keeping and informal communication. Information is not readily accessible to students, updates depend entirely on the

instructor's availability, and maintaining accurate paper records can be time-consuming. This situation highlights several operational challenges that motivate the need for a more structured and accessible information management solution.

4.3. System Overview and Structure

The system to be developed—also named **Vortex Combat**—is a web-based management platform designed to support the daily operations of the Jiu-Jitsu training gym. Its primary goal is to enable instructors and students to record, monitor, and visualize belt-progression data in a transparent and structured way.

At a functional level, the platform offers several key capabilities:

- **Workout Scheduling** – Where the master can create and manage training sessions, including setting dates, assigning exercises, and updating session details when required.
- **Attendance registration and evaluating student performance** – The master can register the students that did in fact participate in a workout that they were enrolled in.
- **Progress visualization** – Both masters and students can consult belt-progression data. Students see only their own progress, while masters can review all students. Information includes attended workouts, completed techniques, and remaining requirements for the next belt level.
- **User and role management** – The master should be able to manage the platform's roles and user accounts.

Although the scope of the platform is intentionally limited—as it serves as a prototype for research purposes—it includes all major components of a functional information system: authentication, persistent data storage, and a web interface. Its structure is sufficient to support realistic workflows within the gym environment and to act as a basis for the subsequent development and evaluation stages.

4.4. Continuity with Previous NOMIS Prototypes

Previous research on NOMIS has explored how the methodology could support the development of information systems, with several prototype applications being proposed over time. Notably, earlier work on a Model-Driven Systems Development (MDSD) approach for NOMIS suggested generating parts of a system directly from NOMIS models through automated transformations. This included deriving system services from human actions, creating interface structures from activities, generating database schemas from information items, implementing state machines based on environmental states, and constructing rule-based components from norms.

While these studies demonstrated the potential of using NOMIS as a basis for system generation, the resulting prototypes remained largely conceptual or only partially implemented. They offered important theoretical contributions and architectural ideas but did not provide a fully developed application that could validate the methodology in practice.

The Vortex Combat system advances this previous work by delivering a complete, domain-specific application built iteratively from NOMIS models. Unlike earlier prototypes, this project moves beyond conceptual demonstration, implementing a functioning system with real data, user interaction, and persistent storage. As such, it represents a significant step forward in demonstrating how NOMIS can be applied throughout the full lifecycle of system design, development, and evaluation, providing practical insights that complement and extend prior explorations.

4.5. Technological Overview

The Vortex Combat application was implemented as a web-based system following modern, service-oriented architecture. The technology stack was chosen to ensure maintainability, scalability, and ease of development, while also leveraging existing experience with the selected tools.

The back-end was developed using **ASP.NET Core 9** with **C#**, adopting the **Clean Architecture** pattern to separate concerns and establish a clear boundary between business logic and technical infrastructure. This structure supports long-term maintainability and facilitates the alignment between conceptual design and implementation.

The front-end was built with **Angular 19**, complemented by **PrimeNG** for component design and **RxJS** for reactive data handling. This combination provides a robust, modular, and responsive user interface suitable for iterative development.

Data persistence is handled by **Entity Framework Core** with a relational database schema. The underlying data structures reflect the core elements of the domain and are mapped to the system's persistent storage in a way that supports consistency, extensibility, and straightforward interaction with the application's domain layer.

Authentication is implemented using **JWT (JSON Web Tokens)**, offering a simple and interoperable mechanism for managing user sessions and access control.

The selection of these technologies was primarily guided by practicality, particularly the student's familiarity with **Angular** and **.NET**. This familiarity helped reduce the learning curve and allowed the project to focus on the main research goal: implementing the intended system with a clear correspondence between the conceptual model and the final application, rather than spending time adapting to unfamiliar tools.

4.6. Applying NOMIS Concepts

The implementation process began by modelling the gym's workflow using the NOMIS representation. This initial phase identified the core human actions, the actors involved, and the environmental states that enable or constrain those actions. Tools such as the Existential Dependency Diagram (EDD) and the Human Action Table (HAT) were particularly useful for structuring the domain and clarifying the relationships between actions and states.

Using the EDD, the main activities of the gym—such as scheduling, enrolling in, attending, and evaluating training sessions—were mapped to the environmental states that make them possible. Examples include "Workout Scheduled", "Student Enrolled", "Attendance Recorded", and "Student Eligible for Promotion". These states provided a stable foundation for identifying rules and dependencies governing the gym's operations.

The HAT complemented this work by listing the observable human actions, their initiators and addressees, and the physical and informational elements involved. Table 4.1 summarizes this set of actions, covering tasks such as creating exercises, scheduling workouts, enrolling students, checking progress, validating techniques, and performing promotions. Each of these actions emerged naturally from the domain and later guided the definition of the system's functional requirements.

This modelling phase ensured that the software's features were aligned with the real practices observed in the gym. Instead of beginning with technical assumptions, the development process was driven by the explicit human actions and states identified in the models. As a result, the system's functionalities—such as workout scheduling, attendance registration, progress consultation, and exercise management—could be traced directly back to the conceptual structures defined during modelling.

Ultimately, the use of NOMIS provided a systematic and structured view of the domain, supporting a clearer understanding of the workflow and informing the system's design. By grounding the solution in observable actions and well-defined states, the implementation remained consistent and coherent as it progressed toward a complete application.

Table 4.1 - Human Actions Table for Vortex Combat

Human Action	**Initiator**	**Addressee**	**Bodies**	**Information**	**Location**
Create exercise	Master	-	Exercise form	Exercise Details	Master dashboard
Schedule workout	Master	-	Calendar, Room	Start and end dates, exercises	Master dashboard

Edit workout	Master		Calendar, Room	Workout details	Master Dashboard
Enroll in workout	Student	-	Calendar	Workout selection	Student Dashboard
Attend workout	Student	-	Room/Training mat	-	Gym
Teach workout	Master	Students	Room/Training mat	Exercises list	Gym
Register attendance	Master	-	Attendance list	Student participation in a workout	Master dashboard
Consult progress	Student/Master	-	-	Belt status, exercises completed, exercises remaining	Student/Master dashboard
Consult exercise list	Student/Master	-	Exercise list	Techniques by belt and degree	Student/Master dashboard
Validate technique	Master	Student	Room/Training mat	Technique information	Gym
Promote to next belt	Master	Student	Room/Training mat	Student meets promotion criteria	Gym

Through this representation, the model clarified both the observable structure of human activity and the rules governing its progression. This conceptual model guided the definition of system features and interactions, ensuring that every software function could be traced back to a NOMIS construct.

4.7. Summary

This chapter presented the Vortex Combat case study as a concrete demonstration of the NOMIS approach. It described the organizational context, the system to be developed, its conceptual grounding in NOMIS, and the technological environment chosen for implementation. By focusing on a realistic domain and mapping it through NOMIS modelling, this project bridges theory and practice, preparing the ground for the next chapter, which details the iterative implementation process and the evolution of the system architecture across development sprints.

Chapter 5
Implementation

This chapter documents how the system was implemented following the NOMIS-based design established in the previous chapter. It describes how the methodology guided development practices, architectural decisions, data modelling, and the iterative construction of the Vortex Combat application.

5.1. Methodology in Practice

The implementation of the Vortex Combat system followed the hybrid Design Science Research (DSR) and Scrum approach defined in Chapter 3. Development was organized into iterative sprints, each concluding with a DSR reflection session to reassess progress, validate assumptions, and adjust the direction of work whenever necessary. This structure ensured that development remained both iterative and research driven.

Work items were managed through Jira, where each task—identified with the prefix “VC”—was associated with sprint backlogs, development branches, and review notes. GitHub was used to host the source code and continuous integration workflows. Sprint duration was flexible rather than fixed, allowing complexity and technical uncertainty to dictate the time required for each cycle.

A key characteristic of the process was the prioritization of back-end use cases and NOMIS-derived validation rules before implementing front-end interfaces. Ensuring the correctness of domain logic early in the process helped maintain conceptual alignment with NOMIS, as environmental states and human actions needed to be properly represented and validated at the domain level before being exposed in the user interface. When tasks were not fully completed within a sprint, they were carried forward with explicit justification recorded in the sprint review.

Overall, the combination of Scrum’s structured iteration and DSR’s reflective component created a development rhythm that supported steady progress while allowing methodological refinement. This dual structure made it possible to handle uncertainty, revisit modelling decisions when needed, and maintain coherence between the evolving system and the research objectives guiding the project.

5.2. Architecture

The implementation of the Vortex Combat system follows the Clean Architecture approach to ensure a clear separation of concerns and to maintain consistency with the domain structure identified during modelling. This architectural choice supports long-term maintainability by isolating the business logic from technical details, making it easier to evolve the system without risking conceptual misalignment.

The architecture is organized into four main layers:

- **Domain** – This layer contains the core entities, value objects, and business invariants derived from the observable human actions and environmental states identified during modelling. It represents the most stable part of the system and is kept independent of any external framework or technology.
- **Application** – This layer orchestrates the system’s use cases. It evaluates the conditions under which actions may take place by invoking Specifications that encode the rules and

constraints reflected in the environmental states. The Application layer coordinates operations, ensuring that each action respects the defined business logic before changes are applied.

- **Infrastructure** – This layer implements technical details, including database access through Entity Framework, repository implementations, and external services. It provides the concrete persistence mechanisms required by the upper layers while remaining replaceable.
- **Presentation** – This layer exposes the system's functionality via REST endpoints. It receives requests from the front-end, triggers Application-level use cases/actions, and returns structured responses. The Presentation layer contains no business logic; its role is limited to protocol handling and data transformation.

Throughout the development process, responsibilities were progressively moved inward—from controllers and infrastructure components into the Application and Domain layers. This refinement improved testability, reduced duplication, and strengthened the semantic alignment between the system's conceptual model and its implementation. The result is an architecture in which the central concepts from modelling—actions, states, and rules—are explicitly and consistently represented in code.

5.3. Data Model

The data model was developed from the case study, the functional requirements, and the NOMIS modelling performed earlier. It served as the foundation for defining the application's scope and identifying the core elements that support each activity. The first step in this process was analyzing the requirement "Register Workout Attendance", a central functionality that involves multiple actors and has several dependencies.

The initial Existential Dependency Diagram (EDD) for this requirement, shown in Figure 5.1, revealed the key domain entities and the relationships between them. It identified the master and student as principal actors, and the state Workout with Workout Attendance as the central condition enabling progress tracking. The EDD also exposed essential dependencies, for example, that attendance registration requires an existing workout and enrolled students, and that workouts must be associated with valid student memberships.

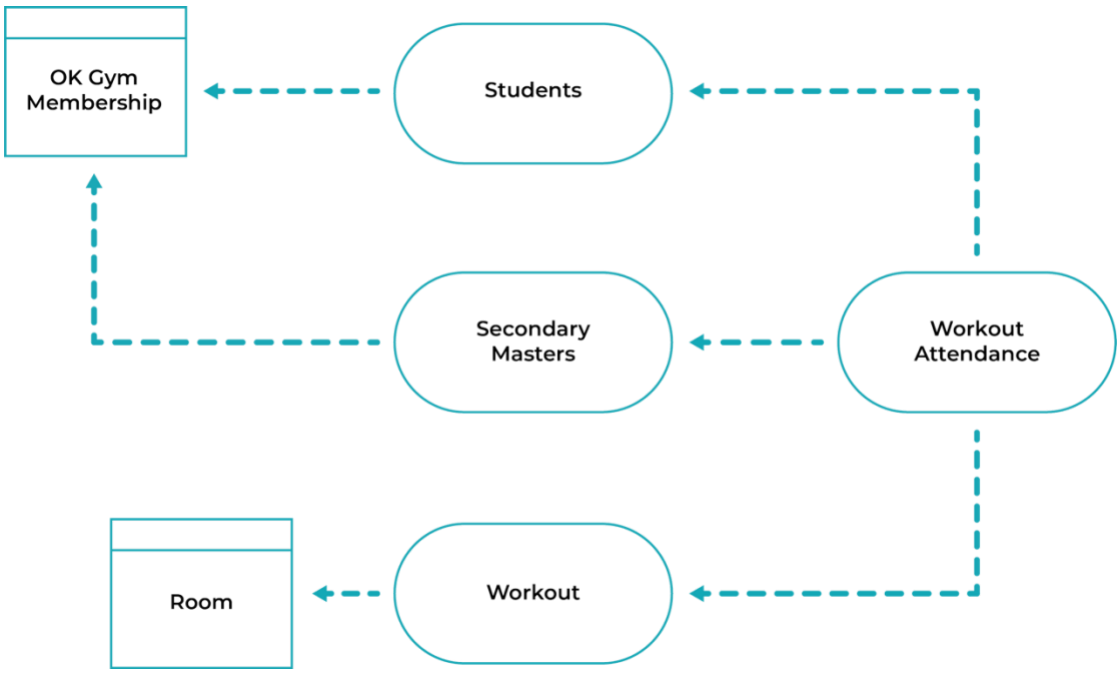

Figure 5.1 - Initial EDD for workout attendance registering

Using these dependencies as a guide, the initial database schema was created with Entity Framework and MySQL. The resulting Entity–Relationship diagram (illustrated in Figure 5.2) reflects the structure derived from the EDD, establishing entities such as *Workouts*, *Students* and *Masters*, along with their required relationships.

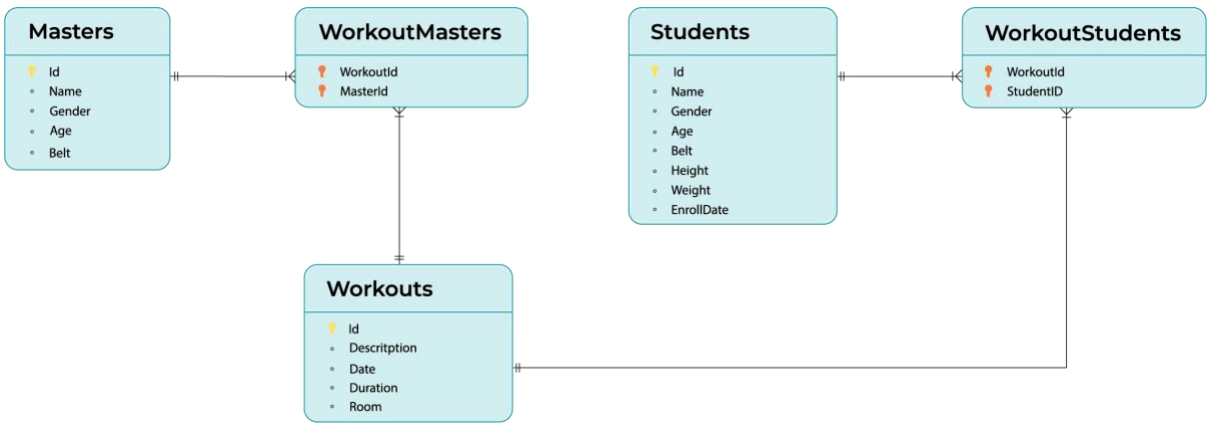

Figure 5.2 - Initial Entity-Relationship Diagram

These early steps were crucial for mitigating development risks and establishing a solid foundation before expanding the application with additional features.

The modeling process revealed several key requirements that needed to be addressed:

- **Workouts** – must support scheduling and editing, with validation rules to prevent double-booking.
- **Exercises/Techniques** – needed to be introduced as a new entity, requiring creation and association with scheduled workouts.

- **Progress Tracking** – should automatically calculate student advancement based on completed exercises in attended workouts.
- **Student Data Model** – required review to identify missing fields necessary for supporting these features, such as the current belt.

Overall, the data model evolved iteratively, guided by both the conceptual structure identified through NOMIS modelling and the practical needs that emerged during implementation. This iterative refinement ensured that the final schema was both consistent with the domain and adequate for supporting the system's core functionalities.

5.4. Sprints

The development of the Vortex Combat system was organized into iterative sprints, each representing a focused cycle of implementation, evaluation, and refinement. This structure aligned with the hybrid DSR with Scrum methodology, ensuring that progress was both incremental and continuously reassessed.

Each sprint contained a set of backlog items—managed through Jira—that defined the specific goals for the iteration. These items included feature development, architectural adjustments, bug fixes, and refinement of NOMIS-related elements such as environmental states and human actions. Every item was linked to development branches and code reviews hosted on GitHub.

Sprint duration was flexible, adapting to the complexity of the tasks involved. More challenging activities, such as designing core domain logic or implementing specifications, naturally required longer cycles, whereas smaller interface adjustments were completed more rapidly. At the end of each sprint, a DSR reflection session was conducted to assess the outcomes, verify alignment with the conceptual model, and determine whether any course corrections were needed.

The following Figure 5.3 presents the project's sprint chronology, highlighting the flexible duration of each iteration.

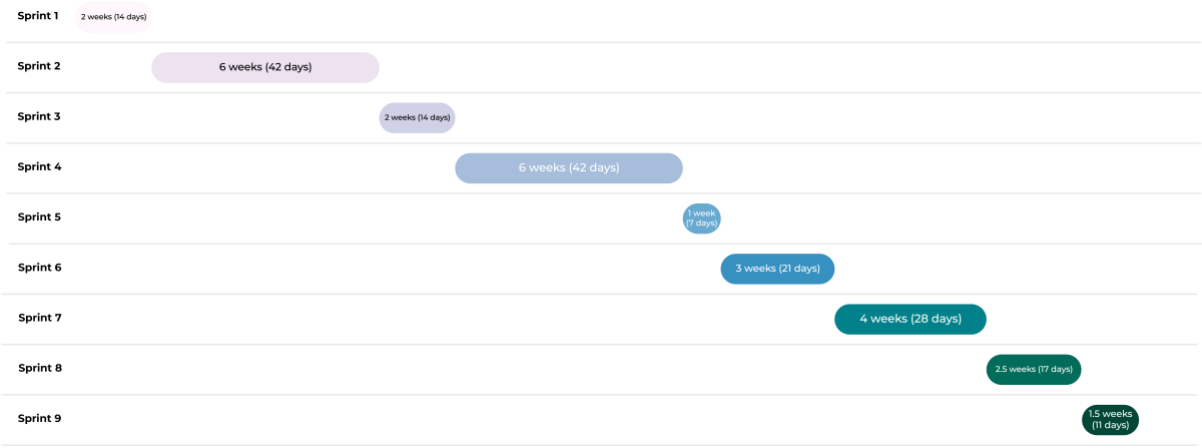

Figure 5.3 - Sprint Chronology

Table 5.1 summarizes the sprints carried out during development, listing the key activities and the progression of work across iterations. These incremental cycles allowed the architecture, data model, and features to evolve in a controlled and conceptually coherent manner, ultimately converging on a stable and functional application.

Table 5.1 - Sprints Overview

Sprint	**Key Decisions & Changes**	**Lessons**
Sprint 1 – Project setup and initial POC.	Established project repository and full-stack integration. Defined core roles and developed an initial feature guided by NOMIS modeling.	Smooth setup accelerated planning; Deferring authentication enabled faster initial progress.
Sprint 2 – Adopt Identity Framework and definition of Functional Requirements.	Integrated JWT authentication and defined the complete set of functional requirements.	User data models required refinement during registration screen development.
Sprint 3 – Implement Functional Requirements.	Improve the existing “Register attendance” feature; Implemented workout scheduling using the schedule-x framework; Added a responsive layout to the front-end.	The initial Proof of Concept provided a solid foundation for further development; The schedule-x framework required significant customization to align with PrimeNG's styling and responsiveness.
Sprint 4 – Workout enrollment.	Enabled student enrollment in scheduled workouts; Enhanced the calendar component to support enrollment actions;	The initial Proof of Concept and NOMIS EDD could be refined effectively once end-to-end testing became possible.

	Refined the Existential Dependency Diagram (EDD) based on new insights.	
Sprint 5 – Workout editing. New toast component. Exercise/technique creation.	Allowed masters to edit workout data (time, rooms, description); Created a reusable toast component for user notifications; Introduced an *AttendanceStatus* enum (None, Enrolled, Attended) to align with NOMIS states. Added a form for masters to create exercises/techniques and associate them with workouts; Implemented a reusable accordion component for students and masters to consult exercises.	Non NOMIS tasks, such as basic data management for exercises, were faster to develop as they are generic and required less analytical overhead.
Sprint 6 – Student progress & back-end refactoring.	Developed an endpoint for automatic student progress calculation per belt; Refactored the back-end to enhance the Clean Architecture structure; Created dedicated SVG assets for each belt and grade on the front-end.	The initial progress calculation was inaccurate due to a lack of exercises defined for junior belts (yellow, green, gray, orange); The exercise dataset was also found to be insufficient for other belts and degrees, highlighting a gap in the core data.
Sprint 7 – Overall architecture refactoring.	Enhanced the back-end architecture by implementing the Repository, Strategy, and Specification patterns. Enhanced the progress calculation logic; Added a dedicated progress page for masters and students;	Specifications provided a direct mapping to NOMIS states, creating a clear link between the models and code; Integrating these patterns required significant, unplanned refactoring of the Clean Architecture and the Identity Framework's user fields.
Sprint 8 – Clean Architecture refactoring.	Refactored the Clean Architecture to align with the newly implemented design patterns; Adapted the Identity Framework to work with the new architecture.	The use of Specifications eliminated code duplication and created a clear, maintainable link to environmental states.
Sprint 9 – Student progress & Front-end refactoring.	Updated front-end DTOs to match the revised API endpoint; Fixed bugs in the schedule-x calendar.	The core application workflow became fully testable, with only minor improvements left for future iterations.

5.5. Sprint Evolution

The technical and conceptual dimensions of the project evolved in parallel throughout the development process. Each sprint contributes not only to the expansion of the system's functional scope but also to the refinement of how NOMIS concepts were interpreted and operationalized in practice. The following subsections present in a structured manner, describing the work implemented in each sprint, the outcomes of the Sprint Review, and the insights that emerged from the corresponding DSR reflection session.

5.5.1. Sprint 1

The first sprint focused on establishing the technical baseline for the project. Its primary objective was to produce a minimal but functional foundation of the application, that would enable subsequent iterations without premature architectural commitments. Basic front-end and back-end integration was achieved, allowing early experimentation with data flow and interaction patterns. This initial setup also supported the creation of the first versions of the EDD (Figure 5.1 in section 5.3) and HAT.

Sprint Review

The Sprint Review highlighted several structural gaps in both the domain model and the initial data entities. The Workout entity lacked essential attributes such as time, room, duration, and description, preventing realistic simulation of user flows. Furthermore, the absence of differentiated user roles (student vs. master) limited interaction with the system. These imperfections informed the decision to integrate the Identity Framework in the next sprint to support role-specific behavior.

DSR Review

The first DSR session revealed that the initial EDD, although valuable as a starting point, lacked the granularity required for consistent NOMIS modelling. Environmental states were not associated with information items, and distinctions between actor types were not yet explicit. These observations clarified that the conceptual model required additional structure and that translating NOMIS artefacts into executable system behavior would demand a clearer modelling discipline. This insight initiated the first architectural discussions regarding how NOMIS concepts should be embodied in code.

5.5.2. Sprint 2

With the technical foundation established in Sprint 1, Sprint 2 introduced formal user management by integrating the ASP.NET Identity Framework. This addition enabled both students and masters to authenticate using different role-specific fields, allowing the system to begin supporting user-dependent actions. At this stage, the complete list of functional requirements was also drafted, providing a structured roadmap for selecting features that were both relevant to the research objectives and feasible within the project timeframe.

Sprint Review

During the Sprint Review, the adoption of “Register Workout Attendance” was the first functional requirement to be implemented according to NOMIS principles. Although login and registration were important to enable authentication and role management, these features rely heavily on ASP.NET Identity’s standard mechanisms and don’t provide meaningful opportunities to model NOMIS-specific concepts. By contrast, “Register Workout Attendance” directly involves interaction between actors, conditional actions, and updates to environmental states, making it well suited to guide the methodological development of the system.

DSR Review

The DSR session reinforced this decision. It was observed that while user management features (such as login and registration) are essential for operating the system, they do not represent distinctive human actions within the NOMIS framework. Their implementation does not meaningfully contribute to exploring NOMIS modelling principles because they follow established patterns and do not require conceptual analysis of actions, dependencies, or environmental states.

In contrast, “Register Workout Attendance” was identified as an action that embodies key NOMIS characteristics: it depends on permissions, is only possible under specific environmental states (such as a workout having been scheduled and a student being enrolled), and may lead to new environmental states being attained. For this reason, the DSR reflection concluded that this requirement would offer a more rigorous and informative starting point for applying NOMIS concepts and validating the methodology in practice.

5.5.3. Sprint 3

The chosen feature, “Register Workout Attendance,” revealed an important dependency: a workout must already exist for its attendance to be registered. This requirement led to the technical decision to adopt the *schedule-x* framework to handle workout scheduling and management. The missing workout fields identified during Sprint 1 were added, and a new responsive UI layout was created to support upcoming functionalities.

Sprint Review

The Sprint Review confirmed that the *schedule-x* framework required significant customization to align it with the existing PrimeNG styling and to achieve the desired responsiveness. Despite these challenges, the initial Proof of Concept demonstrated that the framework could be adapted effectively and established a viable foundation for subsequent development.

DSR Review

As new entities and services related to workouts were introduced, the structure of the codebase became increasingly confused. NOMIS-specific actions were mixed with generic application logic, and

there was no consistent mechanism for translating NOMIS models and artefacts into code. This lack of separation made the conceptual model difficult to trace in the implementation and signaled an architectural limitation. The DSR reflection identified the need for a clearer structural boundary within the system. The initial solution proposed was to adopt a modular Clean Architecture, which would separate domain logic from technical concerns and allow NOMIS-related concepts to be expressed explicitly within the domain layer. This architectural shift directly addressed the “dirty” and increasingly complicated codebase and established the foundation for a more coherent model-to-code alignment in later sprints. To complement this choice, Clean Architecture also aligns closely with Domain-Driven Design (DDD), which was the earlier proposal for a NOMIS DSL.

5.5.4. Sprint 4

With workouts now schedulable, Sprint 4 introduced student enrollment. This addition provided the first complete end-to-end scenario in which NOMIS-related actions could be meaningfully validated. In parallel, the architecture was refactored into a Clean Architecture structure, as decided in the previous sprint. This restructuring established the foundational layers—Domain, Application, and Infrastructure—and introduced a clear organizational convention: NOMIS-aligned services were placed in a dedicated namespace within the Application layer, intentionally separating them from generic application services. Correspondingly, the API endpoints derived from NOMIS actions were grouped under a */nomis* route.

Sprint Review

The Sprint Review highlighted several technical issues requiring attention. Time-zone handling was inconsistent, leading to discrepancies in workout scheduling. Masters were unable to edit workouts after creation, which limited real-world applicability. Additionally, students could not unenroll from workouts, revealing further user states that had not yet been modelled. These issues were documented and added to the backlog as priority refinements for the next Sprint.

DSR Review

The EDD was revisited and refined with additional information items, resulting in a version that more accurately reflected the system as it was beginning to take shape (illustrated in Figure 5.5 from section 5.6.1). While the adoption of Clean Architecture successfully introduced structural separation, the DSR review concluded that the translation of environmental states from the EDD into code remained implicit and insufficiently articulated. It became clear that environmental states needed a more explicit representation in the implementation to maintain alignment between the conceptual model and the evolving system.

5.5.5. Sprint 5

This sprint focused on resolving the issues identified in Sprint 4. Masters were now able to edit and delete workouts, and students were given the ability to unenroll. The central development effort, however, was the implementation of attendance registration by masters. This required the introduction of a new entity, Exercise, associated with workouts and specific belt levels, which is essential for calculating student progression as part of future actions. An *AttendanceStatus* enum was also added to support student attendance registration in a more structured manner.

Sprint Review

The introduction of Exercises led to substantial modifications in the data model and front-end interactions. Although the new *AttendanceStatus* enum proved immediately useful for representing attendance events, it also exposed several architectural weaknesses. Small conceptual extensions, such as adding an entity or a state, were generating disproportionate complexity in the codebase. This indicated that the current Clean Architecture implementation did not yet provide the level of expressiveness or modular separation required to support the growing conceptual model in a scalable way.

DSR Review

The DSR review confirmed these architectural concerns. While Clean Architecture had introduced a layered structure, it did not yet offer a clear mechanism to encapsulate rules derived from NOMIS environmental states. For example, rules such as “a student can only attend if enrolled” or “a workout can only be scheduled in an available room” had no explicit representation in the architecture. Their enforcement remained arbitrary across services and controllers, making the conceptual model difficult to trace in the implementation. This highlighted the need for a more explicit and systematic way to express environmental states and associated constraints within the codebase, setting the stage for the introduction of the Specification Pattern in the following sprint.

5.5.6. Sprint 6

During this sprint, the focus shifted toward implementing student progress calculation. A new endpoint was developed to automatically determine a student’s belt advancement based on the workouts they attended, and the exercises associated with their current belt level. The front-end was enhanced with dedicated SVG assets for each belt and degree, offering a clearer and more visually informative representation of progression.

Sprint Review

The initial version of the progress-calculation feature encountered several limitations due to incomplete data and missing rules. Junior belts—such as grey, yellow, orange, and green—were particularly difficult to evaluate because many of their mandatory exercises had not yet been defined.

The review also noted that belt evolution rules did not consider age-related constraints, which are fundamental in real Jiu-Jitsu progression systems. Additionally, gaps in the definition of degree-specific exercises prevented accurate evaluation for advanced students. These issues reinforced the need for a more comprehensive dataset and clearer business rules before progress calculation could be reliably implemented.

DSR Review

From a DSR perspective, the architectural concerns identified in previous sprints were temporarily set aside, as the sprint's primary focus had shifted to progress calculation. The DSR review therefore did not introduce new conceptual or architectural refinements, noting instead that the methodological work on NOMIS-related structures would resume once the functional scope for student progression had been consolidated.

5.5.7. Sprint 7

This sprint introduced a complete progress page for both students and masters, along with improvements to graphical assets and overall UI consistency. These additions made the system significantly more usable and provided a clearer representation of each student's advancement through the belt hierarchy.

Sprint Review

With the increasingly complete set of user flows, it became possible to test full interactions between masters and students. However, the growing complexity of the domain model made it evident that the architecture required stronger mechanisms to manage rules, dependencies, and role-based behavior. The review concluded that, although functional progress was being made, the architectural foundation was still insufficient to support the expanding conceptual model in a clean and scalable way.

DSR Review

This DSR meeting was one of the most transformative in the project. Three major insights emerged:

1. **Repository Pattern** – needed to isolate data access and ensure that domain logic remained independent from persistence concerns.
2. **Specification Pattern** – offered a direct mechanism to encode NOMIS environmental states as executable logic, enabling a clear one-to-one mapping between conceptual conditions and system behavior.
3. **Strategy Pattern** – provided a clean structure for implementing role-based decision-making, aligning closely with NOMIS's emphasis on differentiated authority and actor responsibilities.

Together, these insights required a significant rethinking of the architecture and defined the primary goals for Sprint 8, which would focus on integrating these patterns into a coherent and NOMIS-aligned foundation.

5.5.8. Sprint 8

This sprint was dedicated to the architectural refactoring defined in Sprint 7. The Repository, Specification, and Strategy patterns were implemented across the back-end, requiring a restructuring of the Clean Architecture layers to incorporate these patterns effectively. As part of this refactoring, a unified User base entity was introduced, from which Student and Master derive, and the Identity Framework was adapted to operate within this revised structure.

Sprint Review

The architectural restructuring revealed that the initial implementation of the Specification pattern had become overly complex. Excessive use of custom criteria and expression builders made specifications difficult to understand and fragile to maintain. This complexity signaled the need for simplification before the pattern could serve as a reliable foundation for representing domain rules.

DSR Review

The DSR session led to a decisive simplification of the Specification pattern, reducing it to a minimal but expressive core: a single *IsSatisfiedBy* method combined with logical operators (*And*, *Or*, and *Not*). This streamlined version aligned closely with the nature of NOMIS environmental state validation and significantly improved both readability and maintainability. The clarity achieved through this refinement marked a turning point in translating NOMIS concepts into executable code, providing a robust basis for future architectural decisions.

5.5.9. Sprint 9

The final sprint focused on refining the architecture based on the previous DSR feedback, simplifying the implementation of the Specification pattern, improving overall UI consistency, and resolving remaining minor bugs. At this stage, Vortex Combat reached a complete and stable form, with all core user flows fully operational.

Sprint Review

The Sprint Review confirmed that the application had achieved the expected level of functionality. All major features were implemented, and the system behaved consistently with the intended design. The project was considered ready for evaluation, with the implemented workflows aligned with the research objectives established at the beginning of the development process.

DSR Review

The concluding DSR review confirmed that the architectural evolution had reached a stable and coherent state. The final versions of the EDD and HAT were now fully translatable into code:

- Environmental states were represented as Specifications.
- Human actions corresponded to application services.
- Role-specific behaviors were encapsulated through Strategies.

At this stage, the NOMIS model no longer existed only as a diagrammatic artefact; it had been structurally integrated into the application's architecture. The core workflow was considered fully testable and conceptually stable, validating the entire iterative process that combined technical development with conceptual refinement across the sprints.

5.6. Integration and Consolidation

The final data model, shown in Figure 5.4 below, reflects significant evolution from the initial version (presented earlier in Figure 5.2).

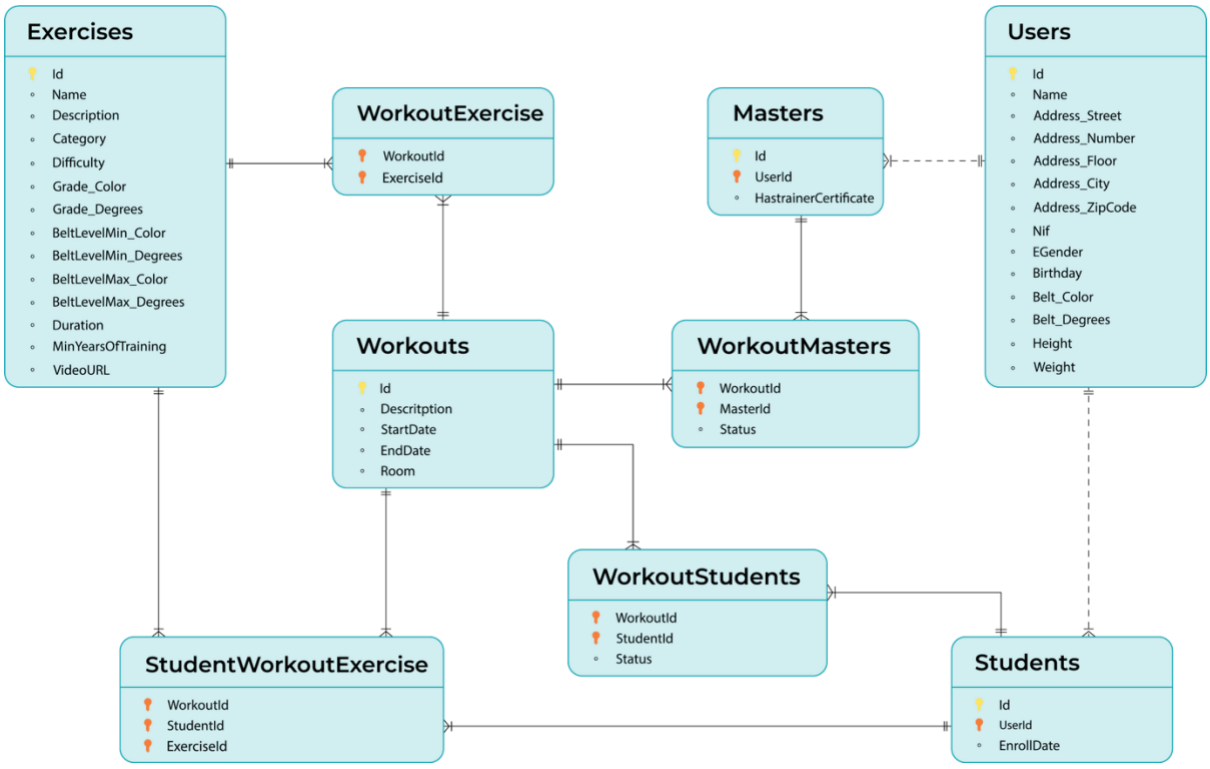

Figure 5.4 - Final Entity-Relationship Diagram

The database Entity-Relation diagram evolved through several key structural changes to support the application's requirements.

User management was restructured by integrating the ASP.NET Identity Framework. Both *Masters* and *Students* now inherit from a common *Users* table, which itself extends the framework's *AspNetUsers* table. This separation keeps authentication details secure while storing role-specific application data efficiently. A new AspNetUserRoles table was added to manage authorization and permissions.

New core entities were introduced to model the domain. An *Exercises* table was created to store techniques with their associated belt levels and prerequisites, which is essential for both progress tracking and student safety. Relationships were established to link *Exercises* to *Workouts*, *Students*, and *Masters* through junction tables.

Existing relationships were enhanced with state tracking. The *WorkoutMaster* and *WorkoutStudent* tables now include a status field (None, Enrolled, Attended) to manage the workout lifecycle. This state information is crucial for calculating student progress and enforcing business rules, such as preventing double-booking.

5.6.1. User Flows

Now that the final data model has been presented, the main user flows can be detailed. Figure 5.5 displays the final EDD that evolved from its previous version present in Figure 5.1 in section 5.3, including new states and a clearer distinction between user concerns.

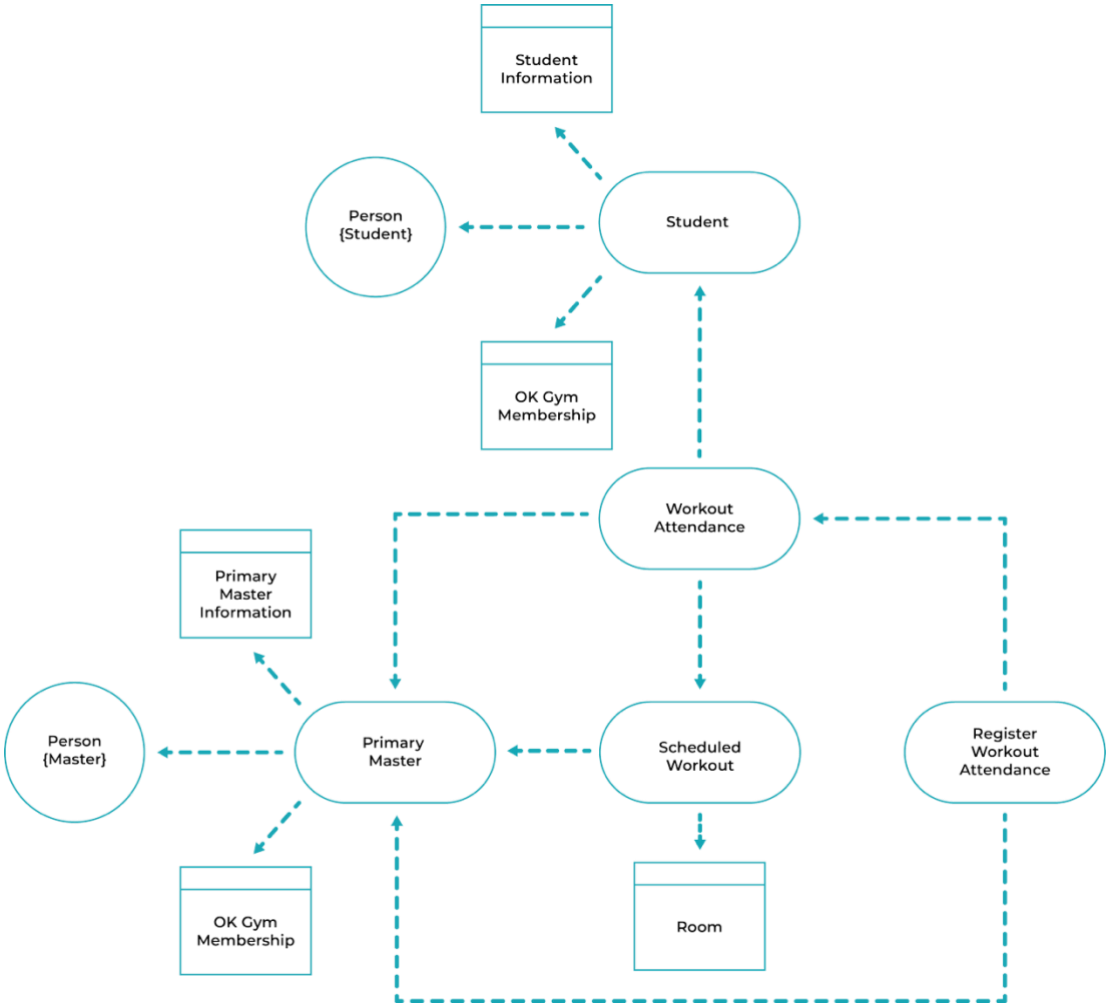

Figure 5.5 - Final EDD for workout attendance registering

These dependencies establish the system's operational rules by linking each permissible human action to a required enabling environmental state. The enforcement of this dependency—where a validated state permits a subsequent action—is implemented through Specifications that guard the business logic.

Master Attendance Registering

1. Create exercises/techniques via the form.
2. Schedule a workout and assign exercises to it.
3. Perform the training.
4. Consult the list of students and select who in fact attended the workout.
5. Submit attendance.

The human actions for this feature are **Workout Scheduling** and **Attendance Registering**. The key environmental states are **Workout Scheduled** and **Attendance Registered**, which can be directly verified by the *WorkoutIsSchedulableState* and *AttendanceState* Specifications.

Student Progress

1. Enroll in a scheduled workout that contains exercises related to their belt.
2. Attend it and successfully apply the train the techniques for their belt.
3. Wait for master to register attendance.
4. Consult the progress page and see how many exercises remain for the next graduation. A green message will show when they are ready to graduate.

The human actions for this feature are **Enroll in Workout** and **Consult Progress**. The key environmental states are **Student is Enrolled** and **Student is Eligible for Promotion**, which can be directly verified by the *EnrollmentState* and *ProgressState* Specifications.

5.6.2. Architectural Coherence

In a typical API call, such as a student enrolling in a workout, the architectural layers cooperate in a clear sequence. The request is first received by a Controller in the **Presentation** layer, which handles HTTP-specific concerns. It then invokes a use case in the **Application** layer, which orchestrates the business logic. This case employs Specifications to validate business rules and uses Repository interfaces defined in the **Domain** to request data. The **Infrastructure** layer provides the concrete implementation of these repositories, using Entity Framework to persist the enrollment data in the database.

5.6.3. Scope

With a working application that supported the core user flows, it became necessary to define the boundaries of what would be considered complete for this phase of the project. The development focused primarily on features that directly implemented and validated NOMIS concepts, while several potential enhancements were intentionally deferred.

These included:

- Improving the belt progress calculation to consider factors like repeated technique practice and training duration, rather than assuming mastery from a single attendance.
- Expanding the exercise library for more accurate progress tracking.
- Implementing an announcement board for notifications were also identified as valuable but non-essential features.

These items were excluded not due to technical constraints, but because they represent incremental improvements rather than core validations of the NOMIS methodology.

5.7. Summary

This chapter documented how architecture, data model, and features evolved through iterative sprints under a DSR with Scrum cycle. The implementation converged to a Clean-Architecture codebase where NOMIS concepts are explicit in domain entities and Specifications, and end-to-end flows reflect observable human actions and their enabling states. The next chapter evaluates the system and the approach with respect to applicability, limitations, and lessons learned.

Chapter 6

Results

This chapter presents the practical outcomes of developing the Vortex Combat system using the NOMIS methodology. It examines three key aspects: the functional application that was created, the effectiveness of NOMIS, and the insights gained from the hybrid DSR with Scrum process. The analysis connects the implemented features directly to their foundation in NOMIS models and specifications, demonstrating how theoretical concepts translated into working software. The findings provide answers to the research questions about NOMIS's influence on system design, architectural decisions, and practical implementation challenges. This evaluation forms the basis for understanding the methodology's strengths and identifying opportunities for future refinement.

6.1. Evaluation

Evaluating a complete NOMIS-based system poses unique challenges, as few comparable implementations exist. For this reason, the evaluation draws primarily on observations made throughout the iterative development process—particularly the DSR reflection sessions performed after each sprint. These reflections made it possible to verify whether the emerging system remained aligned with the conceptual model and to adjust the design whenever inconsistencies or misunderstandings appeared.

Once the final version of Vortex Combat was completed, the system was tested end-to-end using all key user flows. This final evaluation assessed how well the implemented features reflected the observable human actions and dependencies identified during modelling, providing a practical basis for analyzing the effectiveness of NOMIS.

6.2. Results and Findings

This section presents the outcomes of the project, examining the final Vortex Combat application, the practical application of the NOMIS methodology, and the effectiveness of the development process.

6.2.1. Vortex Combat

The Vortex Combat application addresses the core problem identified in the case study: the manual and non-transparent process of tracking student progress in a Jiu-Jitsu gym. The system provides a clear alternative to paper-based records and reliance on direct master feedback by giving students autonomous access to their belt progression, completed techniques, and attendance history. The final, functional version of the application is available for review in its public repository at https://github.com/andreibalcan/vortex-combat.

Authentication

The application implements a straightforward authentication system that uses the Identity Framework and is split into two distinct roles: master and student. Masters have a pre-existing account, while students can self-register through the application.

Figure 6.1 shows the login screen, which provides a unified entry point for both masters and students.

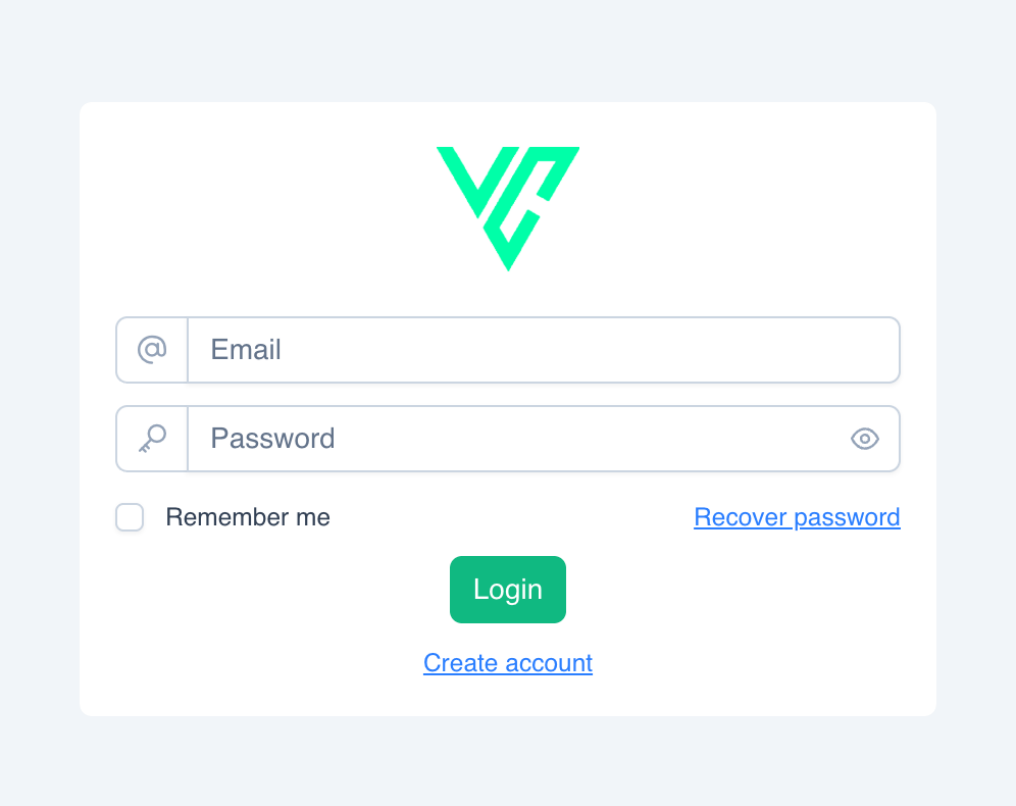

Figure 6.1 - Authentication Screen

Figure 6.2 shows the student registration process. It requires them to provide personal details, authentication credentials and their current belt and degrees in Jiu-Jitsu which will be used as a baseline to automatically calculate and track their progress towards the next graduation.

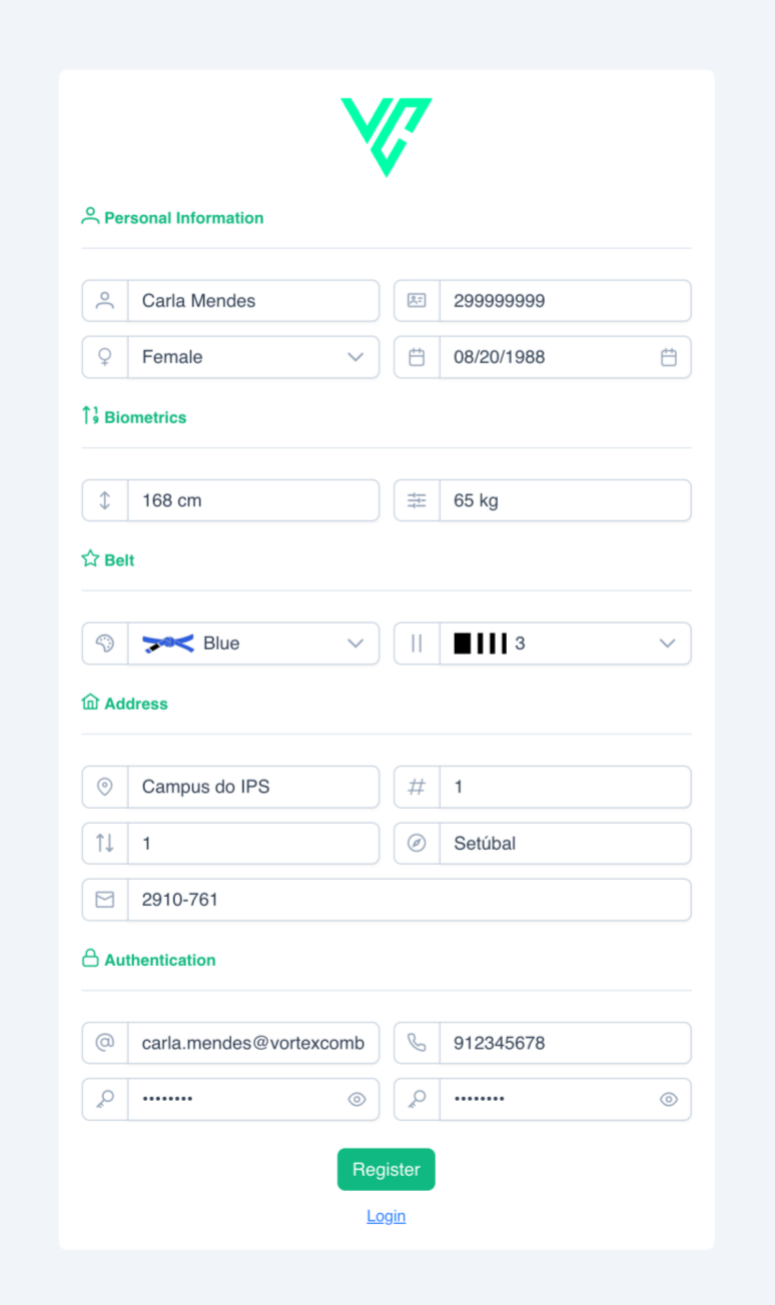

Figure 6.2 - Registration Screen

Exercises

The system requires an initial setup phase where a master defines the techniques and exercises. These exercises form the foundation of the application, as they are assigned to scheduled workouts and made available for students to consult.

Figure 6.3 displays the form used by masters to create a new exercise, capturing all necessary details such as the name, description, and the specific belt and degree it belongs to.

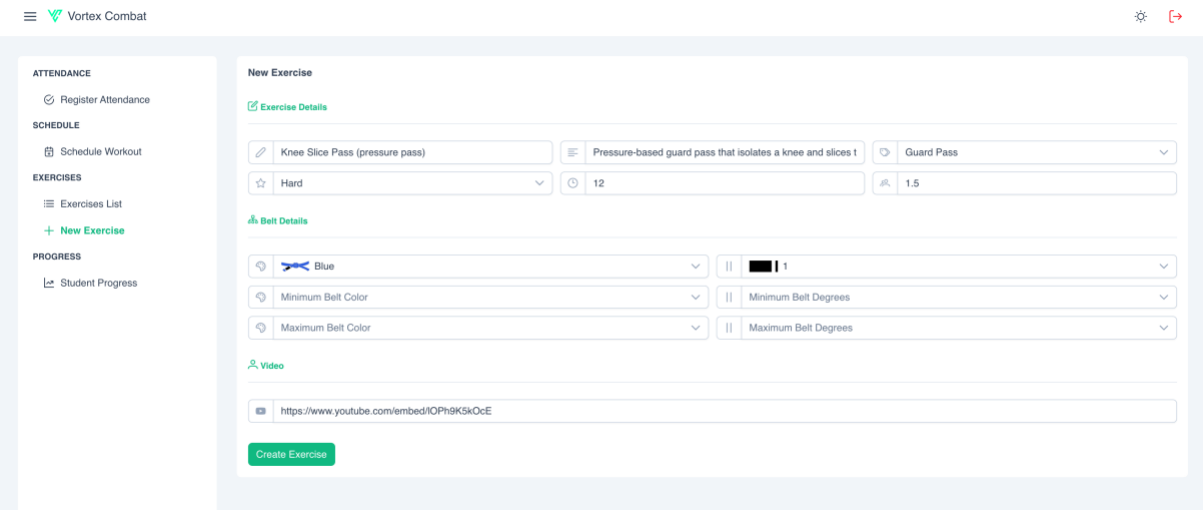

Figure 6.3 - Exercise Creation Screen

Once created, this library of exercises becomes accessible to both masters and students for reference. The interface for consulting exercises, shown in Figure 6.4, is organized hierarchically. It uses tabs to separate exercises by belt color and expansion panels within each tab to categorize them by degree.

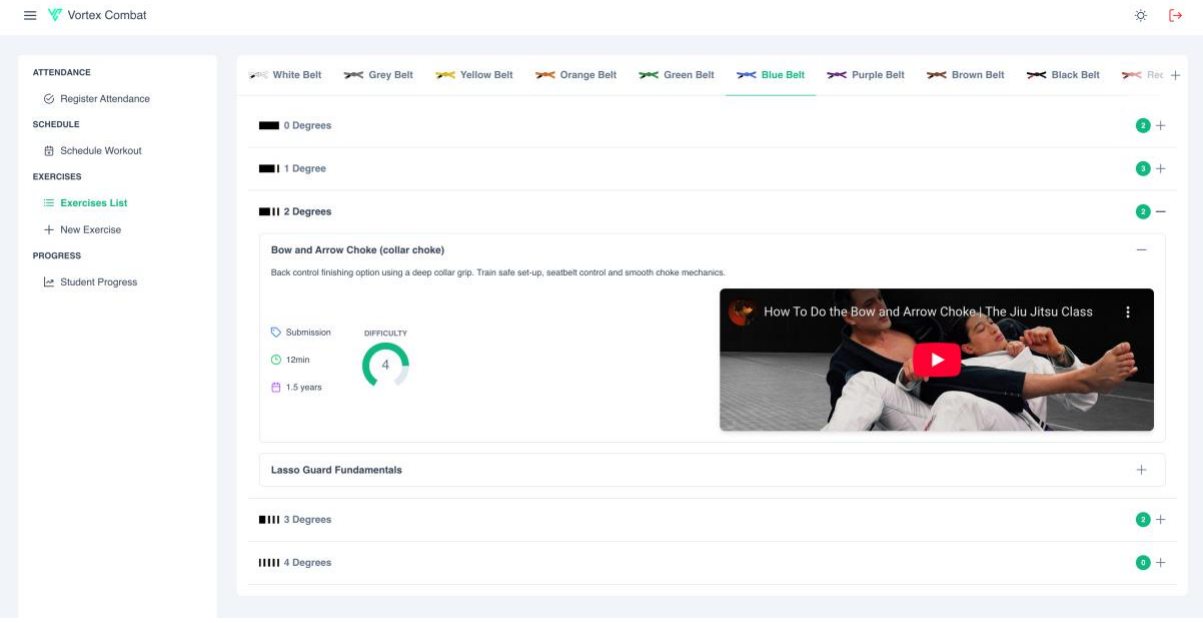

Figure 6.4 - Exercise List Screen

Workouts

Once exercises are defined for the various belts and degrees, the master can schedule workouts using the calendar interface. A scheduled workout can later be edited either by clicking on it to open a form or by directly dragging it to a new time slot on the calendar. The system validates this action using the *WorkoutIsSchedulableState* specification to ensure the scheduling follows the rules. Figure 6.5 illustrates the workout scheduling interface available to the master.

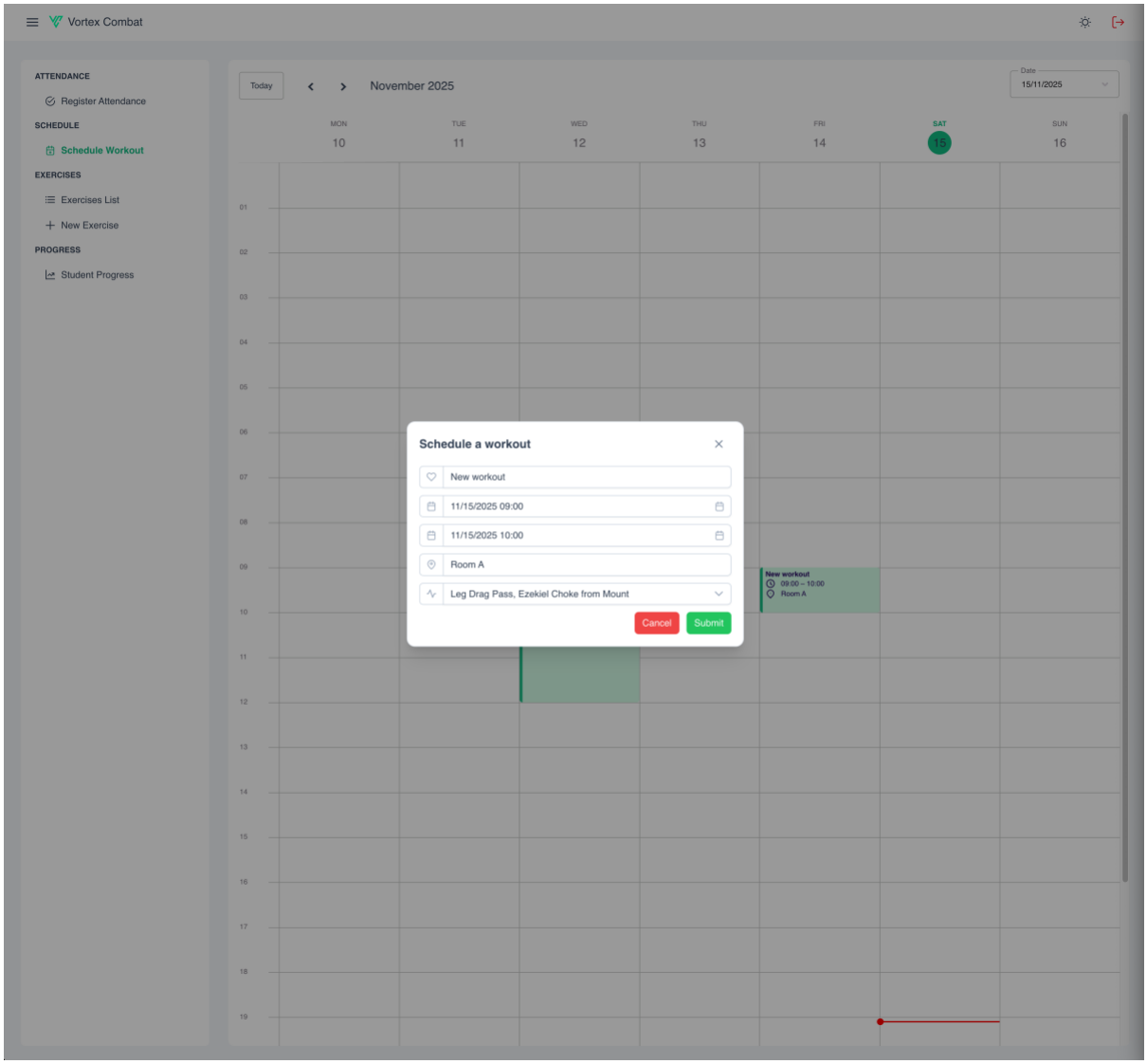

Figure 6.5 - Workout Scheduling Screen

After a workout is scheduled, students can enroll in it. This enrollment is a prerequisite for the subsequent attendance validation performed by the master. The enrollment process is validated by the *EnrollmentState* specification, which checks that both the student and workout exist and that the student doesn't have a scheduling conflict with another workout. Figure 6.6 shows the enrollment screen, where a student can view the exercise list, the assigned master, and other enrolled students. Students also have the option to unenroll from a workout before it takes place.

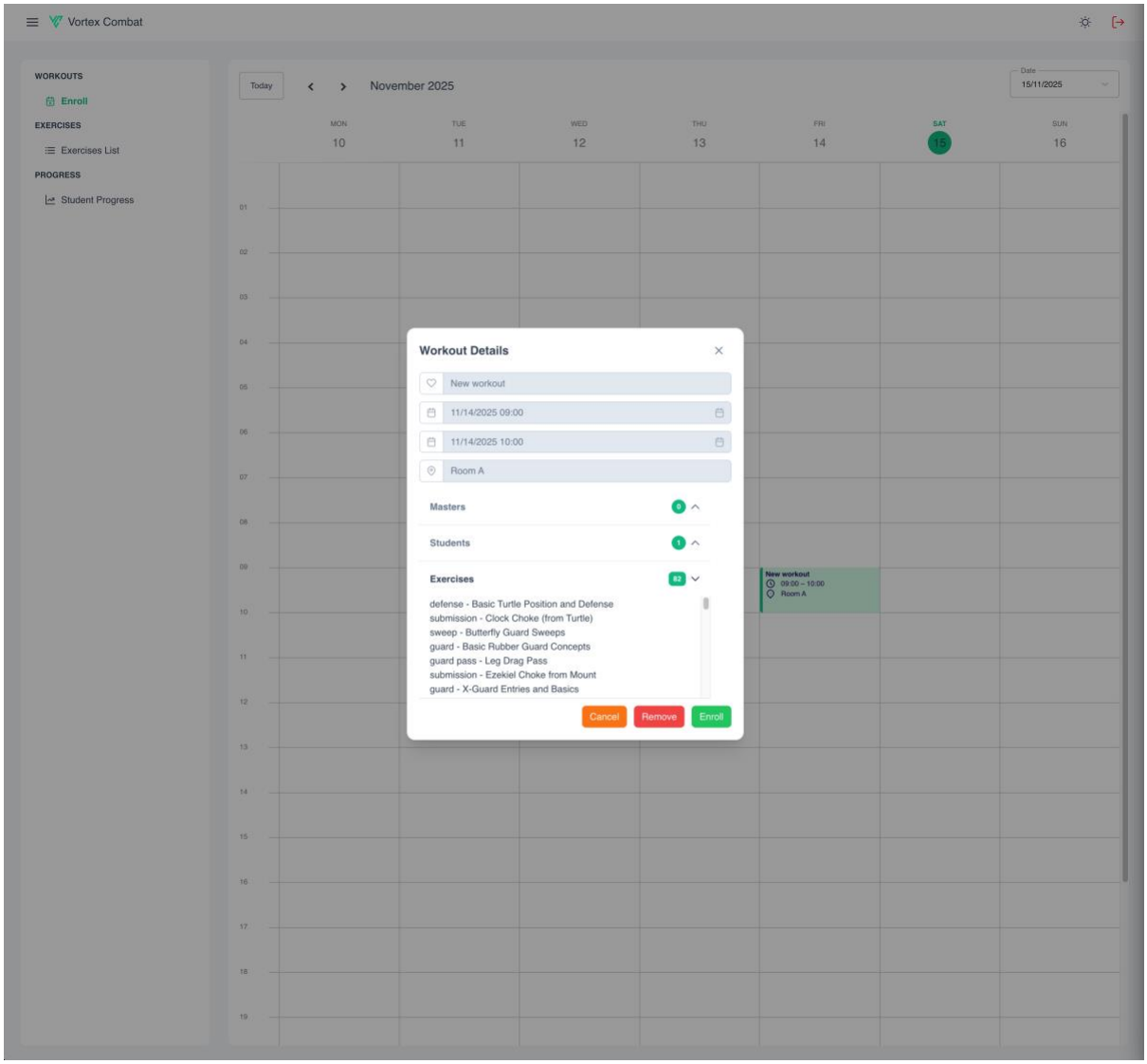

Figure 6.6 - Workout Enrollment Screen

Attendance

Once a workout is completed, the master can record which students attended. This is done by selecting the finished workout from a dropdown menu and then marking the participating students from a list. This process is validated by the *AttendanceState* specification, which ensures that the workout and student exist, confirms the student was enrolled, and verifies that the workout's end date has passed. Figure 6.7 shows the interface for registering attendance, where the master can efficiently confirm participation.

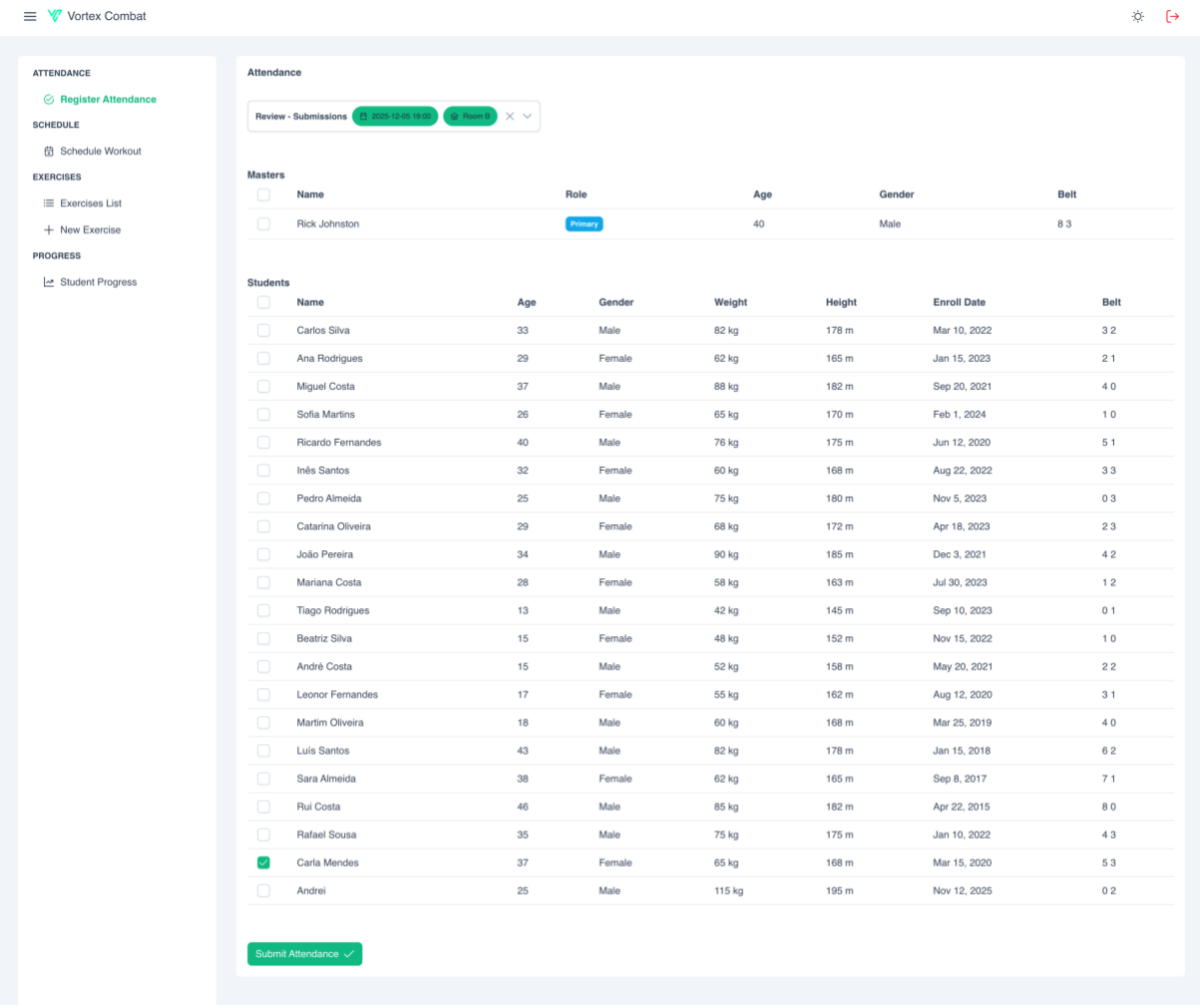

Figure 6.7 - Attendance Registering Screen

Progress

Once attendance is recorded, both masters and students can review belt progression. Masters have access to all student progress, while each student can only view their own advancement. This feature is governed by the *ProgressState* specification.

Figure 6.8 shows the progress dashboard from a master's perspective. The interface displays whether a student is ready for promotion to the next belt, provides a history of attended workouts, and lists the remaining exercises the student needs to perfect for their current rank.

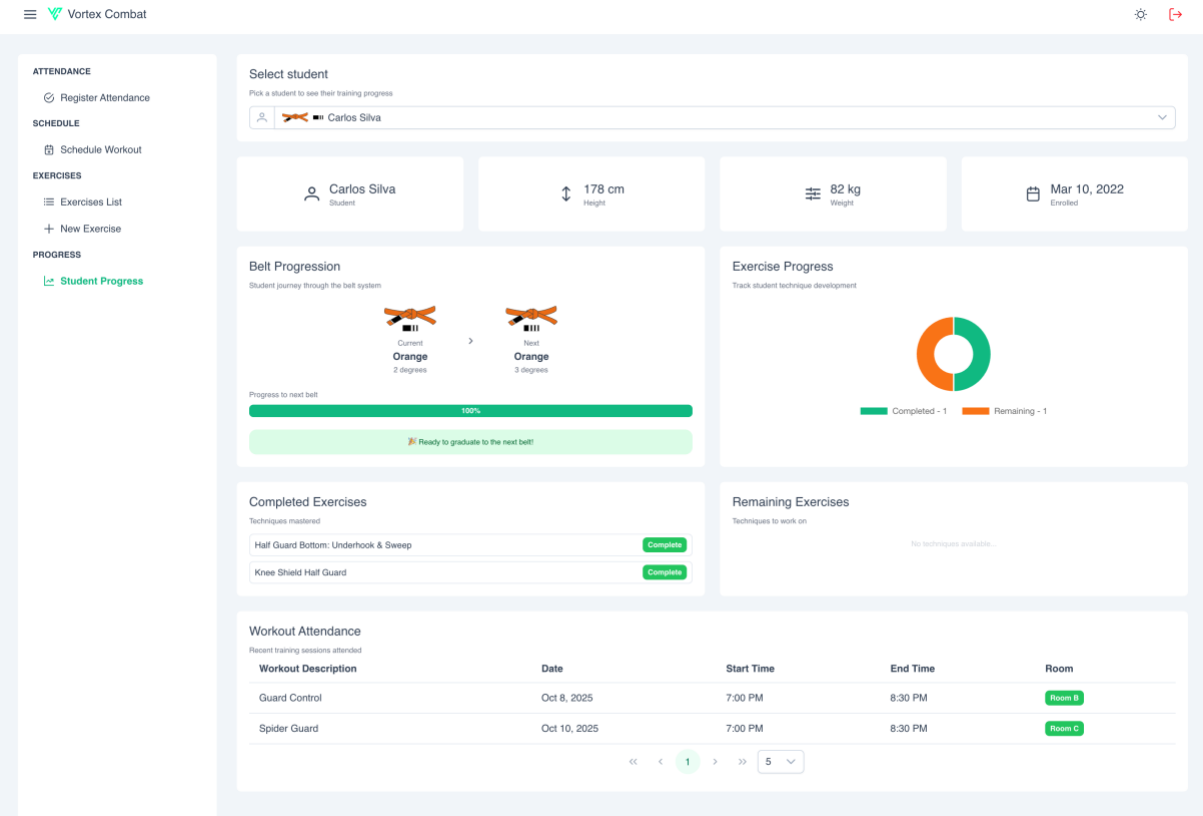

Figure 6.8 - Progress Screen

6.2.2. NOMIS Applicability

Applying NOMIS to the Vortex Combat project proved effective from the initial stages. Beginning with a concrete case study allowed for a straightforward translation of real-world gym operations into a formal model. The initial modeling phase using Existential Dependency Diagrams (EDD) and Human Action Tables (HAT) was very valuable, as it systematically identified the core actors, observable human actions, and enabling environmental states. This clear, structured starting point provided a solid foundation that guided the subsequent development.

NOMIS's focus on the business domain aligned well with the chosen technical implementation. Clean Architecture provided a natural framework for separating the core domain logic from technical concerns, while design patterns like Specification created a direct, one-to-one mapping between NOMIS environmental states and code. For example, the environmental state *AttendanceCanBeRegistered* was directly implemented as an *AttendanceState* specification, verifying that the workout has occurred and that the selected students were enrolled. This synergy between methodology and architecture made the translation from model to system more intuitive.

A key insight from using NOMIS was the prioritization of back-end development. By ensuring that the core business logic—implementing actions, states, and their rules through specifications—was robust and correct, the front-end could be developed with confidence as a flexible interface layer. This

approach confirmed that a well-structured domain model, guided by NOMIS, provides a stable foundation for the entire system.

6.2.3. Methodology

The hybrid DSR with Scrum approach proved effective for this research-oriented development. The flexible, adaptive nature of the process allowed the focus to remain on the core objectives: building a functional system and refining the understanding of NOMIS through practical application.

The flexibility of deadlines created an environment that could focus on deep analysis and iterative learning. The cyclical process of building in Scrum sprints and then reflecting in DSR sessions was crucial. This meant that development could move forward even when the full path was not entirely clear, as each DSR review provided the necessary insights to correct course and define the subsequent steps. The methodology essentially turned into a structured process of discovery, ensuring that each iteration contributed both to the application and to the methodological research.

6.2.4. Analysis

The project outcomes provide concrete answers to the initial research questions, demonstrating how NOMIS functions in practice.

RQ1 – How does NOMIS influence the design and development of an information system when applied as the main conceptual framework?

NOMIS significantly influenced the system's design by providing a clear and structured path from problem domain to solution. The methodology's modeling steps—particularly the initial creation of Existential Dependency Diagrams and Human Action Tables—offered a systematic way to identify core system components. This approach ensured development progressed with a clear understanding of human actions and their relationships from the very beginning.

RQ2 – How can software architecture be defined and adapted to align with NOMIS's socio-technical principles during implementation?

The software architecture evolved naturally from NOMIS principles. Clean Architecture provided the structural foundation, creating distinct layers that separated core business logic from technical implementation. The Specification pattern proved crucial for maintaining alignment with NOMIS concepts, creating a direct correspondence between environmental states in the models and validation rules in the code.

The four views of NOMIS directly informed the core architectural decisions:

- **State View** – guided the core business logic through the Specification pattern, which directly implemented environmental states like *EnrollmentState* to validate business conditions before actions could execute.
- **Interaction View** – defined the system's API structure, where endpoints like "Register Attendance" and "Enroll in Workout" directly correspond to observable human interactions between masters and students.
- **Physical View** – informed the repository layer and data models, ensuring that entities like *Workout* and *Student* properly represented the physical elements involved in gym activities.
- **Information View** – shaped the data contracts and interfaces, maintaining clear separation between business information and its technical representation.

RQ3 – What challenges and limitations arise when translating NOMIS's theoretical concepts into practice?

The main challenge was the initial learning curve. Before writing a single line of code, it was necessary to understand the philosophical stance of Human Relativism and how theories like Organizational Semiotics fit together. This required a significant investment of time. However, this investment paid off. Once the core concepts clicked, the path forward became much clearer.

The results obtained can be analyzed along three complementary dimensions: the functioning of the system developed, the applicability of NOMIS during design and implementation, and the effectiveness of the combined DSR with Scrum methodology.

Impact on the system

The implemented prototype provides a coherent and effective solution to the operational needs of the gym. Core user flows—such as scheduling workouts, registering attendance, and viewing student progression—were implemented consistently and supported by a clear domain structure. The early conceptual work reduced ambiguity during development, resulting in cleaner logic and more predictable behavior across the application.

Applicability of NOMIS

A key objective of the project was to assess whether NOMIS could guide the development of a real information system. The project offers concrete evidence that this is feasible. Environmental states identified during modelling were implemented as Specifications in the domain layer, and human observable actions informed the definition of system services. The iterative refinement of the EDD ensured that the conceptual understanding of the domain evolved together with the implementation. Although some learning effort was required to identify actions, states, and dependencies correctly, NOMIS proved useful in structuring the domain and supporting clear decision-making.

Effectiveness of the methodology

The hybrid DSR with Scrum approach supported both iterative progress and methodological reflection. Sprint Reviews helped improve technical aspects such as architecture, user experience, and feature completeness, while DSR reflections led to deeper conceptual changes, such as adopting Clean Architecture and introducing the Specification and Strategy design patterns. These changes were not merely technical decisions: they emerged from analyzing the conceptual model and ensuring alignment between NOMIS constructs and system behavior. This dual process strengthened the coherence between modelling and implementation.

In summary, the analysis shows that the project successfully aligned conceptual modelling with practical development. NOMIS contributed to clarifying the domain and guiding key architectural decisions, while the DSR with Scrum process ensured continuous reflection and adaptation. Together, these results validate the relevance of NOMIS for designing and implementing human-centered information systems.

6.3. Lessons Learned

The primary lesson from this project is that NOMIS can in fact guide development by focusing on what truly matters: the human actions within a system. Unlike traditional methodologies that often involve extensive documentation of system states or complex data flows, NOMIS provided a direct path from observing real-world activities to implementing software features. This approach eliminated unnecessary abstraction and planning steps that typically consume time without adding clear value.

Another lesson learned was that NOMIS's modeling tools forced a clear understanding of the problem domain before any technical decisions were made, which in turn made the subsequent development more purposeful and efficient.

One last lesson learned is that a well-structured back-end, guided by NOMIS principles, creates a stable foundation for the entire application. By solidifying the core business logic, the front-end development became a more straightforward task of creating interfaces for these predefined actions. This contrasts with approaches where front-end and back-end development are tightly coupled, often leading to confusion and rework.

6.4. Limitations and Future Work

The project successfully validated the core NOMIS concepts, but the implemented system has several limitations that point to opportunities for future work. The current application focuses on automating the tracking of progress but lacks a direct way for a master to officially promote a student to the next belt. Furthermore, the progress calculation is relatively simplistic, assuming that attending a single workout where a technique is taught translates to mastery. A more realistic approach would consider how much practice someone needs and how long they usually spend at each belt level.

Finally, to solidify NOMIS as a practical methodology, the development steps and architectural decisions taken in this project should be formally documented into a more structured guide. Complementing this, deploying the application in a real gym setting would provide invaluable data on its practical utility and uncover new insights for refining the NOMIS methodology itself through sustained real-world use.

Chapter 7
Conclusion

This chapter summarizes the work conducted throughout the project and highlights its main contributions, limitations, and opportunities for future development. The goal is to consolidate the findings obtained during the design, implementation, and evaluation of the Vortex Combat application and to assess the broader implications of this work for the NOMIS methodology.

7.1. Summary of the Work

The project explored the use of NOMIS as a conceptual foundation for developing a human-centered information system. After studying the theoretical background of NOMIS and modelling the domain using diagrams such as the EDD and HAT, a complete system was designed and implemented using a hybrid DSR with Scrum methodology.

The resulting web application supports core gym operations—including workout scheduling, attendance registration, and belt-progress tracking—and reflects the observable human actions and environmental states identified during modelling.

The result was a working system that operationalizes NOMIS concepts and provides empirical evidence of the methodology's applicability in practice.

7.2. Main Contributions

The project delivers several contributions of practical and scientific relevance:

A functional NOMIS-based system

The Vortex Combat application demonstrates the feasibility of implementing a complete system directly informed by NOMIS modelling. It is the first fully operational prototype developed in this context.

A practical bridge between NOMIS and modern architecture

Concepts such as environmental states and human observable actions were mapped to architectural constructs, especially using Clean Architecture and patterns such as Specification, Strategy, and Repository.

A validated methodological process

The hybrid DSR with Scrum process proved suitable for maintaining conceptual coherence while supporting iterative software development. The dual review structure (Sprint Review + DSR Review) was particularly effective.

Collectively, these contributions showcase NOMIS as a viable methodology for building human-centered information systems.

7.3. Validation and Limitations

The evaluation showed that NOMIS can effectively guide the development of a real information system. The system's domain logic, user flows, and architectural structure all reflect the concepts identified during modelling. Environmental states were successfully implemented as Specifications, and

human actions were translated into system services. These results support the applicability of NOMIS as a modelling and implementation methodology.

Despite these achievements, the work has some limitations:

- The implemented system simplifies certain aspects of gym progression, such as detailed belt-progress rules and technique mastery.
- The project validates NOMIS only in a single domain, which limits the generalization of results.
- The methodology requires a learning curve, particularly when identifying environmental states and structuring dependencies.

These limitations do not undermine the success of the project but indicate areas where further refinement and experimentation would be beneficial.

When looking back at the initial objectives of the project, all four goals were successfully met, demonstrating the practical viability of the NOMIS methodology:

O1 – Develop a fully functional application that demonstrates how NOMIS concepts can be implemented in a real-world scenario.

The Vortex Combat application delivers a complete working system where students can track belt progress and masters can manage gym operations. All core features—from workout scheduling to attendance tracking—were implemented following NOMIS principles, proving the methodology's applicability to a concrete problem domain.

O2 – Define and evolve a software architecture consistent with NOMIS's socio-technical foundations, ensuring consistency between conceptual models, system design and technical implementation.

This objective was validated through the implementation of Clean Architecture complemented by Repository, Specification, and Strategy patterns. This combination successfully provided a structured yet flexible codebase that maintained a consistent alignment with the NOMIS conceptual models throughout the development process.

O3 – Apply and document a methodological approach that integrates reflective and iterative development practices, allowing adaptation throughout the process.

The hybrid DSR with Scrum approach provided the necessary structure for continuous adaptation. Regular DSR reflections after each sprint enabled course corrections and methodological refinements, demonstrating how research and development can effectively coexist in a single project.

O4 – Evaluate the practical outcomes of applying NOMIS, identifying strengths, limitations, and lessons learned for future research and development.

Chapter 6 provides a comprehensive evaluation of the methodology's strengths in creating human-centered systems, while also identifying limitations such as the initial learning curve and opportunities for enhancing the progress calculation logic.

7.4. Future Work

Future improvements could extend both the system and the methodology:

- Enhance the progression model with more realistic rules and additional exercise types.
- Deploy the application in an operational environment to gather user feedback and assess long-term usability.
- Apply NOMIS to different domains—such as healthcare, education, or administrative systems—to further validate its generality.
- Formalize the architectural and development patterns used in this project into practical guidelines for NOMIS-based development.

7.5. Final Remarks

This project represented a unique opportunity to combine personal interest with academic research. Developing a system for a Jiu-Jitsu gym, a domain the student is passionate about, while simultaneously testing the NOMIS methodology with a familiar technology stack, made the entire process both engaging and effective.

The work successfully demonstrates that NOMIS offers a practical path for building information systems that genuinely reflects how people work. By focusing on observable actions rather than abstract processes, the methodology helped create a system that feels intuitive and aligned with real gym operations. The project leaves behind not just a functional application, but a proven approach that can guide future development of human-centered information systems.

References

[1] J. A. M. Cordeiro, "Normative Approach to Information Systems Modelling. PhD Thesis.," 2011.

[2] J. Cordeiro, J. Filipe and K. Liu, Towards a Human Oriented Approach to Information Systems Development, Sofia, Bulgaria, 2009.

[3] L. Kecheng, Semiotics in Information Systems Engineering, Cambridge, UK: Cambridge University Press, 2000.

[4] R. Stamper, Norms, and Information Systems. In Signs of Work, Holmqvist B. et al (eds), Berlin: Walter de Gruyter, 1996.

[5] R. Stamper, Analysing the Cultural Impact of a System, Butterworth & Co Ltd, pp. 107-122.

[6] A. W. Holt, Organized Activity and Its Support by Computer, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1997.

[7] J. Dietz, Enterprise Ontology, Theory and Methodology, Berlin Heidelberg: Springer-Verlag, 2006a.

[8] J. A. M. Cordeiro, "A Model Driven Systems Development Approach for NOMIS – From Human Observable Actions to Code," 2017.

[9] E. Mumford, Effective Requirements Analysis and Systems Design: the ETHICS Method, Basingstoke: Macmillan, 1995.

[10] P. Checkland and J. Scholes, Soft Systems Methodology in Action, Chichester: Wiley, 1990.

[11] T. Winograd and F. Flores, Understanding Computers and Congition, Norwood, NJ, USA: Ablex Publishing Corporation, 1986.

[12] J. Venable and R. Baskerville, "Eating our own Cooking: Toward a More Rigorous Design Science of Research Methods," Perth, Western Australia, Australia; Atlanta, Georgia, USA, 2012.

[13] A. Dresch and e. al, Design Science Research: A Method for Science and Technology Advancement, Springer International, 2015.

[14] J. v. Brocke and A. Maedche, The DSR grid: six core dimensions for effectively planning and communicating design science research projects, 2019.

[15] K. Schwaber, "The Home of Scrum," [Online]. Available: https://www.scrum.org/. [Accessed 2025].

[16] J. A. M. Cordeiro, "Applying NOMIS - Modelling Information Systems using a Human Centred Approach," 2017.

[17] R. C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure, Pearson / Prentice Hall, 2017.

[18] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2003.

[19] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Boston, MA, USA: Addison-Wesley Professional, 1994.