Arkitektura Koadernoa IO-System

1. Helburua

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.
[Always address Sections 2 through 6 of this template. Other sections are recommended, depending on the amount of novel architecture, the amount of expected maintenance, the skills of the development team, and the importance of other architectural concerns.]

2. Arkitektura helburuak eta filosofia

[Describe the philosophy of the architecture. Identify issues that will drive the philosophy, such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

Formulate a set of goals that the architecture needs to meet in its structure and behavior. Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions?]

3. Hipotesiak eta menpekotasunak

[List the assumptions and dependencies that drive architectural decisions. This could include sensitive or critical areas, dependencies on legacy interfaces, the skill and experience of the team, the availability of important resources, and so forth]

4. Betekizun arkitekturalki esanguratsuak

[Insert a reference or link to the requirements that must be implemented to realize the architecture.]

5. Erabakiak, mugak eta justifikazioak

[List the decisions that have been made regarding architectural approaches and the constraints being placed on the way that the developers build the system. These will serve as guidelines for defining architecturally significant parts of the system. Justify each decision or constraint so that developers understand the importance of building the system according to the context created by those decisions and constraints. This may include a list of DOs and DON'Ts to guide the developers in building the system.]

- Decision or constraint and justification
- Decision or constraint and justification

6. Mekanismo arkitekturalak

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.]

Architectural Mechanism 1

[Describe the purpose, attributes, and function of the architectural mechanism.]

Architectural Mechanism 2

[Describe the purpose, attributes, and function of the architectural mechanism.]

7. Funtsezko abstrakzioak

[List and briefly describe the key abstractions of the system. This should be a relatively short list of the critical concepts that define the system. The key abstractions will usually translate to the initial analysis classes and important patterns.]

8. Geruzak edo arkitektura frameworka

[Describe the architectural pattern that you will use or how the architecture will be consistent and uniform. This could be a simple reference to an existing or well-known architectural pattern, such as the Layer framework, a reference to a high-level model of the framework, or a description of how the major system components should be put together.]

9. Arkitektura ikuspegiak

[Describe the architectural views that you will use to describe the software architecture. This illustrates the different perspectives that you will make available to review and to document architectural decisions.]

Recommended views

- **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
- **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn't necessary if the system runs in a single process and thread.
- **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.