# Architecture Description of MVC for ERP SYSTEM

Version 2

# Table of contents

# 1 Introduction

This chapter describes introductory information items of the AD, including identifying and supplementary information.

## 1.1 Identifying information

The architecture used in our ERP system is the Model view controller (MVC) architecture. This architecture divides the system into three interconnected components in order to improve the organization and maintainability of the codebase. The three components of the MVC architecture are the Views, the Controllers, and the Models.

A view represents the visual information presented to the user. It is what the user sees.

A controller responds to the interactions of the user. For example, the click of a button may trigger the controller to perform a certain action.

A model can be seen as the brain of the system. It manages the data, logic operations, and rules of the system. The model is responsible for updating the views when changes occur.

Furthermore, the system of interest is an Enterprise Resource Planning (ERP) System for bike manufacturing. The ERP system allows users to store, update and retrieve important information regarding their business. In fact, the ERP system will be responsible for tracking raw materials, semi-finished goods, and finished goods, orders, shipments, sales, and manufacturing.

## 1.2 Supplementary information

The system is not intended to be sold to companies or third parties. The system and its architecture will meet the requirements proposed by the Product Owner and provide the desired functionalities.

## 1.3 Other information

A detailed breakdown of the requirements which must be included in the ERP system can be seen here: https://docs.google.com/document/d/1a7AbEKoKOJlIpJtZ2wHKtrLEMlTvwd_1VaWwLZQVCw4/edit?usp=sharing

### 1.3.1 Overview

This document provides an overall overview of the architecture of the ERP system. It describes the important architectural decisions which have been made prior to developing the system.

This document covers the various stakeholders and their respective concerns, the architectural viewpoints, models, views, and the known inconsistencies of the ERP system.

### 1.3.2 Architecture evaluations

The MVC architecture is implemented using ExpressJS, a nodeJs web application framework. This architecture allows for a faster development process, multiple programmers can work on different parts, the models, the views, and the controllers. It can also provide multiple views for a single model. The support of asynchronous techniques allows the website to load faster than another website.

In the ERP system, the controllers fetch data from the database to get the information of the users in order to let them log in. The models are used by the controllers to send data to the views and the database. The view receives and sends data to the controller to process the information.

# 2 Stakeholders and concerns

This section contains information about the stakeholders of the architecture, the stakeholders' concerns for that architecture, and the traceability of concerns to stakeholders.

## 2.1 Stakeholders

| Name | Description | Responsibilities |
|---|---|---|
| Product Owner | This is the primary stakeholder for which the product is made for and who will use the functionalities provided by the product. | The responsibility of the Product owner is to provide feedback on the product and to test its usability so that the product keeps evolving with each new update. |
| Requirements Engineer | These are the stakeholders in charge of gathering and eliciting the requirements of the product from the primary stakeholders. | The responsibilities of the requirements engineers are to define the functional and non-functional requirements of the product via different methods so that the needs of the primary stakeholders are met. In the end, the requirements engineers are responsible for providing a vision document in which the aforementioned requirements will be present as well as other data pertinent to the product (such as its purpose, scope, stakeholders, etc.). |

|  |  |  |
|---|---|---|
| Software Architect | These are the stakeholders that will plan and design the blueprint of the product that is to be built. | The responsibilities of the software architects are to create the structure/architecture of the project by taking into account the different requirements as well as to provide clarification of their designs to the development team which will create the product according to the software architects' schematics. |
| Project Manager | These are the stakeholders who organize the project and keep it on track so that the product is built on time, on budget, and according to the customer's needs. | The responsibilities of the project managers are to manage the different resources of the project (time, money, people, etc.), to monitor the different performance metrics in order to make sure that the project is on track, and to direct the team of developers so that they know what to work on. Also, he will make decisions on behalf of the team. |
| Software Developer | These are the stakeholders that create the actual product according to the blueprints provided by the architects and according to the directives of their project managers. | The responsibilities of the developers are to organize the project described by the architectural and design documents into different tasks, to separate them, and to work on them (e.g. by coding) so that all the needed functional and non-functional requirements arereflected in the features of the final product. In addition, the developers have the responsibility of maintaining and updating the product after its launch. |

| | | |
|---|---|---|
| Quality Assurance Developers | These are the stakeholders that will ensure that the final product being built by the developers respects a certain coding standard, that it is as error-free as possible, and that the product fulfills the needs of the customer. | The responsibilities of the quality assurance developers are to test and verify the product that the developers are creating as well as to ensure that it upholds the wanted quality guidelines and to detect any bugs and missed customer requirements. In addition, quality assurance technicians are responsible for creating a feedback loop where the product gets constantly checked by the QA and improved by the development team until both parties are satisfied. |
| Shareholder | These are the stakeholders which provide the funding for the product to be built and wish to see it grow to be successful. | The responsibilities of the shareholders are to fund the project to provide feedback on the direction that the final product is taking and on the direction that it should be taking. |

## 2.2 Concerns

| **Concerns** | **Description** |
|---|---|
| Failure of the Software | The software shall run 24/7. |
| Lack of staff availability/time | The implementation of the architecture is expected to take 4 sprints of 3 weeks each. 8 people will be required to complete each sprint. |
| Unrealistic cost estimates | The project manager shall manage the cost for the project to not be too expensive for the product owner. |

| | |
|---|---|
| Volatile requirements | The requirements shall be elicited and well documented before determining the architecture and its respective views, viewpoints, and models. |
| Lack of communication | The decisions taken regarding the architecture and the development of the system shall be made with respect to the product owner**.** |
| Legal action | The system shall not utilize copyrighted material or try to sell its services to unauthorized parties. |
| Database restructuring due to new requirements | The requirements, entities, and relations among the data shall be clearly identified and communicated to the database engineer to avoid future restructuring causing major costs and schedule overruns. |
| Real-time data | The product owner shall receive data on his ERP system in real-time in order to manage orders and products. |

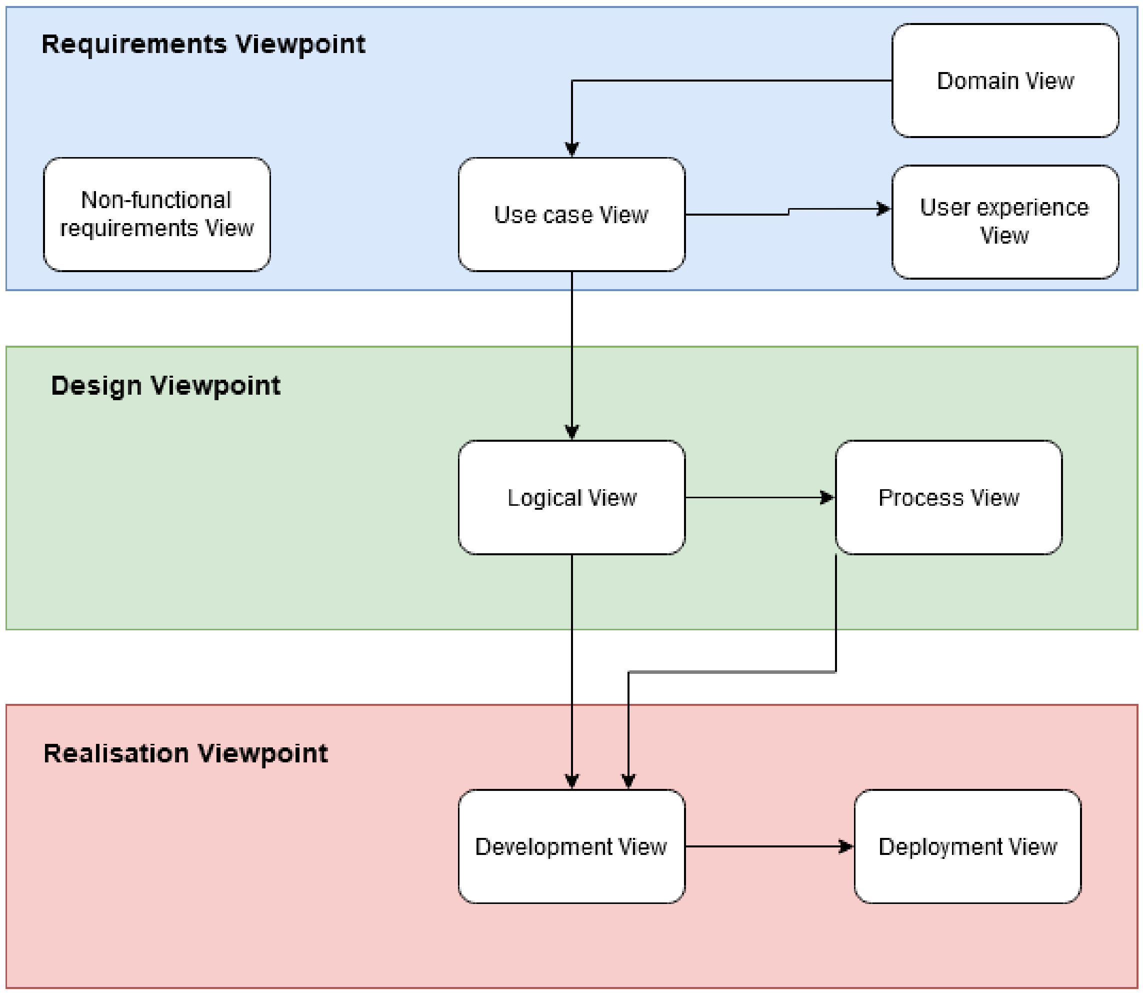
## 2.3 Concern–Stakeholder Traceability

The traceability of concerns to stakeholders can be depicted by the following table:

| | Product Owner | Requirements Engineer | Software Architect | Project Manager | Software Developer | Quality Assurance Developers | Shareholder |
|---|---|---|---|---|---|---|---|
| Failure of the Software | x | x | x | x | x | x | x |
| Lack of staff availability/time | | | | x | | | x |
| Unrealistic cost estimates | x | | | x | | | x |
| Volatile requirements | | x | x | x | x | x | |
| Lack of communication | x | x | x | x | x | | |
| Legal action | x | | | x | | | x |
| Database restructuring due to new requirements | | x | x | | x | | |
| Real-time data | x | | x | | x | | x |

# 3 Viewpoints+

A viewpoint is a way of looking at systems. They describe and illustrate the interests and concerns of various stakeholders regarding certain views. Many views can describe a viewpoint. For example, the design viewpoint includes the logical view and the process view.



## 3.1.A Requirements Viewpoint

## 3.2 Overview

This viewpoint describes the requirements of the ERP system. It depicts the essential elements and functionalities of the system. This includes the functional and non-functional requirements.

## 3.3 Concerns and stakeholders

### 3.3.1 Concerns

1) Functional capabilities: Are all the system's functional capabilities responsive and implemented?
2) External interfaces: Are the external interface understood and accepted by the product owner?
3) Internal structure: Will the software developers build the system as the software architects told them to?
4) Functional design philosophy: Are the software architects using good design to build the system or are they using an old design not in use anymore in the industry?
5) Enterprise-owned information: Is the information safely stored and not being leaked to hackers?

### 3.3.2 Typical stakeholders

The stakeholders concerned by the requirements viewpoint are all stakeholders that will interact with the system. This includes:

- the product owner
- the requirements engineer
- the software architect
- the project manager
- the software developer
- the QA developer
- the shareholders.

## 3.4 Model kinds+

### 3.5 Use-Case Diagram

A UML use case diagram is a sequence of interactions that provides value to the primary actor.

### 3.5.1 Use-case diagram conventions

The convention used for the use case diagrams is the Unified Modeling Language (UML). The use case diagram contains multiple actors and relationships between those actors. There are use cases that are actions that the actors perform. This whole diagram is in a system boundary box.

A use case describes a series of actions that are useful to the user.

- A use case is represented by a circle
- The name of the use case is contained inside the circle

An actor can be a person, a group, or any external system that interacts with the system.

- The primary actors are placed at the outer left of the use case diagram while the secondary actors are placed on the outer right
- The actors' names are precise and relevant nouns
- The actors are represented as stick figures. If the actor is not a human or organization (system), we add " <<system>> " to define what it is.
- Actors don't interact with one another but with use cases

A relationship represents an association between entities. This can be between actors, between use cases, between an actor and a use case.

- A relationship is represented by an arrow
- A relationship can have the word "extend" attached to it. This represents a use case that can occur once the parent use case is triggered.
- A relationship can have the word "include" attached to it. This means that the included use case will occur if the parent use case is triggered.

A system boundary box is a rectangle around the use cases, and it indicated the scope of our use case diagram.

## 3.6 Mockups

A mockup allows us to visualize products with graphics and images without having to build them.

### 3.6.1 Mockups conventions

A mockup will be built according to the specifications of the project. It is a non-responsive representation of the project with colors and multiple graphics. A single website can contain multiple mockups that represent each page of the website. The details are built according to the specifications of the client.

## 3.7 Operations on views

Operations define the methods to be applied to views and their models. Types of operations include:

- Construction method: The views will be constructed using draw.io. To construct the views the convention symbols will be maintained, allowing others to interpret and analyze the views. This will help the software architect to design the system and allow the developers to implement the system.
- Interpretation method: The user will need to have a basic understanding of a UML use case diagram. For further help, they will need to refer to an online description. (See 3.10 sources)

## 3.8 Correspondence rules

The use case model corresponds to the Requirements viewpoint because it highlights the essential functional requirements and actions of the system.

## 3.9 Sources

[1] *UML 2 Use Case Diagramming Guidelines*, agilemodeling.com/style/useCaseDiagram.htm.

## 3.1.B Design Viewpoint

## 3.2 Overview

This viewpoint supports the software architects and software developers in the development of the system

## 3.3 Concerns and stakeholders

### 3.3.1 Concerns

Some concerns regarding this viewpoint are:

- Installation and upgradability: How easy is it to install and upgrade the software for various platforms?
- Backup and Restore: In case of failure, how will the system be restored and will any data be lost? How often will backups occur?
- Availability: Will the system be available at all times or only during specific hours?

### 3.3.2 Typical stakeholders

The stakeholders concerned by the design viewpoint are the stakeholders that will work on designing the system. This includes:

- the system engineers
- the software developers
- the software architects

## 3.4 Model kinds+

## 3.5 Sequence diagram

The sequence diagram represents a timeline that begins at the top and descends to define the sequence of interactions. Every actor has its own column and the messages exchanged between the actors are represented by arrows.

### 3.5.1 Sequence diagram conventions

The sequence diagram is made of multiple lifeline notations depending on the number of actors. It also contains activation bars on the timeline representing the activity of an object. Message arrows are pointed from those bars from left to right and from right to left. Commented can be added.

Lifeline notation:

- A lifeline notation with a figure stick represents an actor and its own sequence
- A lifeline notation with a box represents an object of your system
- The lifelines are parallel to each other

Activation bars:

- It represents that the object or actor is active during an interaction of two actors/objects.
- The activation bar of the object who sends the message is connected to the activation bar of the person who receives the message, meaning they are both active during the same time.

Message arrows:

- It represents the synchronous or asynchronous message from the sender to the receiver.
- A message from the receiver to the sender is a return message. It gives back the data that the sender wants
- A reflexive message represents an object that sends a message to itself.
- A pointed arrow represents a comment.

## 3.6 Operations on views

Operations define the methods to be applied to views and their models. Types of operations include:

- Construction method: The sequence diagram is implemented with draw.io. It uses the proper general conventions to allow the software architects and developers to understand themselves. The sequence diagram allows anyone with little to no experience, to understand what is going on during the activity. Mainly, it is used by the developers to get a better visual of how they are supposed to implement the feature.

- Interpretation methods: The user does not need to have much experience to understand the views under the design viewpoint. Regarding the sequence diagram, it is easy to understand, and the user can seek help from the developer if he does not understand. He can also look into the link in Sources

- Implementation methods: The software developer will analyze the view (Sequence diagram), and will need to implement the features according to how it is specified.

## 3.7 Correspondence rules

The use case diagram corresponds to the design viewpoint because it describes the process view. In fact, the sequence diagram is a process implemented in the system. The feature of logging in to the ERP system is implemented by following the rules of the sequence diagram.

## 3.8 Sources

- [1] Author:Amanda Athuraliya, Amanda Athuraliya is the communication specialist/content writer at Creately, et al. “Sequence Diagram Tutorial: Complete Guide with Examples.” *Creately Blog*, 5 Jan. 2021, creately.com/blog/diagrams/sequence-diagram-tutorial/#WhatIs.

## 3.1.C Realisation Viewpoint

## 3.2 Overview

This viewpoint describes the implementation of the software components as well as the environment in which the system is intended to run for usage by external users.

## 3.3 Concerns and stakeholders

### 3.3.1 Concerns

Some concerns regarding this viewpoint are:

- Technologies used: Are the technologies used to develop the system easy to learn and work with?
- Feasibility: Is the implementation of the project feasible?
- Organization: Is the structure and organization of the codebase suitable for the ERP system being developed?
- Hosting reliability: How reliable is the third-party application in which the system will be hosted?
- Time: How long does it take to deploy the system? In case of a new release, will the time be longer?
- Network speed and capacity: How much memory is required and what transaction speeds are required to deploy the application successfully?
- Computational resources: Does the deployment require large computational power?

### 3.3.2 Typical stakeholders

The stakeholders concerned by the realization viewpoint are all stakeholders that will contribute to making the system available to users. This includes:

- the software architects
- the project manager
- the software developers
- the QA developers

## 3.4 Model kinds+

### 3.5 Package Diagram

A package diagram organizes related elements into groups and describes the relationships between the various groups.

### 3.5.1 Package diagram conventions

The convention used for the package diagram is the Unified Modeling Language (UML).

A package groups related elements.

- A package is represented by a rectangular box
- The name of the package can be seen in the top left corner

Relationships between packages represent associations, communications or dependencies.

- A relationship is represented by an arrow
- A relationship with a circle containing a plus sign inside represents a composition.

## 3.6 Deployment Diagram

A deployment diagram illustrates the runtime environment of the system. This includes the required hardware and the communication between nodes.

### 3.6.1 Deployment diagram conventions

The convention used for the deployment diagram is the Unified Modeling Language (UML).

Nodes represent hardware or software entities.

- A node is represented by a three-dimensional box
- The name of the node can be seen in the top left corner of the box

A database represents data storage for the system.

- A database is represented by a cylinder.

A relationship describes the connection or communications between nodes.

- Relationships are represented by an arrow

Components describe a software element.

- A component is often represented by a rectangle or by its logo.

## 3.7 Operations on views

- Construction method: The views will be constructed using draw.io. To construct the views the convention symbols will be maintained, allowing others to interpret and analyze the views. This will help the developers to implement the system and facilitate the deployment phase.

- Interpretation method: The user will need to have a basic understanding of a UML use case diagram. For further help, they will need to refer to an online description. (See 3.9 sources)

- Implementation methods: Using the views, developers shall make sure to understand the proposed implementation and make sure they are familiar with the various technologies and hardware components being used. Then, they can start working on completing the system.

## 3.8 Correspondence rules

The package diagram corresponds to the Realisation viewpoint because it highlights the organization of elements into related groups. This offers a general overview of the system and how it should be organized, providing structure to the codebase.

The deployment diagram corresponds to the Realisation viewpoint because it highlights the physical and software components required for the system to be used by external users once completed.

## 3.9 Sources

[1] Lynch, Warren. "UML: What Is Package Diagram? How to Use It?" *Medium*, Medium, 27 Mar. 2019, warren2lynch.medium.com/uml-what-is-package-diagram-how-to-use-it-dbd317c07d5d.

[2] "Deployment Diagram Tutorial." *Lucidchart*, www.lucidchart.com/pages/uml-deployment-diagram.

# 4 Views+

Much of the material in an AD is presented through its architectural views. Each view follows the conventions of its governing viewpoint. A view is made up of architecture models.

## 4.1 View: Logical View

The logical view describes the main parts that comprise the system and their respective interactions

The logical view of the ERP system will be composed of three important layers: The views, the models, and the controllers.

The views will provide graphical interfaces for the user to interact with the ERP system. All the classes regarding UI components will be contained in this package.

The controllers will allow the communication between the user and the data they are requesting.

The models will provide access to the data requested by the user and allow the system to be updated in real-time.

### 4.1.1 Models+

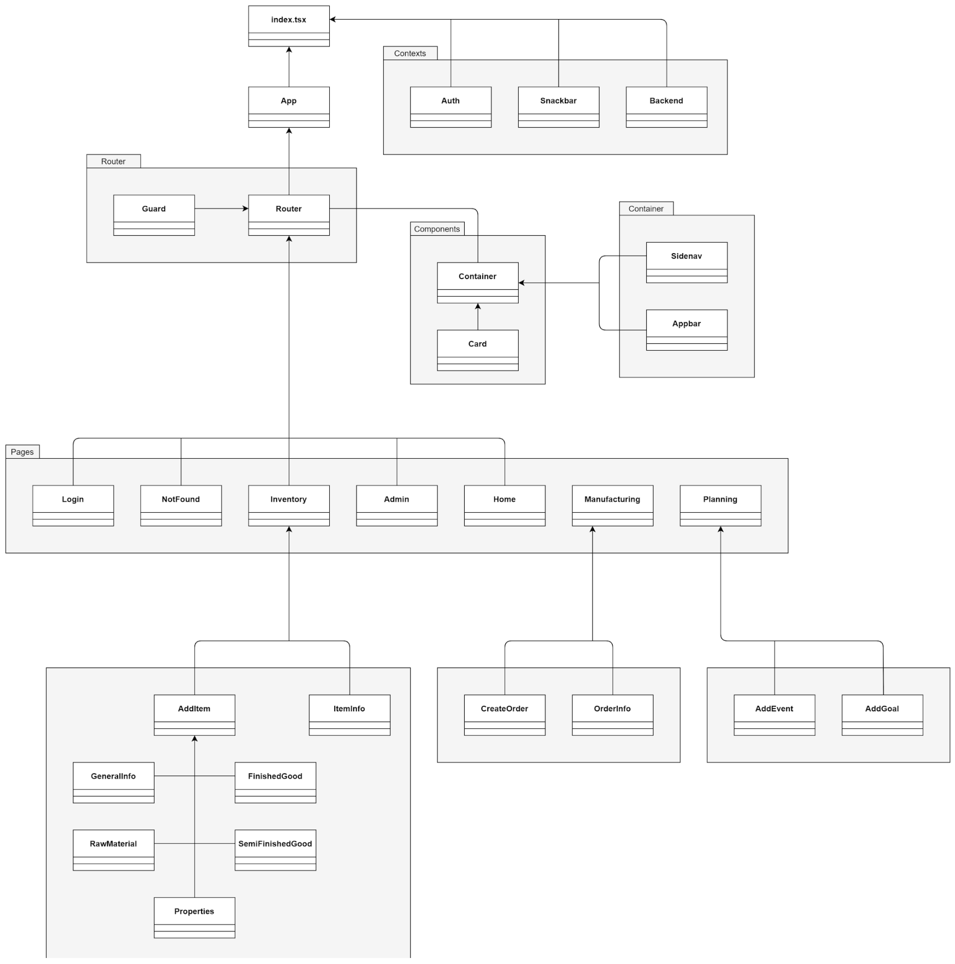
### 4.1.2 User interfaces

The user interfaces will be essential to allow the user to perform their desired actions. These will include the following:

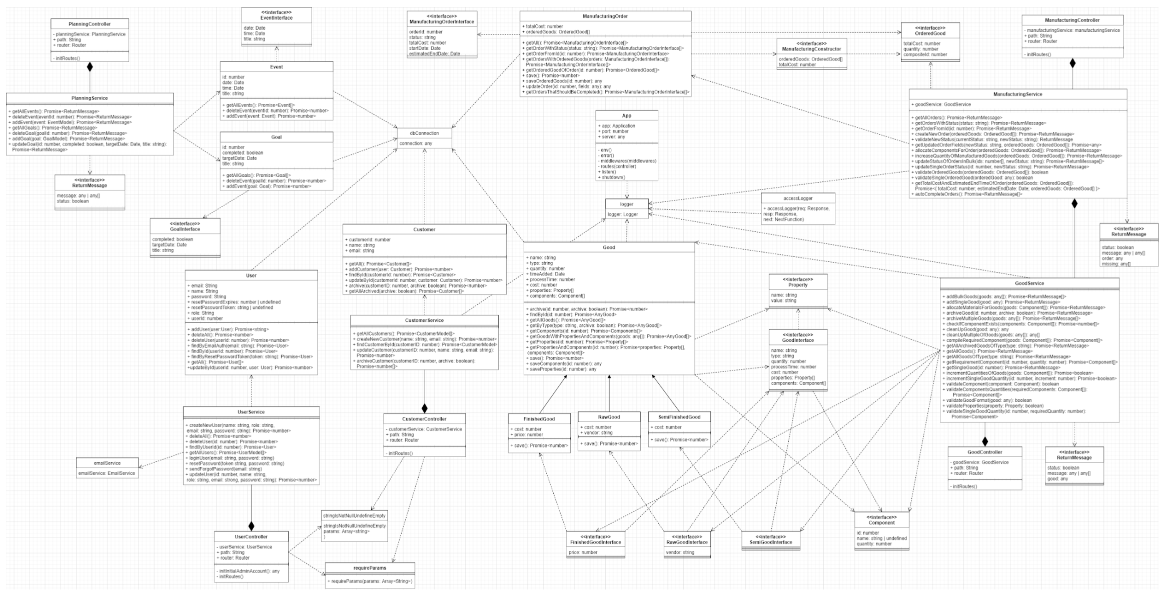
| **User interface** | **Description** |
|---|---|
| Login | User can log in to the ERP system by entering their email and password |
| Navigation bar | Users can see the various sections of the ERP system and can select their desire section. |
| Products | - Can add, modify, delete raw materials<br>- Can add, modify, delete semi-finished goods<br>- Can add, modify, delete finished goods |
| Inventory | User can see the available quantity of |

| | |
|---|---|
| | raw materials, semi-finished goods, and finished goods |
| Manufacturing | Users can create orders for buying raw materials or creating semi-finished goods and finished goods. |
| Sales | - User can sell bikes to customers<br>- User can observe the cost and profit of orders |
| Accounting | User can observe the incoming and outgoing balance transactions |
| Planning | User can indicate their goals:<br>- Average bike production cost<br>- The target number of sales<br>- Target profit |
| Scheduling | User can see the schedule of machines:<br>- Uptime<br>- Downtime<br>- Maintenance |

## Frontend Class Diagram:



Backend Class Diagram:



### 4.1.3 Known Issues with View

Since the logical view represents abstract objects, their implementation may sometimes be inconsistent with their description.

A change in requirements can impact some components of the logical view which can lead to cost and schedule overruns.

Users may agree with the functionalities proposed by the logical view but disapprove of them once the implementation is complete.

## 4.2 View: Process View

The process view describes the system's processes and the communication between these processes, it illustrates the decomposition of the system. It takes a deeper look at what should happen within the system.

It takes into account the processes and thread involved in the runtime of the system

This view primarily considers all the non-functional requirements of the system, including performance, resilience, usability, and many more.

### 4.2.1 Models+

### 4.2.2 Processes

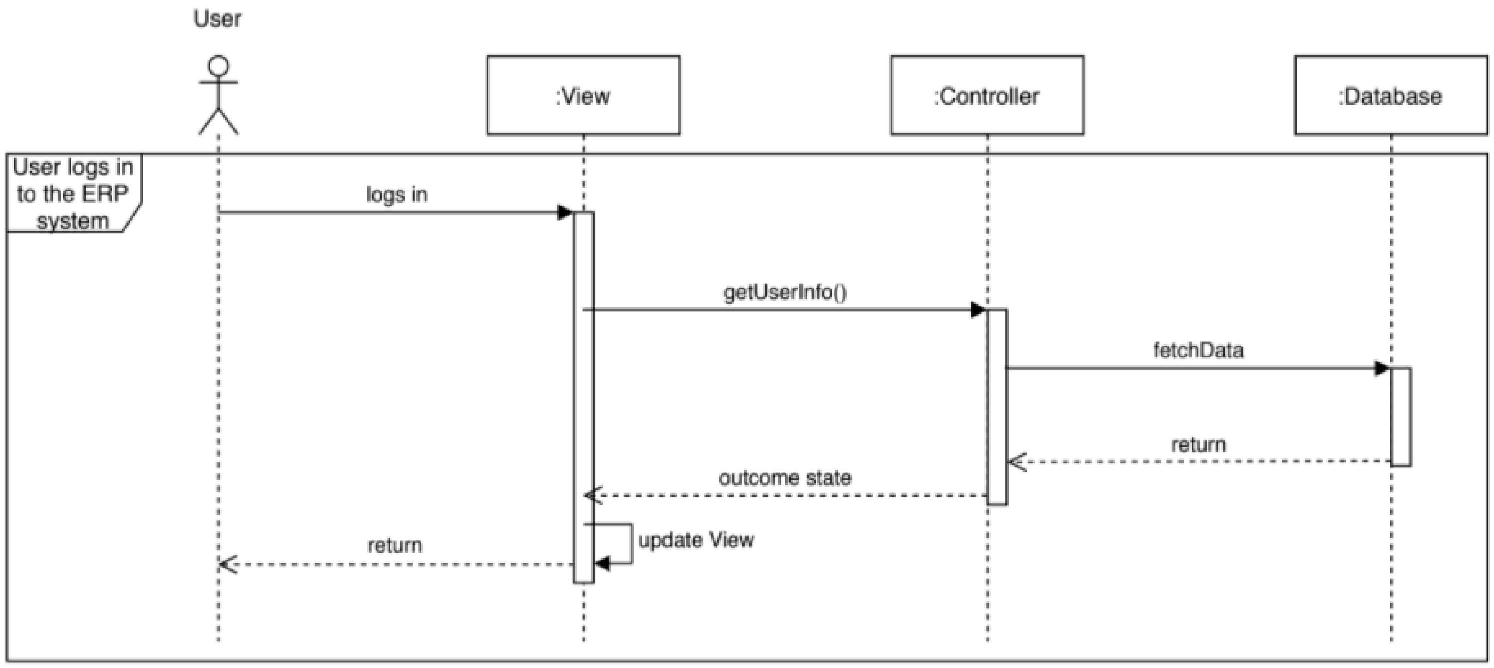


Figure X: Interactions between the ERP system components and user when interacting with the view

<u>Users logs in to the ERP System</u>

1) The user enters his credentials on the view.
2) The View sends requests to the controller to check the user credentials
3) The controller analyzes the user credentials in the database.
4) The database sends a confirmation to the controller, that sends the same message to the view.
5) The view displays the confirmation to the user by updating itself

### 4.2.3 Known Issues with View

Can be difficult to predict the impact that the various processes and threads will have on the system in terms of transaction speeds, throughput, and buffering.

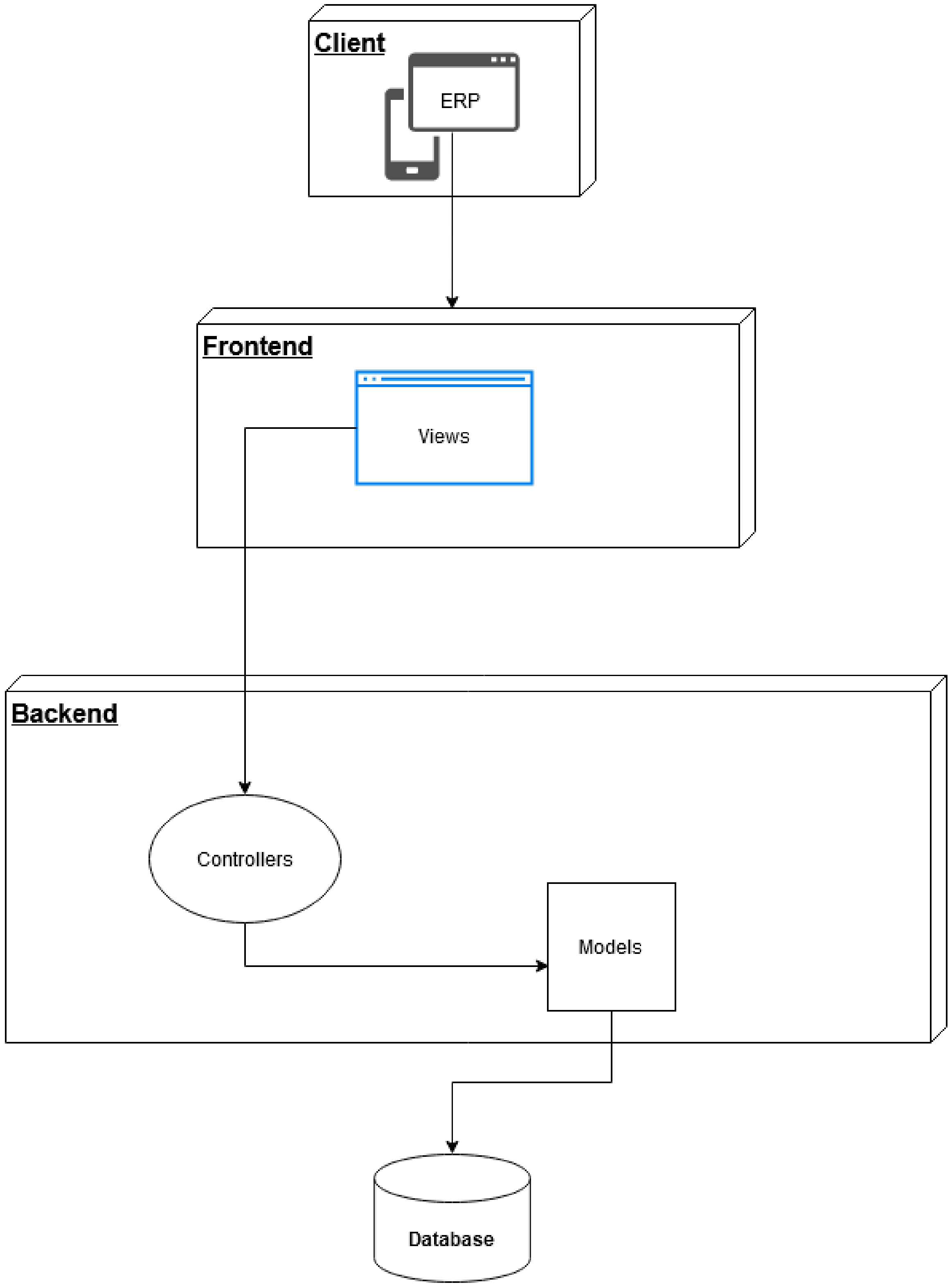
Can be difficult to determine the scalability and performance of the runtime environment without any implementation or real-time system to provide user feedback.

## 4.3 View: Developpement View

The development view of the ERP system describes the organization of the source code and the software development environment.

### 4.3.1 Models+

### 4.3.2 Overall Implementation (block diagram)



### 4.3.3 Known Issues with View

In order to implement the development view proposed above, the developers must structure their code accordingly and avoid confusing a controller from a model as this can lead to extra maintainability and refactoring in the future.

The usage of the development view proposed above increases the number of requests and event-driven actions on the system which can cause slowdowns for clients and make it hard for developers to debug.

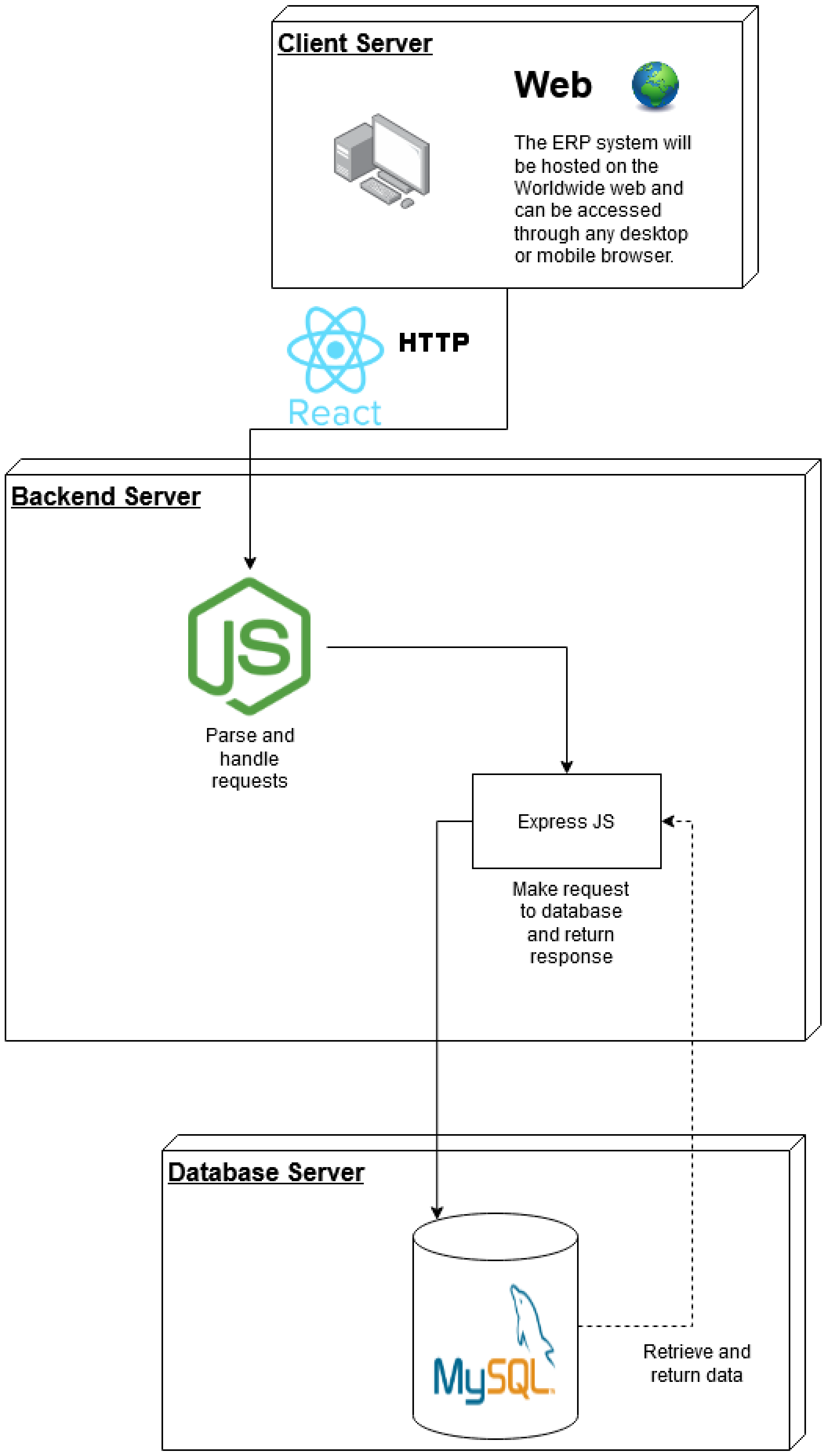
Even though the models, views, and controllers can be seen as independent components, the developers must make sure that by upgrading or modifying one component, the other components remain accessible and provide the desired output.

## 4.4 View: Physical View

The physical view describes the ERP system's execution environment. It highlights the mapping of software to its physical components and the physical network configuration between nodes.

### 4.4.1 Models+

### 4.4.2 Deployment Diagram


Client Server
Web
The ERP system will be hosted on the Worldwide web and can be accessed through any desktop or mobile browser.
React
HTTP
Backend Server
Parse and handle requests
Express JS
Make request to database and return response
Database Server
MySQL
Retrieve and return data

### 4.4.3 Known Issues with View

One of the issues with this view is that it does not include the security of components. In fact, firewalls, encryption, and threat protection are key to the sustainability of the system.

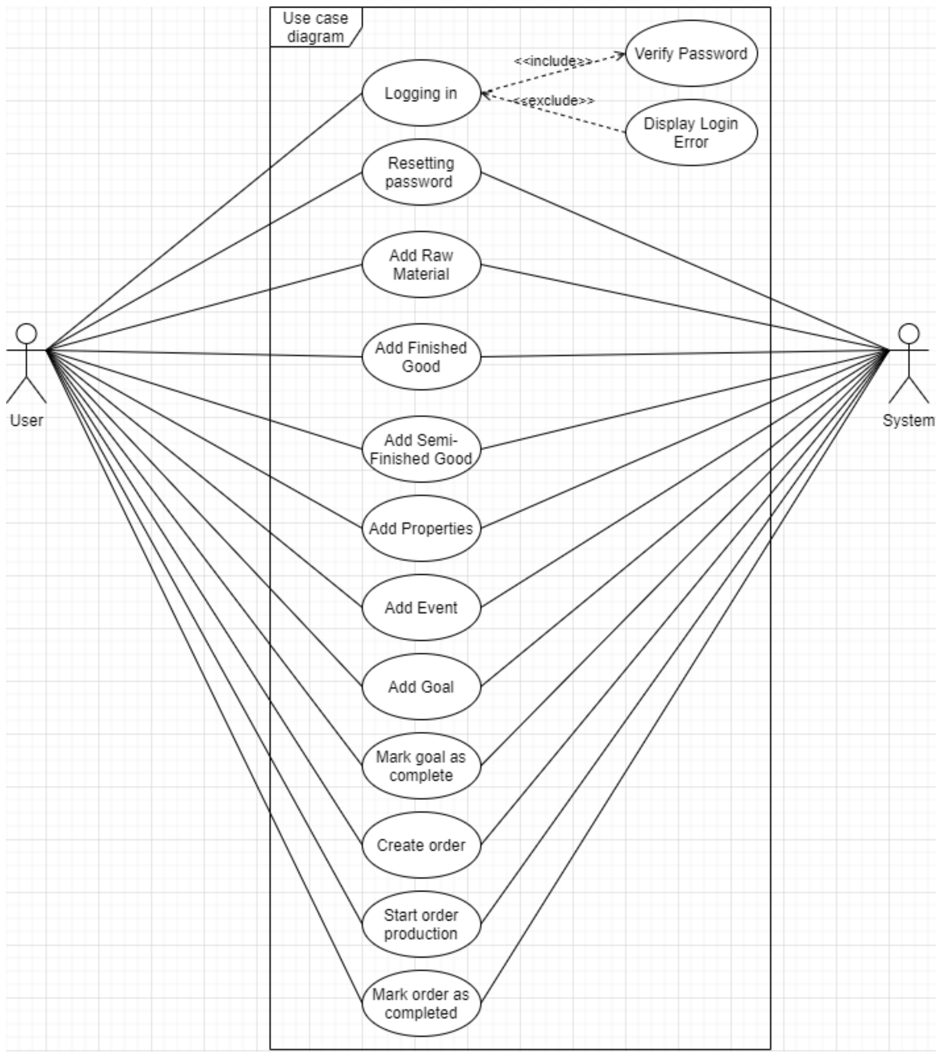
Furthermore, this view does not highlight the storage of logs in case of failures and crashes.

## 4.5 View: Use-case View

The main goal of the use-case view is to capture the dynamic aspects of a system. It displays the system functionality on how the users see them. It explains the system behavior and the functionality that he provides.

### 4.5.1 Models+

### 4.5.2 Use-case Diagram



### 4.5.3 Known Issues with View

One of the issues with this view is that it is not object-oriented. It can lead to multiple problems such as a decomposition of the system in terms of related use cases. Also, we do not know when to stop while trying to read the use-case diagram.

Also, the use case view lacks formal definitions of the actors and the use case terms. We do not always have more information about the actors.

# 5 Consistency and correspondences

This chapter describes consistency requirements, recording of known inconsistencies in an AD, and the use and documentation of correspondences and correspondence rules.

## 5.1 Known inconsistencies

Currently, the AD does not contain any inconsistencies. However, potential inconsistencies may arise if changes in the requirements occur. In fact, this can lead to inconsistencies in the architecture, implementation as well as deployment.

## 5.2 Correspondences in the AD

The AD is composed of three viewpoints: The Requirements viewpoint, the design viewpoint, and the realization viewpoint.

The requirements viewpoint corresponds to the use case view (section 4.5) and its associated stakeholders and concerns which can be seen in section 3.1.

The design viewpoint corresponds to the logical view and process view (section 4.1 and 4.2). Their respective stakeholders and concerns can be seen in section 3.1.

The realization viewpoint corresponds to the development view and physical view (section 4.3 and 4.4). Their respective stakeholders and concerns can be seen in section 3.1.

## 5.3 Correspondence rules

To avoid schedule and cost overruns as well as future maintainability and refactoring issues, it is important that the stakeholders concerned by each viewpoint follow the corresponding views in section 4. This will ensure that the MVC architecture is properly implemented within the ERP system and that various phases of the project will be completed successfully.

# Bibliography

[1] *Use Cases: the Pros and Cons*,
www.cs.hmc.edu/~mike/courses/mike121/readings/reqsModeling/firesmith.htm.

[2] “Viewpoints.” *Software Systems Architecture Viewpoints Comments*,
www.viewpoints-and-perspectives.info/home/viewpoints/.

[3] “Informit.” InformIT, www.informit.com/articles/article.aspx?p=2461675&amp;seqNum=4.