

# **Vendomatic Project**

## **Architecture Notebook**

*Version 0.3*

**Prepared By:** Code Busters

## Revision History

Version	Description	Author	Date
0.1	<ul style="list-style-type: none"><li>Draft</li></ul>	Kadir Kılıçoğlu	2023-04-12
0.2	<ul style="list-style-type: none"><li>Architecturally Significant Requirements are added.</li><li>Architectural Mechanisms are added.</li><li>Domain model is added under Section-7 Key Abstractions</li></ul>	Tuğçe Sözer	2023-04-23
0.3	<ul style="list-style-type: none"><li>Review and format</li></ul>	Ezgi Özkan	2023-04-24

---

## Table of Contents

Revision History	2
Table of Contents	3
<b>1. Purpose</b>	<b>4</b>
<b>2. Architectural Goals and Philosophy</b>	<b>4</b>
<b>3. Assumptions and Dependencies</b>	<b>5</b>
<b>4. Architecturally Significant Requirements</b>	<b>5</b>
<b>5. Decisions, Constraints and Justifications</b>	<b>5</b>
<b>6. Architectural Mechanisms</b>	<b>6</b>
6.1. Architectural Mechanism 1- Publish-Subscribe Style	6
6.2. Architectural Mechanism 2- Availability	6
6.3. Architectural Mechanism 3- Modifiability	6
6.4. Architectural Mechanism 4- Performance	7
6.5. Architectural Mechanism 5- Security	7
<b>7. Key Abstractions</b>	<b>7</b>
<b>8. Architectural Framework</b>	<b>7</b>
<b>9. Architectural Views</b>	<b>8</b>
9.1. Context	8
9.2. Deployment	9

## 1. Purpose

This document describes the architectural philosophy, decisions, constraints, justifications, significant elements, and overarching aspects of the Vendomatic system, a vending machine management and customer interaction platform that includes a web application for vending machine owners/operators and a mobile application for customers.

The architecture notebook will serve as a reference for the development team throughout the development process, helping to ensure that the application is designed and implemented according to best practices and industry standards. Additionally, it will provide a foundation for future development and maintenance, enabling the platform to adapt and evolve as business needs and technology requirements change over time.

## 2. Architectural Goals and Philosophy

The architectural goals and philosophy of the Vendomatic system are centered around addressing specific issues and considerations that drive the design decisions.

**Scalability:** The architecture must be designed to handle increasing user loads efficiently, ensuring the system remains performant even during peak usage periods. This includes considering the deployment of services across multiple servers or using cloud-based infrastructure to accommodate growth.

To address scalability concerns, Vendomatic backend consisting of the database and the API for both mobile and web applications should be hosted on the cloud. Though, the services should be hosted on a single server until reaching a broader audience to ease faster deployment during Elaboration and Construction phases.

**Integration:** The architecture should be built with a focus on integrating seamlessly with external systems and services, such as payment gateways, social media login providers, and third-party APIs. This will enable the system to leverage existing technologies and simplify the implementation of new features.

**Security:** The architecture must prioritize the protection of sensitive user data and transactions. This includes employing secure communication channels, proper data encryption, and adhering to best practices for handling user authentication and authorization.

**Maintainability:** The architecture should be designed for long-term maintenance, ensuring that the system can be easily updated and extended as business requirements evolve. This includes the use of well-defined interfaces, clean code practices, and comprehensive documentation to support future development efforts.

**Platform Independence:** The architecture should minimize hardware dependencies and enable the system to run on different platforms and environments, ensuring that it remains flexible and adaptable to various deployment scenarios.

By addressing these critical concerns, the architecture aims to provide a solid foundation for the Vendomatic system, ensuring that it meets the demands of its users while remaining adaptable to future changes and enhancements.

### 3. Assumptions and Dependencies

- 3.1. The Vendomatic Web Application and Mobile Application will communicate with a centralized backend.
- 3.2. The vending machines will be equipped with the Vendomatic adapter for remote management.
- 3.3. The system depends on the user's device providing accurate location data to enable features such as searching for nearby vending machines or filtering based on proximity. This dependency requires the user to grant the necessary location permissions and assumes that their device's GPS functionality is functioning correctly.
- 3.4. The development team is familiar with modern web and mobile development technologies.
- 3.5. The system will rely on third-party services for payment processing, map integration, and social authentication.
- 3.6. The system assumes that users have access to a reliable internet connection to interact with the Vendomatic Mobile Application and access the various features and functionalities. This dependency may affect the system's performance in areas with limited or unstable network connectivity, necessitating the implementation of offline capabilities or caching mechanisms where possible.
- 3.7. The system assumes that it complies with any applicable regulations and industry standards, such as data protection and privacy laws.

### 4. Architecturally Significant Requirements

1. Secure user authentication and role-based access control for different user types shall be provided (vending machine owners, operators, customers).
2. Owners shall be registered during the setup phase, and a single Vendomatic Software on the web platform shall have a single owner.
3. Social logins shall be supported only by the Vendomatic Mobile Application.
4. Username shall be the email of the user.
5. Passwords shall be at least 8 characters containing at least one letter and one number.
6. All inputs shall be both client-side and server-side validated.
7. Access to the system shall use OAuth 2.0, OIDC supported JWT token and Firebase as IdAAS to authenticate all users.
8. No password will be kept by the system, hence passwords won't be tracked.

### 5. Decisions, Constraints and Justifications

1	Decision	Use a modular monolith architecture to enable scalability, maintainability, and flexibility.
	Justification	<p>The Vendomatic system will have multiple components with different responsibilities, and a modular monolith architecture will allow them to evolve independently while supporting a less complex deployment strategy.</p> <p>One alternative is the pure microservice architecture, but considering its deployment complexity and lack of distributed teams, we eliminated this option.</p>

		Another alternative, the vanilla monolith, is hard to maintain and doesn't fit well into iterative life cycles, thus we eliminated this option as well..
2	<b>Constraint</b>	All data transmitted between the web/mobile applications and backend services must be encrypted.
	<b>Justification</b>	Ensuring the privacy and security of user data is a priority.  Though encryption slightly degrades the performance, the tradeoff of using unencrypted traffic may have a huge impact by resulting in a vulnerable model. Hence, encryption is preferred over performance.
3	<b>Decision</b>	Use a responsive design for the web application.
	<b>Justification</b>	This ensures that the web application will be usable on various devices and screen sizes.  Responsive design has become the mainstream recently. Working with a strict layout may be easier but since there are a wide variety of resolutions and screen types, we skipped this option.
4	<b>Constraint</b>	The mobile application must be developed for both Android and iOS platforms.
	<b>Justification</b>	To reach a wider customer base.  Otherwise, with a single platform target, Vendomatic wouldn't reach millions of users of the ignored platform.

## 6. Architectural Mechanisms

### 6.1. Architectural Mechanism 1- Publish-Subscribe Style

A message queue (RabbitMQ) will be used to relay messages from vending machines to subscribers to manage system state which yields a highly efficient one-way distribution of information with very low-coupling of components.

### 6.2. Architectural Mechanism 2- Availability

Monitoring the system to detect faults and rollback to revert the system to a previously known good state to recover from faults will be used as availability tactics.

### 6.3. Architectural Mechanism 3- Modifiability

Reducing size of each module, increasing cohesion and reducing coupling will be used as modifiability tactics.

## 6.4. Architectural Mechanism 4- Performance

Limiting event responses and bounding execution times will be used as performance tactics.

## 6.5. Architectural Mechanism 5- Security

Actor authentication and authorization will be used to resist possible attacks. Also, limiting access to the system if there are repeated failed attempts to access will be used as a security tactic.

## 7. Key Abstractions

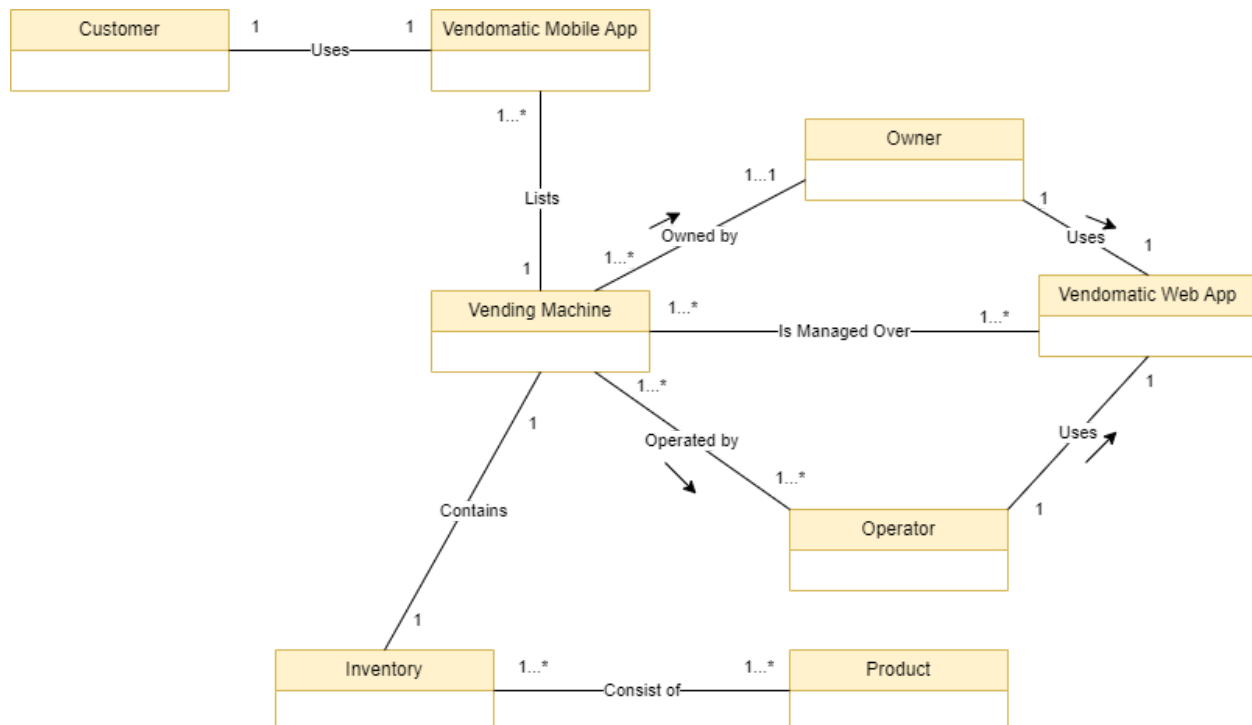
7.1. **Vending Machine:** Represents a physical vending machine in the system.

7.2. **Product:** Represents a product available in a vending machine.

7.3. **User:** Represents a customer, vending machine owner, or operator in the system.

7.4. **Maintenance Task:** Represents a maintenance activity assigned to an operator.

7.5. **Rating:** Represents a user's rating for a vending machine or product.



## 8. Architectural Framework

A layered architecture will be used, consisting of the following layers:

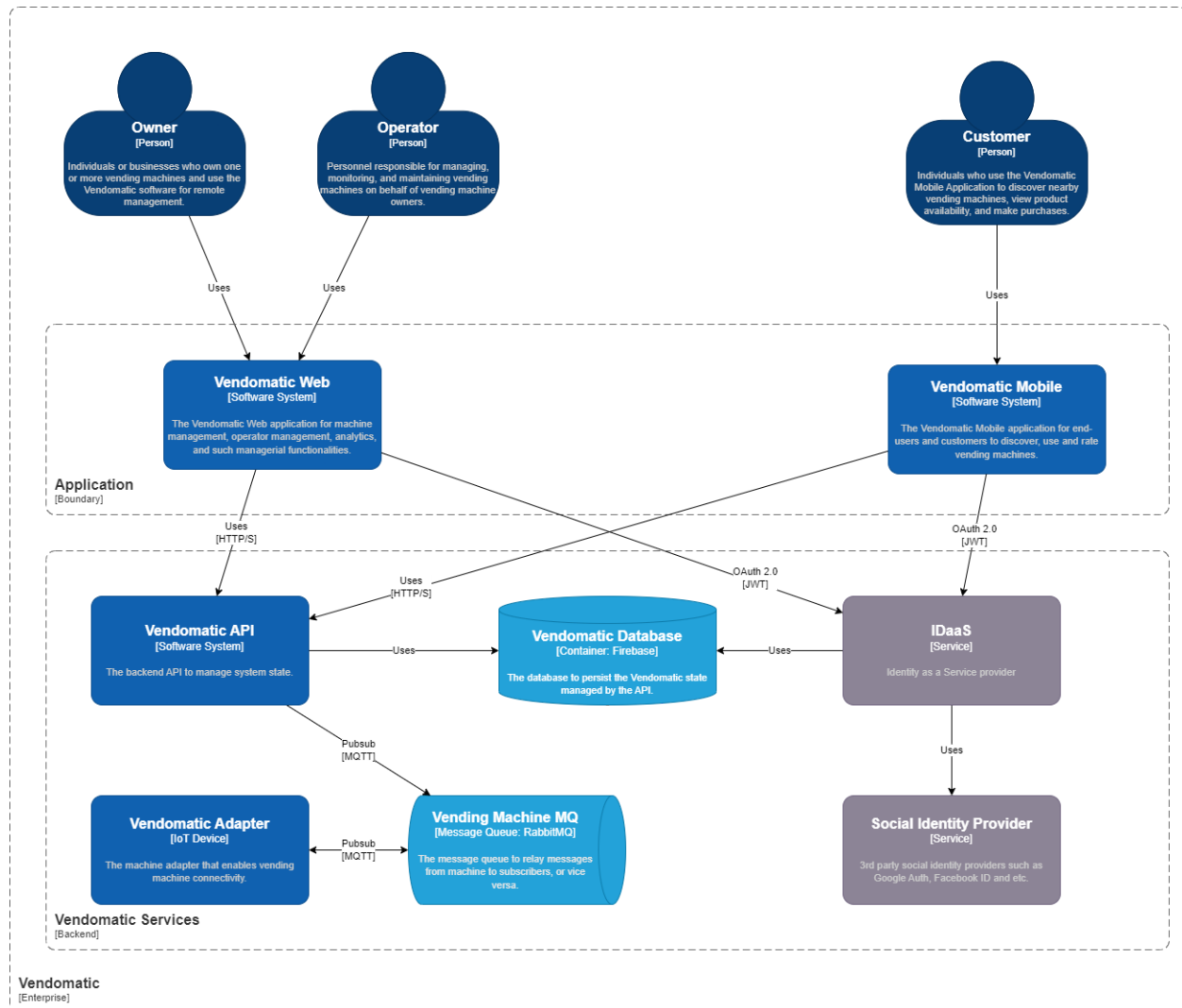
- **Presentation:** Web Application (for owners/operators) and Mobile Application (for customers).
- **Application:** Backend services responsible for business logic and coordination between other layers.
- **Domain:** Core domain model and business logic.
- **Data Access:** Responsible for storage and retrieval of data, including database access

and caching.

- **Integration:** Facilitates communication with external systems, such as payment processors and map services.

## 9. Architectural Views

### 9.1. Context





## 9.2. Deployment

