INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Department of Electronical Engineering, Telecommunications and Computers

Content Management System with Identity Verification and Non-Repudiation Support

Márcia Krus Policarpo

(Bachelor's Degree in Informatics and Multimedia)

Final Project to obtain Master's Degree
in Informatics and Multimedia Engineering

Advisor : Professor Doutor Paulo Trigo

Jury:

President: Professor Doutor Carlos Gonçalves

Members: Professor Doutor António Teófilo
Professor Doutor Paulo Trigo

October, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Department of Electronical Engineering, Telecommunications and Computers

Content Management System with Identity Verification and Non-Repudiation Support

Márcia Krus Policarpo

(Bachelor's Degree in Informatics and Multimedia)

Final Project to obtain Master's Degree
in Informatics and Multimedia Engineering

Advisor : Professor Doutor Paulo Trigo

Jury:

President: Professor Doutor Carlos Gonçalves

Members: Professor Doutor António Teófilo
Professor Doutor Paulo Trigo

October, 2023

Ever tried. Ever failed. No matter.
Try again. Fail again. Fail better.

Samuel Becket, Worstward Ho, 1983

Acknowledgments

I would like to express my heartfelt gratitude to Professor Paulo Trigo, whose unwavering guidance and expertise have been invaluable throughout this research journey. His mentorship, patience, and dedication to academic excellence have been crucial in shaping this project.

I am also deeply thankful to Professor Paulo Vieira and Professor Luís Faria for their valuable contributions, insightful feedback, and the time they generously devoted to advising and supporting my work.

Beyond the academic realm, I want to extend my appreciation to my mother, whose unwavering encouragement and belief in my abilities have been a constant source of strength. Your boundless love and support have been my foundation.

To my friends, thank you for being there through the challenges and triumphs, for lending an ear, and for the countless moments of inspiration and camaraderie. Your friendship has enriched this academic endeavor in ways beyond measure.

This project represents the culmination of a collective effort, and I am grateful to each person who has played a part in its realization.

Abstract

Every historical event is perceived in different ways by different people and each person involved has their own testimony of what happened. Traditional social networks and wiki platforms lack the space for personal testimonies. Wikis limit the ability to showcase diverse viewpoints by offering a single-page unified vision. Social networks, characterized by anonymity and ephemeral content, often prioritize the latest trends over preserving individual narratives.

To bridge the gap, a knowledge-sharing platform, named ADDA, was developed to commemorate the historical significance of Portugal's 25th of April. It stands as a testament to the memories of those who experienced this moment in history. The platform is currently publicly available with a subset of its planned features, with ongoing plans for gradual expansion. The ADDA platform aims to balance pseudonymity and identity verification, fostering trust through verification while allowing users the freedom to openly share stories under a pseudonym. This blend promotes responsible sharing, linking accountability with expressive freedom.

Digital signatures have been incorporated into the platform to safeguard the authenticity and integrity of each shared testimony (and all of its versions). These signatures serve as a seal of trust, ensuring that the contributions are tamperproof and verifiable.

Performance tests were conducted to compare encryption methods, allowing the establishment of criteria and the identification of differences for an informed decision on the method to be used. Additionally, usability tests yielded positive feedback and valuable user suggestions. All of this contributes to the ongoing platform enhancements and reinforces the sense of need for a social interaction platform with these characteristics.

Keywords: Knowledge-sharing Platform, Historical Events, Non-Repudiation, Digital Signatures, Identity Verification, Historical Understanding

Resumo

Cada evento histórico é vivenciado de diferentes formas por pessoas diferentes, e cada pessoa envolvida possui seu próprio testemunho do que ocorreu. Redes sociais e plataformas wiki carecem de espaço para testemunhos pessoais. As wikis, com a sua versão unificada em uma única página, limitam a capacidade de apresentar perspectivas diferentes. Redes sociais, caracterizadas pela anonimidade e conteúdo efémero, frequentemente dão prioridade às últimas tendências em detrimento da preservação de narrativas individuais.

Para preencher essa lacuna, foi desenvolvida uma plataforma de partilha de conhecimento, designada por ADDA. Esta plataforma, criada para comemorar o 25 de abril em Portugal, serve de repositório para as memórias daqueles que vivenciaram esse momento na história. A plataforma está atualmente disponível ao público, apenas com parte das suas funcionalidade. O objetivo da plataforma é o de equilibrar a pseudonimização e a verificação de identidade, de forma a promover confiança por meio da verificação, mas simultaneamente permitir aos utilizadores partilharem histórias abertamente sob um pseudónimo. As assinaturas digitais, também, foram incorporadas na plataforma para proteger a autenticidade e integridade de cada testemunho partilhado (e todas as suas versões).

Foram realizados testes de desempenho para comparar métodos de encriptação, estabelecendo critérios e identificando diferenças para uma decisão informada sobre o melhor método a usar. Os testes de usabilidade revelaram um resultado positivo e trouxeram sugestões dos utilizadores que contribuíram para a melhoria contínua da plataforma.

Palavras-chave: Plataforma de Partilha de Conhecimento, Eventos Históricos, Não Repúdio, Assinaturas Digitais, Verificação de Identidade, Compreensão Histórica

Contents

List of Figures

List of Tables

Acronyms

2FA Two-Factor Authentication. 18

ADDA Antes, Durante e Depois de Abril. 2, 10, 11, 12, 13, 22, 25, 37, 54, 57, 65, i
API Application Programming Interface. 26, 49, 62
ASiC Associated Signature Containers. 25

CMS Content Management System. 19, 20, 21, 22, 41, 42, 58

DBMS Database Management System. 21, 22, 54, 57
DoS Denial of Service. 19, 65
DSA Digital Signature Algorithm. 15, 16

ECDSA Elliptic Curve Digital Signature Algorithm. 15, 16, 25, 53, 59, 61, 62, 64, 65
ER Entity-Relationship. 54
EUF-CMA Existential Unforgeability under Chosen-Message Attack. 15

HCI HyperConverged Infrastructure. 25, 50
HTML HyperText Markup Language. 22
HTTP Hypertext Transfer Protocol. 36, 53, 62

IDE Integrated Development Environment. 4
IoT Internet of Things. 16

KMS Knowledge Management System. 9

mo Machine Object. 48, 49
MTV Model-Template-View. 22, 23, 42
MVC Model-View-Controller. 22

ORM Object-Relational Mapping. 42
OS Operating System. 50

po Portable Object. 48
PSS Probabilistic Signature Scheme. 15

RBAC Role-Based Access Control. 19
RDBMS Relational Database Management System. 21, 54
RSA Rivest-Shamir-Adleman. 14, 15, 16, 25, 53, 59, 61, 62, 64, 65

SDK Software Development Kit. 24, 25, 26, 49
SMTP Simple Mail Transfer Protocol. 26

UC Use Case. 33, 34, 35, 42, ix, x, xi, xii
UML Unified Modeling Language. 37
URL Uniform Resource Locator. 23, 47, 48

VSCode Visual Studio Code. 4

WYSIWYG What You See Is What You Get. 21, 66

XAdES XML Advanced Electronic Signatures. 24, 25

1

Introduction

This chapter explains what inspired this project and its core objectives. It also highlights the key contributions and impacts of the project. It gives insights into the tools and workflow used during development, and provides an outline of what to expect in the rest of this document.

1.1 Motivation

In today's digital age, the need for a platform where individuals can share their perspectives, knowledge and testimonies has become more pronounced than ever. Traditional social network platforms may allow for sharing thoughts and experiences, but they prioritize the latest trends over the preservation of collective memory. On the other hand, while encyclopedias and knowledge repositories provide valuable information, they often lack the individuality and subjective experiences that can be crucial for a comprehensive understanding of the various perspectives taken from the same event. This limitation stems from the conventional wiki practice of preserving a single authoritative version of an article, where new editions replace the prior content.

This project aims to bridge the gap between a conventional encyclopedia and a social network by developing a platform that fosters collaboration and diverse viewpoints while ensuring data integrity and user identity verification.

1.2 Objective

This project is driven by the goal of creating a custom platform specifically dedicated to the events of the 25th of April in Portugal, aligned with the commemoration of its 50th anniversary [2].

The 25th of April holds significant historical importance in Portugal as it marks the Carnation Revolution of 1974, a pivotal moment in the nation's history. This revolution led to the overthrow of the authoritarian "Estado Novo" regime and the establishment of a democratic government. The event was characterized by the peaceful military coup, symbolized by the placement of carnations in the barrels of soldiers' rifles.[2]

One of the notable contexts of the 25th of April is its connection to the Colonial War, a series of conflicts and colonial wars fought by Portugal in its African overseas territories. The Carnation Revolution not only brought an end to the dictatorship but also signaled the start of decolonization and the eventual independence of these territories. [2]

Over time, the memories and stories associated with the 25th of April have become increasingly valuable, serving as a testament to the resilience and determination of the Portuguese people in their quest for freedom, democracy and the end of colonialism. However, as the years pass, these precious memories risk fading away, lost to the passage of time. The created platform, from now on referred as Antes, Durante e Depois de Abril (ADDA), serves as a digital archive and repository for those memories.

The project is specifically directed towards those who lived through the period of the 25th of April (more than 2.5 million people), offering them the means to commemorate it by memories and thoughts about how it changed their lives and the life of the country. In this sense, the ADDA platform aims to be a valuable resource for society at large, generating a comprehensive collection of sources about this historical period.

By providing a space for people to share their recollections, photographs, documents and personal stories, there's a hope to ensure that the memories of the 25th of April continue to resonate with current and future generations.

1.3 Contributions

The primary contributions of this project encompass both the implementation and deploy of the platform for public use and the submission of the article titled "Personal

Testimony Sharing with Identity Verification and Non-Repudiation Support" to the ICISSP 2024 conference [8].

The key features introduced by the platform include:

- **Collaborative Article Creation and Editing -** Interface that enables collaborative article creation and editing, facilitating the incorporation of diverse perspectives on various subjects.
- **Data Integrity, Authentication, and Non-Repudiation -** Mechanisms that ensure the content integrity of articles. Utilizing cryptographic hashing and digital signatures, contributions are authenticated and equipped with a verified tag, maintaining the credibility of the data.
- **Identity Verification -** Identity verification system that allows users to establish authenticity while safeguarding privacy, creating a trusted environment for knowledge-sharing.
- **Multilingual Support -** Multilingual feature that empowers users to access and contribute to content in their preferred language. This feature promotes accessibility for a global audience.
- **Revision History and Validation -** An articles' revision history that offers users a transparent overview of changes over time. The development of a validation process that ensures the authenticity and integrity of each article revision.
- **User-Friendly Interface -** An intuitive and visually appealing interface that encourages user engagement and facilitates smooth navigation across the platform.
- **Scalability and Future Extensibility -** A modular and scalable architecture that allows for seamless incorporation of potential future enhancements and additions without significant structural alterations.

1.4 Development Tools and Workflow

In the course of the project's development, a combination of tools and technologies were employed to facilitate a seamless workflow and efficient collaboration. These were the key components that shaped the development process:

- **Version Control and Development Environment -** To ensure effective version control and collaborative coding, Git in conjunction with GitHub was used (Attachment A). This combination allowed to manage the codebase efficiently and coordinate the development efforts. Python [34] was selected as the programming language and the capabilities of Visual Studio Code (VSCode) [44] were harnessed as the preferred Integrated Development Environment (IDE).
- **Task Management -** To organize work and track progress, Trello [40] was used. Trello served as the primary task management tool, offering an intuitive interface for creating, assigning and monitoring tasks. It served as a base foundation for project management.
- **Team Communication and Collaboration -** Effective communication was essential throughout the project. To facilitate collaboration and ensure that everyone was synchronized, periodic team meetings were held to discuss progress, share insights and address challenges.
- **Workflow Integration -** The seamless integration of these tools into the workflow enhanced the overall efficiency. Git and GitHub enabled version control and code synchronization, ensuring that team members could work concurrently without conflicts. Python and VSCode provided a powerful development environment, streamlining the coding process. Trello's task management capabilities helped maintain clarity and organization in the project tasks.

The development tools and workflow were selected during the early stage of this project to promote productivity, collaboration and project success. This combination of version control, task management and communication tools allowed the navigation through the complexities of this project while maintaining a structured and coordinated approach.

1.5 Document Outline

The document begins with the introduction and background knowledge, followed by an exploration of related work, the proposed solution, implementation details, evaluation, and concluding with a comprehensive summary and future directions. Therefore:

- **Chapter 1 - Introduction**: The current chapter depicts an overview of the project's motivations and contributions. The need for a knowledge-sharing platform is

discussed, and the main objectives of the platform are outlined, providing the context for the rest of the document.

- **Chapter 2 - Related Work**: The second chapter explores existing knowledge-sharing platforms. An analysis of wikis, social networks, and collaborative systems is conducted to understand their strengths and limitations.

- **Chapter 3 - Background Knowledge**: This chapter covers the essential concepts and principles that form the foundation of the platform. Topics such as encryption techniques, algorithms for secure communication, identity verification methods, and non-repudiation mechanisms are explained.

- **Chapter 4 - Proposed Solution**: In this chapter, various aspects, including use cases, system attributes and functions, main scenarios, technology choices and system architecture are discussed. This chapter aims to provide a comprehensive overview of how the platform functions and the features it offers to users.

- **Chapter 5 - Implementation**: This chapter covers the practical implementation of the platform. It outlines the key steps involved in creating the database and adding new endpoints and applications to the system. Additionally, it highlights the design patterns utilized during the implementation to ensure a robust and efficient codebase.

- **Chapter 6 - Evaluation**: The evaluation chapter focuses on assessing the performance and effectiveness of the platform. Various evaluation metrics and methodologies used to gauge the platform's success in meeting its objectives are presented.

- **Chapter 7: Conclusions** The concluding chapter encapsulates the key findings derived from this project. Additionally, there is a discussion of potential future developments and improvements that can further enhance the platform's capabilities.

Written Conventions Adopted:

- **Quotation Marks (" ") -** Used for quotations, prefixes, words in Portuguese and when referring to a specific word. For example: They answered "Yes."

- **Different Font Style (e.g., Formula Font) -** Applied to code components like names of classes, tables and methods.

- **Brackets (<>) -** Encloses buttons within brackets for differentiation.
- **Bold -** Reserved for giving emphasis and highlighting important information.

2

Related Work

This chapter explores existing knowledge-sharing platforms with a primary focus on essential concepts. Wikis, social networks, and collaborative systems are examined in order to develop a more efficient and clearly differentiated platform for information sharing. By thoroughly examining their strengths and limitations, the aim is to identify unique features and opportunities that represent the main contributions of this project when compared to these existing platforms.

2.1 Well-Known Social Networks

In this section, we present a concise overview of some prominent knowledge-sharing platforms and collaborative systems. These platforms have become integral parts of our digital lives, enabling information exchange and knowledge dissemination on a global scale.

2.1.1 Wikipedia

Wikipedia [47], a well-known collaborative encyclopedia, empowers users to create and edit articles across a vast array of subjects, emphasizing open collaboration and information sharing. However, it faces certain limitations.

As Sam Vaknin observes, 'Wikipedia is opaque and encourages recklessness' [42], primarily due to the prevalence of anonymous or pseudonymous contributors, which

hinders accountability for their actions. In the mid-2007, the Wikiscanner tool unveiled self-serving edits made by various entities, including government bodies and corporations [49]. It is interesting to note that the Wikiscanner tool was taken down as the founder, Virgil Griffith, said it was costing him "several thousand USD per month" [48].

Wikipedia's content, encompassing discussion and history pages, undergoes frequent edits and revisions, obscuring the true editorial process and raising concerns about identity, credibility, and the dynamic nature of information within the platform [42].

Another limitation lies in Wikipedia's format, where content exists as a single page with a unified version edited collectively by all contributors, lacking the ability to showcase individual perspectives and testimonies.

2.1.2 Facebook

Facebook [14], one of the most popular social networking platforms globally, serves as a hub for users to connect and engage in a wide array of activities. While its primary purpose is to facilitate social interactions, it also plays a role in information sharing. Users can create posts, share articles, and engage in discussions on various topics. However, it's important to note that much of the content on Facebook is ephemeral, often reflecting recent trends and superficial topics, with posts quickly fading from users' feeds over time. The lack of a robust identity verification system leads to the proliferation of anonymous profiles, resulting in a disconnection between content and its authors.

2.1.3 X

X [50] stands out as a microblogging platform that has revolutionized real-time information dissemination and trends. With its succinct format, X enables users to share short messages, links and multimedia content swiftly and efficiently. This platform's immediacy and ease of use make it a valuable tool for sharing breaking news, updates and trending topics. However, this emphasis on brevity, trending topics and superficial engagement does not align with the objective of sharing testimonies that contribute to the creation of a collective memory formed by diverse individual narratives. Moreover, similar to Facebook, it lacks a robust identity verification system, further disconnecting content from its authors and contributing to the transient and shallow nature of the platform's content.

2.2 Research Contributions

This section presents two key studies that have made substantial contributions to the fields of knowledge sharing and identity validation within social networks.

2.2.1 University Corporate Social Network

The research focused on the knowledge sharing with the university environment [9], it explores the design and implementation of a corporate social network tailored to the unique needs of a university's operational environment. This research focuses on leveraging social networking principles to enhance knowledge sharing among faculty, staff and students. By analyzing the challenges and opportunities within the university context, the study offers insights into creating effective knowledge-sharing platforms. The most significant findings of the study include a strong interest among university staff (82 percent) in utilizing a corporate social network for knowledge and operational information sharing. Additionally, the recommended architecture for the network involves a 3-Tier Architecture enriched with Knowledge Management System (KMS) concepts. This architecture facilitates effective data management, content categorization, and user identification, all of which align with user preferences.

2.2.2 Model for Enhancing Identity Validation

The study delved into a game theoretical model for identity validation [15], addressing challenges akin to our project, particularly in identity verification on social sites. The authors observed a prevalent issue, despite advanced authentication mechanisms little verification occurs during user registration, allowing for the easy provision of false information or the creation of fake accounts. The paper underscores the challenge of enabling users to maintain privacy while preventing misuse of profiles.

The authors conducted a formal analysis of user and server behavior concerning identity validation and trust issues by employing game theory techniques. Using mathematical abstractions, they explored multi-party decision-making in user-server interactions on social sites. The goal was to comprehend strategies adopted by social site providers (servers) and users for identity validation, with a focus on developing algorithms to address digital identity challenges. Their formulation of a two-player general sum game depicted the interaction between service providers and users. They chose the scenario with the most challenging situation in a two-player game, i.e., the players have complementary (joining the online community and accepting a new user) yet

competing goals (limit information disclosure vs. obtaining truthful data). Other scenarios, like when the user is willing to disclose as much information as needed to be validated, are less interesting from a design perspective.

The study uncovered that the optimal situation arises when there are well-defined privacy rules and clear pre-agreements in place, ensuring smooth cooperation between users and servers.

2.3 Comparison to Proposed Platform (ADDA)

While existing social networks and collaborative systems have their merits, they often fall short in providing a comprehensive solution for secure, reliable, and multi-perspective knowledge exchange. The ADDA platform aims to bridge these gaps and add significant value to the user experience. In Table 2.1 the key features and benefits of the proposed solution are compared with previously mentioned existing platforms and projects.

Table 2.1: Comparison of Knowledge-Sharing Platforms

Platform	**Data Integrity**	**Pseudonym Support**	**Non-Repudiation**	**Identity Verification**	**Diverse Perspectives**
Wikipedia	Low	No	No	Limited	No
Facebook	Low	Yes	No	Limited	No
X	Low	Yes	No	Limited	No
Corporate Social Network [9]	Low	Yes	No	N/S	No
Game Theoretical Model [15]	Low	No	No	Robust	No
ADDA	**High**	**Yes**	**Strong**	**Robust**	**Yes**

The key criteria are described below:

- **Data Integrity -** this gauges the platform's ability to maintain data accuracy and prevent unauthorized alterations;
- **Pseudonym Support -** this indicates whether users have the flexibility to use pseudonyms or alternate identities for privacy;
- **Non-Repudiation -** this measure assesses the platform's capability to prevent users from denying their actions or contributions;

- **Identity Verification -** this examines the platform's methods for confirming user identities, enhancing trust and accountability;
- **Diverse Perspectives -** this reflects the platform's capacity to encourage users to share unique viewpoints and personal testimonies.

When it comes to data integrity, Wikipedia, Facebook and X have limitations. They allow multiple users to edit content without robust verification, potentially exposing it to unauthorized alterations. In contrast, the proposed knowledge-sharing platform excels in data integrity. It employs cryptographic hashing, SHA-256 [38], to securely link each article revision to its corresponding hash. This ensures that any unauthorized changes to the content are easily detectable, preserving the integrity of the shared knowledge.

Pseudonym support varies across these platforms. Wikipedia typically requires users to employ their real names, lacking pseudonym support. In contrast, Facebook offers users the flexibility to use user-chosen names alongside their real identities, providing pseudonym support. X follows a similar approach to Facebook in supporting pseudonyms. The ADDA platform also supports pseudonyms, granting users the freedom to engage using pseudonyms or alternate identities while preserving privacy.

Non-repudiation measures are generally lacking in all platforms. Wikipedia, Facebook, X and both the studied projects, [9] and [15], do not provide strong non-repudiation mechanisms, making it challenging to prevent users from denying their contributions. In contrast, the ADDA platform excels in non-repudiation. It implements digital signatures, to prevent users from denying their contributions, ensuring accountability.

Identity verification methods vary among these platforms. Wikipedia offers basic user accounts with limited verification, often requiring email verification for account creation but lacking extensive identity verification measures. Facebook primarily focuses on real identities without rigorous verification. X encourages real identities, asking users to provide their real name, but without implementing robust verification. In contrast, the ADDA platform places a strong emphasis on identity verification during the user signup process. This approach enhances trust and accountability, creating a reliable environment for knowledge exchange.

Regarding diverse perspectives, Wikipedia primarily concentrates on encyclopedic content, which tends to prioritize a neutral and standardized tone over diverse personal viewpoints. Facebook is designed for social interactions, emphasizing individual opinions. X excels in real-time information dissemination but lacks a dedicated focus on diverse perspectives. In contrast, the ADDA platform stands out by encouraging diverse

viewpoints and personal testimonies. It fosters a wide-view by enabling users to share unique perspectives, enriching the knowledge-sharing experience.

By combining these essential features, the ADDA platform offers a unique value proposition to users seeking a secure, authentic and diverse space for sharing knowledge and experiences.

3

Background Knowledge

This chapter outlines the concepts which serve as a basis for this project, including data protection and knowledge authenticity mechanisms. Moreover, it also delves into explaining the range of different evaluated technologies and on the specific choices made, providing insight into the rationale behind their selection.

3.1 Concepts

This section provides essential explanations to grasp the project, particularly emphasizing on the critical components of non-repudiation and identity verification.

3.1.1 Non-Repudiation

In the digital context non-repudiation is a property which prevents an individual or entity from denying having performed a particular action related to data [20]. There are multiple mechanisms to enforce non-repudation, including biometric authentication, notaries and digital signatures.

Within the ADDA platform, non-repudiation ensures that once a user creates or modifies an article, he cannot later deny his authorship or contributions. The chosen approach was to implement digital signatures, which provides non-repudiation in a publicly verifiable manner [10].

3.1.2 Digital Signatures

In 1976, Diffie and Hellman introduced the concept of a digital signature as a solution for creating effective authenticated electronic communications that could be used as valid legal evidence in a courtroom setting [24]. Building upon this pioneering breakthrough, Ron Rivest, Adi Shamir, and Leonard Adleman advanced the field further in 1978 by devising the Rivest-Shamir-Adleman (RSA) asymmetric encryption method [35]. Symmetric encryption methods operate using a single shared key for both encryption and decryption, offering speed as an advantage. However, they require a secure connection for the initial exchange of this shared key between parties. In contrast, asymmetric encryption utilizes a pair of keys: a private key for encryption and a public key for decryption. This approach eliminates the need for a prior secure connection, making it more convenient for secure communication over untrusted networks. [27] This methodology provides the basis for digital signatures; utilizing the private key, a signer generates a unique signature for a message, and the corresponding public key is used for signature verification. The core processes of digital signature schemes encompass three steps: key generation, signature creation, and signature verification.

Key Generation

The key generation process involves creating a pair of cryptographic keys - a private key and a corresponding public key. These keys are intricately linked, but while the private key is kept securely by the signer, the public key can be openly shared [27].

Signature Creation

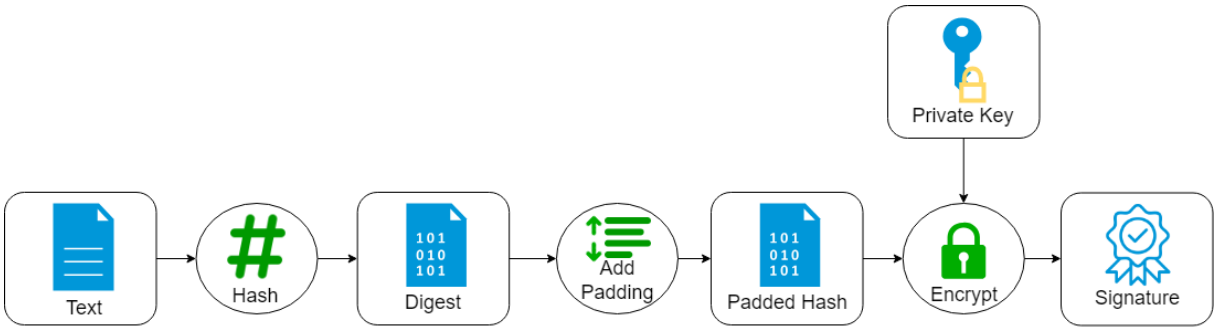

Figure 3.1: Signature Creation Diagram

The signer utilizes its private key to create a digital signature, as illustrated in Figure 3.1. This digital signature serves as a cryptographic representation, encompassing both the content being signed and the identity of the signer [35]. In the realm of asymmetric

encryption, where computational demands are typically higher, an optimization technique involves encrypting a digest of the content. This digest is produced through the application of a hashing function, such as SHA-256 [38], to the message, resulting in a compact representation. This approach maintains the security of the signature while addressing the performance overhead that arises when encrypting the entire content [27].

In RSA encryption, the use of padding is recommended to enhance security and protect against various potential attacks. Padding plays a crucial role in making cryptographic operations more robust and resistant to vulnerabilities. PKCS#1 v1.5 padding [31], for example, offers advantages such as preventing known-plaintext attacks and ensuring consistent padding length.

One notable characteristic of RSA signatures is their determinism, which means that they produce equal signatures for the same input data. RSA-Probabilistic Signature Scheme (PSS) padding [36] introduces a valuable feature by making signatures non-deterministic. This is achieved by injecting randomness and structure into the data, which serves as a crucial defense against specific attack scenarios, particularly those involving chosen-message attacks, known as Existential Unforgeability under Chosen-Message Attack (EUF-CMA) [13].

However, it's essential to note that RSA-PSS does come with certain trade-offs. It requires additional information to be sent along with the signature for verification, which can increase overhead. Moreover, the RSA-PSS scheme tends to be computationally heavier compared to PKCS#1 v1.5 and is not as widely adopted.

Signature Verification

Signature verification is the counterpart to signature creation, Figure 3.2. This process involves using the public key associated with the signer to verify the authenticity and integrity of the digital signature. By comparing the computed hash with the received one, using the public key to decrypt the signature, anyone can confirm that the content hasn't been tampered with and that it indeed originated from the claimed signer. [27]

Other Encryption Methods

In addition to the renowned RSA encryption method, other encryption techniques have gained prominence in the realm of digital signatures. Notable among these are the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). Each of these asymmetric encryption methods offers unique advantages and use cases.

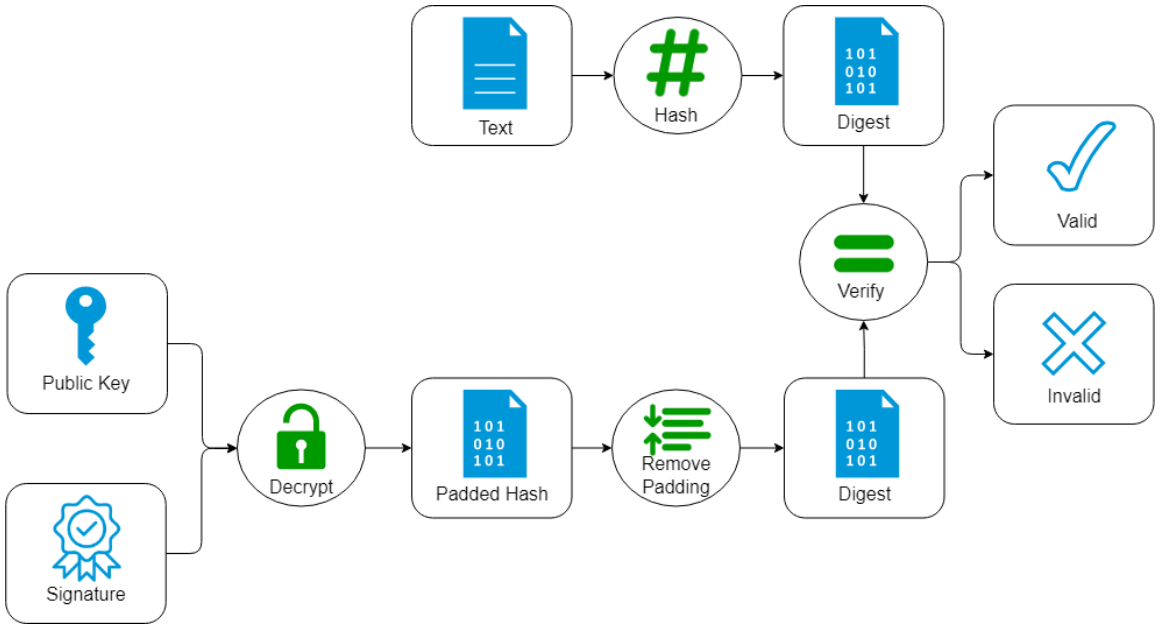

Figure 3.2: Signature Verification Diagram

The contrast between DSA and ECDSA lies in their mathematical foundations and efficiency. DSA relies on the discrete logarithm problem in finite fields, necessitating larger key sizes for security. In contrast, ECDSA utilizes elliptic curves, providing stronger security with shorter key lengths.

Moreover, while RSA relies on the difficulty of factoring large composite numbers into their prime factors, ECDSA bases its security on the difficulty of solving the elliptic curve discrete logarithm problem [39]. This formal difference makes RSA computationally more demanding. Another noteworthy divergence is that ECDSA by itself can be non-deterministic by allowing the inclusion of a nonce, a random value, into the signature generation process. It's important to highlight that this nonce does not affect the signature verification process. The verifier is not required to know the specific value of the nonce. Instead, the verification process involves confirming that the signature satisfies specific mathematical equations and properties. This feature sets it apart from RSA, as ECDSA signatures do not require padding, further enhancing its efficiency and versatility.

In this project, ECDSA and RSA were both implemented and compared as they are both widely used, each with its own strengths and suitability for various applications. ECDSA excels in efficiency, offering strong security with shorter key lengths, making it ideal for resource-constrained environments like Internet of Things (IoT) and mobile devices. On the other hand, RSA, though more computationally demanding, remains

a trusted choice for applications where computational resources are not a primary concern. It provides robust security and is well-established in secure communications and key exchange scenarios.

3.1.3 Identity Verification and Pseudonyms

The proliferation of multiple online personas has brought a significant problem, the erosion of trust in digital interactions. With the ability to hide behind pseudonyms and avatars, individuals can adopt various online identities, making it increasingly difficult to discern the genuine from the deceptive. As a consequence, even the most fundamental aspects of online engagement, such as product reviews, social media interactions, and e-commerce transactions, have become plagued by doubts and uncertainties [5].

Pseudonymous interaction online has proven to be better than either identifiable or anonymous interaction. Platforms like Disqus, the well-known online comment management platform used by many news organizations, have found that allowing pseudonyms not only leads to more comments but also improves the quality of the comments. People who use pseudonyms tend to post better comments on Disqus, with more likes and generating more discussion. This phenomenon can be attributed to the fact that under pseudonyms, individuals often feel more liberated to express their genuine likes, dislikes, prejudices, and opinions, contributing to richer and more open discussions [5].

However, for pseudonyms to truly have value, they need to be underwritten by trusted institutions. Imagine a scenario where an online pseudonym is underwritten by a reputable bank, ensuring that the bank knows the real identity behind the pseudonym. In such cases, website administrators and users can be more comfortable engaging with individuals using these pseudonyms because, if any issues arise, they can rely on the trusted institution, such as the bank, to resolve them. Conversely, when pseudonyms lack such backing, their acceptability can vary significantly, with some cases being deemed acceptable and others not [5]. Alternatively, websites and online platforms can play a pivotal role in verifying the authenticity of pseudonymous identities. By implementing robust verification processes, administrators can ensure that users' pseudonyms are linked to their true identities, further enhancing trust within their digital communities.

This is where identity verification emerges as a crucial cornerstone of the digital world. Identity verification serves as the keystone in rebuilding trust in the digital realm by empowering pseudonymous interactions with a layer of authenticity and accountability. It acts as a powerful tool in fostering transparency and credibility in the constantly

evolving digital landscape. By anchoring online interactions to verified identities, it paves the way for more trustworthy online reviews, secure financial transactions, and reliable information exchanges.

There are several methods of identity verification, ranging from simple to more complex, depending on the level of security required. Some common methods include [30]:

- **Email Verification -** This is a basic form of verification where the user is required to verify their email address by clicking on a link sent to them during the signup process. While it is easy to implement, it may not provide a high level of reliability.
- **Two-Factor Authentication (2FA) -** With 2FA, the user is required to provide a second form of verification, such as a one-time code sent to his mobile phone, in addition to his password. This adds an extra layer of trust and helps prevent unauthorized access.
- **Government-Issued ID Verification -** In more stringent verification processes, users may be asked to upload a scanned copy of their government-issued identification, such as a driver's license or passport. This method helps verify the user's identity against official records. This method also can be used to make sure that one person can have only one account.
- **Biometric Verification -** Biometric data, such as fingerprints or facial recognition, can be used for advanced identity verification. This method is highly secure and difficult to fake.

3.1.4 Information Security

Information security is a crucial aspect of modern society, focusing on the protection of sensitive and valuable information from unauthorized access, disclosure, disruption, modification or destruction [7]. It encompasses a range of practices, technologies and policies aimed at ensuring the confidentiality, integrity and availability of information.

The three fundamental principles of information security are:

- **Confidentiality -** This principle ensures that only authorized individuals or entities can access and view sensitive information. Confidentiality aims to prevent unauthorized disclosure of data [28], protecting it from falling into the wrong

hands. Role-Based Access Control (RBAC) is a common method for ensuring confidentiality. By assigning roles to users and granting permissions based on these roles, organizations can control who has access to what parts of the system. Identity verification mechanisms, such as multi-factor authentication, also contribute to confidentiality by confirming the legitimacy of users attempting to access the system.

- **Integrity -** The integrity principle focuses on the accuracy and trustworthiness of data and systems. It ensures that information remains accurate, unaltered, and reliable throughout its lifecycle [22]. To maintain integrity, techniques like hashing and digital signatures are utilized. Hash functions generate fixed-size hash codes unique to the content of a file. By comparing the hash before and after transmission, one can verify if the data remains unchanged. Digital signatures use cryptographic methods to provide a verifiable attestation of the origin and integrity of a message or document.

- **Availability -** Availability ensures that authorized users have timely access to information and resources when needed [43]. It guards against disruptions, downtime and Denial of Service (DoS) attacks that could render systems or data unavailable. Techniques to enhance availability include robust infrastructure design, load balancing to distribute traffic efficiently and resilient architectures to withstand DoS attacks. Additionally, employing strategies like redundancy and failover mechanisms ensures continuous service even in the face of unexpected failures.

United by a shared objective, these three principles establish the strong foundation of information security, ensuring the safety and trustworthiness of the digital landscape.

3.2 Technology

In this section, we will explore the technologies chosen for the implementation of the system and delve into the reasoning behind their selection over alternative options.

3.2.1 Content Management System (CMS)

Using a Content Management System (CMS) instead of building a website or application from scratch offers several advantages. A CMS provides pre-built features, templates, and user-friendly content management interfaces that save time and reduce

development costs. It offers customization options, scalability and benefits from a supportive community. By leveraging a CMS, the focus can be directed toward delivering quality content and functionality while speeding up the development process [45].

Table 3.1: Comparison of Content Management Systems

CMS	Programming Language	User Management	Multi-lingual	Searchable Content	Visual Editor	Documentation and Community
Wiki.js [46]	JavaScript (Node.js)	Yes	Yes	Yes	Yes	High
django-wiki [11]	Python (Django)	Yes	Yes	Yes	Yes	Moderate
Media Wiki [23]	PHP	Yes	Yes	Yes	Yes	High

There are multiple CMSs that fill the project needs, so it's crucial to evaluate their features and see which one meets the requirements best. As seen in Table 3.1, in this comparison there will be an examination of Wiki.js [46], django-wiki [11] and MediaWiki [23] based on a number of important criteria:

- **Programming Language -** Wiki.js is built using JavaScript (Node.js), django-wiki is developed using Python (Django framework) and MediaWiki is primarily written in PHP.
- **User Management -** All three CMS options provide user management capabilities, allowing you to create and manage user accounts with different access levels.
- **Multilingual Support -** Wiki.js and MediaWiki both offer built-in multilingual support, allowing the creation of content in multiple languages. Django-wiki also has powerful support for multilingual websites however, it may require additional customization to achieve full multilingual capabilities. This means that configuring and managing multilingual content in django-wiki may involve some additional effort compared to the native multilingual support provided by Wiki.js and MediaWiki.
- **Searchable Content -** Wiki.js, django-wiki and MediaWiki all offer a powerful search functionality, making it easy for users to search and find desired content within the system.

- **Visual Editor -** Wiki.js and MediaWiki both have visual editors that simplify content creation and formatting, providing a user-friendly What You See Is What You Get (WYSIWYG) interface. On the other hand, django-wiki primarily focuses on a markup-based editing experience.

- **Database Management System (DBMS) Support -** Wiki.js and django-wiki have the same range of DBMS support: PostgreSQL, MySQL, MariaDB, MS SQL and SQL Lite. MediaWiki also supports the same DBMSs but using database software different from MariaDB or MySQL is not recommended for production use.

- **Community and Support -** All three CMS options, Wiki.js, django-wiki and MediaWiki, have documentation available, including user and installation guides. Additionally, both Wiki.js and django-wiki have active releases, with ongoing development and engagement from their respective developer communities. They offer regular updates, bug fixes and new features to enhance the CMS functionality and user experience. The community support and user base for both Wiki.js and MediaWiki are well-established, with a wide range of plugins, extensions and user forums available. While django-wiki also has a supportive community, it may have a slightly smaller user base compared to the other two options.

Based on these factors, the chosen CMS for the project is django-wiki. The decision was influenced by several factors, but mainly on its Python-based development. It also fills the requirements of having user management capabilities, support for multiple databases, an active community and multilingual support.

3.2.2 Database Management System (DBMS)

As mentioned before django-wiki offers support for various Relational Database Management System (RDBMS)s, including PostgreSQL, MySQL, MariaDB, MS SQL and SQL Lite. Each of these databases has its own characteristics that influence the choice for a particular project.

PostgreSQL, being an open-source RDBMS, is well known for its reliability, scalability and wide range of features. It is a popular option for applications that demand high performance and reliability because it supports complex queries, data integrity and concurrency control. Additionally, PostgreSQL provides excellent spatial data support and sophisticated indexing options, both of which can be helpful for specific kinds of CMSs.

While MySQL and MariaDB are also popular options known for their performance and usability, PostgreSQL's advanced features and reputation for handling complex data models make it a preferred choice for many enterprise-level projects. MS SQL, a commercial database management system developed by Microsoft, is widely used in Windows-based environments, especially for projects with existing Microsoft infrastructure and technologies. SQL Lite, a lightweight embedded database, is appropriate for small-scale applications or projects with minimal resource requirements.

In the end, the choice of PostgreSQL for django-wiki was influenced by its extensive feature set, scalability and performance, along with the development team's familiarity and expertise with this particular DBMS.

3.2.3 Architectural Pattern

The ADDA platform adopts the Model-Template-View (MTV) architectural pattern, aligning with the pattern used by the chosen CMS, django-wiki. The MTV pattern is similar to the Model-View-Controller (MVC) pattern but with some key differences:

- **Model -** The Model component represents the data and business logic of the application. It defines the structure and behavior of the articles, user accounts, revisions and other entities. The Model interacts with the database and provides an abstraction layer for data manipulation and storage.
- **Template -** The Template component is responsible for defining the presentation logic of the web pages. It contains HyperText Markup Language (HTML) with placeholders for dynamic content. Templates in Django enable the separation of design and logic, allowing developers to create reusable HTML templates that can be populated with data from the Model.
- **View -** The View component handles the user's requests and defines the logic for processing those requests. It retrieves data from the Model and renders the appropriate Template to generate the final HTML response. Views handle the interaction between the user and the application, performing actions such as displaying articles, creating or editing content and managing user accounts.

MTV architecture promotes the "fat model, thin view" principle, where the majority of the application's logic resides in the Model, while the View remains lightweight, focusing on coordinating the Model and the Template. This approach encourages a clean separation of responsibilities and facilitates code reuse across different views and templates.

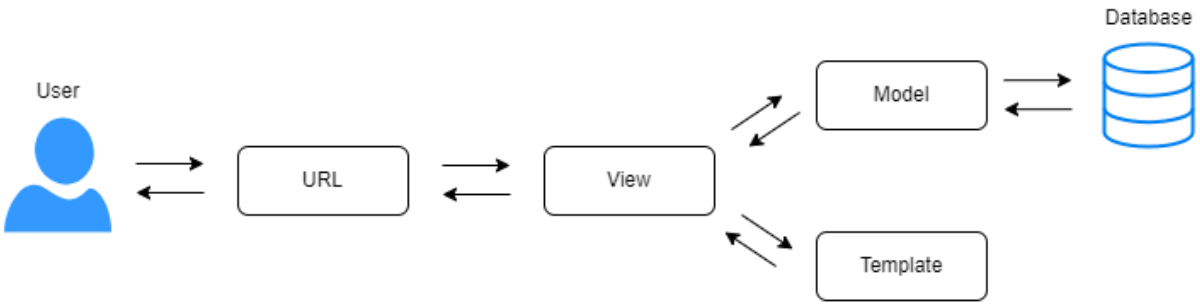

Figure 3.3: Model-Template-View Architecture

URL Component Additionally, a Uniform Resource Locator (URL) component is used as illustrated in Figure 3.3. This component maps incoming user requests to the appropriate View functions.

Benefits The MTV architecture offers several key benefits that contribute to its scalability and maintainability. By adhering to this architectural pattern, applications leverage the following advantages:

- **Scalability -** Promotes a clear separation of concerns, allowing developers to organize the application's components efficiently. This separation enables the application to scale more effectively, as different components can be optimized independently without affecting others.

- **Modularity -** Encourages modularity by breaking down the application into distinct components. Each component can be developed and maintained separately, making it easier to add new features or make changes without affecting the entire codebase. The modularity also fosters collaboration among developers working on different parts of the application.

- **Code Reusability -** Promotes code reusability, as components are designed to handle specific tasks independently. Custom views, templates and models can be reused across different parts of the application, reducing redundant code and simplifying maintenance.

- **Testability -** Enhances the testability of the application. With well-defined and isolated components, it becomes easier to write unit tests for individual views, templates and models. This helps identify and fix bugs early in the development process, leading to a more stable and reliable application.

3.2.4 Identity Verification

For identity verification, two options were considered: Autenticação .Gov (Portuguese government app) [4] and Persona [29]. Each option's benefits and drawbacks are listed in Table 3.2.

Table 3.2: Identity Verification Options

Authentication	**Advantages**	**Disadvantages**
Autenticação .Gov	- Recognized government method. - Familiar to Portuguese users. - Multiple methods of authentication.	- Only supports Portuguese government IDs. - Non-customizable level of security. - Requires registration in 3rd party applications. - Requires email to eid@ama.pt for more information. - Only available through API requests.
Persona	- Customizable level of security. - Dashboard to view and manage verifications. - Supports multiple government IDs. - Available through API requests, Android Software Development Kit (SDK) or JavaScript Embedded.	- Authentication not supported. - Free version has only 500 free government ID and selfie verifications per month.

Based on the evaluation of these options, Persona was chosen as the preferred identity verification solution due to its customizable security level, user-friendly verification management dashboard and support for multiple government IDs. However, it's important to note that Persona does not support authentication, which may need to be considered based on others projects with such requirement.

3.2.5 Non-Repudiation

Two non-repudiation methods were considered for ensuring the integrity and authenticity of user actions: Autenticação.Gov SDK and Asymmetric Key Signature Application. Autenticação.Gov SDK leverages the XML Advanced Electronic Signatures

(XAdES) technology, generating a zip file with Associated Signature Containers (ASiC) format. It can be used with the most recent Portuguese citizen card and a card reader or by activating "Chave Móvel Digital" [6], which involves associating a key with a phone number through the Portuguese government.

In contrast, the Asymmetric Key Signature Application refers to an application built from scratch that will use RSA or ECDSA signature methods. This application can be customized to be executed in background after computer start, being invisible for the user after installation. Table 3.3 shows the characteristics of both methods.

Table 3.3: Non-Repudiation Methods

Method	**Advantages**	**Disadvantages**
Autenticação .Gov SDK	- Supports XAdES (any file type). - Most code already created.	- Only possible for users with Portuguese government IDs. - Might need card reader. - User needs to download application prior to use, unless "Chave Móvel Digital" is used. - User needs to execute before every post.
Asymmetric Key Signature Application	- Supports multiple file types. - No restriction for users.	- Code from scratch. - Needs initial setup. - User needs to download application prior to use.

It's important to note that some older users may face difficulties with Autenticação .Gov SDK method if they possess older forms of identification, such as a lifelong identity card. Additionally, setting up "Chave Móvel Digital" can be challenging for some users. This reason along with the region restriction, made choosing to create a signature application from scratch a much more reasonable choice.

3.2.6 Web Application Server Infrastructure

For the server infrastructure of the ADDA platform, the project team has selected TrueNAS SCALE, an open-source HyperConverged Infrastructure (HCI) [41]. TrueNAS provides a secure and reliable environment for data storage and management, that can accommodate the anticipated influx of users and storage requirements of the platform.

TrueNAS, with its utilization of the OpenZFS file system, offers advanced data management capabilities, data protection features and efficient storage utilization, all of

which are essential for the successful operation of the platform.

In addition to supporting the platform's core functionality, the server infrastructure powered by TrueNAS enables efficient data storage, retrieval and backup processes. It provides the necessary resources to handle user-generated content, including text, images, videos and audio, ensuring that the platform remains accessible and responsive even during peak usage periods.

Furthermore, the user-friendly web-based interface of TrueNAS simplifies the management and administration of the server infrastructure. It allows the project team to monitor the system's performance, allocate resources effectively and proactively address any potential issues.

3.2.7 Email Service Provider

An email service provider, commonly referred to as an Simple Mail Transfer Protocol (SMTP) server, is a critical component of the email communication process. It is a specialized server designed to send, receive and relay outgoing email messages between email clients and email servers. SMTP servers ensure the reliable delivery of email messages across the internet by following a set of rules and protocols.

There are many SMTP servers available, the most commonly used are: Gmail [18], Outlook [26] and SendGrid [37].

SendGrid was chosen for several reasons:

- **Reliability -** SendGrid has a reputation for high email deliverability rates, ensuring that emails sent from the platform reach their intended recipients' inboxes reliably.
- **Scalability -** SendGrid is a scalable solution, making it suitable for businesses of all sizes. It can handle high volumes of emails without compromising performance.
- **Developer-Friendly -** SendGrid offers extensive Application Programming Interface (API)s and SDKs that make it easy for developers to integrate email functionality into applications and websites.
- **Analytics and Tracking -** SendGrid provides detailed email tracking and analytics, allowing users to monitor the performance of their email campaigns and make data-driven decisions.

- **Compliance and Security -** SendGrid complies with industry-standard email regulations and provides security features to protect against email abuse and fraud.
- **Customization -** SendGrid allows for email template customization, making it possible to create branded and personalized email communications.

4

Proposed Solution

Within this chapter lies the core of the project's blueprint—a comprehensive exploration of our proposed solution, designed to meet the project's objectives. This chapter unfolds into two key sections: Requirements (Section 4.1), which lays the foundation for the project's functionality, and Methodology (Section 4.2), offering insights into the implementation approach.

4.1 Requirements

This section defines the system requirements and organizes them into use cases.

4.1.1 Overall Description

In this section, is described a summary of objectives, the target audience and goals to be achieved.

Target Audience

The target audience for the proposed system includes people with stories to share about the 25th of April of 1974. The following individuals could make up the target market:

1. **Participants and witnesses -** people who were present at the events, of the 25th of April, either personally or as witnesses and who want to share their experiences, memories and insights;
2. **General public -** individuals who are curious about the historical background, significance and effects of the 25th of April, from a variety of viewpoints and stories;
3. **Enthusiasts and contributors -** people who are interested in history, social movements or storytelling and who want to actively participate in disseminating content related to the 25th of April;
4. **Researchers and historians -** individuals who are interested in examining various points of view on the 25th of April, for research and educational purposes.

4.1.2 System Functions and Attributes

This section describes the functional and non-functional requirements of the system, also referred to, respectively, as functions and attributes of the system.

System Functions

The system functions represent everything that the system should do once it is completed. In order to define priorities, the functions are classified into the following categories:

- **Evident** - the function must be performed and is visible to the user;
- **Invisible** - the function must be performed but is not visible to the user;
- **Adorn** - the function is optional, soft to implement and a bonus to the user.

The functions have been grouped according to the modules in which the application is divided: authentication, identity verification, user, article and non-repudiation.

The authentication module aims to register the user so that he can be identified when accessing the application. The functions of this module are identified in Table 4.1.

The identity verification ensures that the person authenticating is indeed who he or she claims to be. It prevents individuals from having multiple accounts and enables

Table 4.1: Authentication Functions

Ref	Description	Category
R1.1	Allow users to create an account in the system	Evident
R1.2	Allow users to sign in	Evident
R1.3	Allow users to sign out	Evident
R1.4	Save user session	Invisible

Table 4.2: Identity Verification Functions

Ref	Description	Category
R2.1	Verify user identity on sign up	Evident

the attribution of credits and responsibility for the produced content. In the Table 4.2, we can observe the functions of this module.

Table 4.3 describes the user functions which allow the user to change his information. Users are allowed to have a pseudonym which provides a certain degree of anonymity, as only administrators have knowledge of their true identities.

Table 4.3: User Functions

Ref	Description	Category
R3.1	Allow users to define a pseudonym on registration	Evident
R3.2	Allow users to edit or add pseudonym after registration	Evident
R3.3	Enable users to change their password	Evident
R3.4	Enable users to recover their password	Evident

Table 4.4 includes all article-related functions that enable users to manage the article content and view articles created by other individuals.

Table 4.4: Article Functions

Ref	Description	Category
R4.1	Enable users to create new articles on the platform	Evident
R4.2	Allow users to attach photos to articles during the publishing process	Evident
R4.3	Support the display of photo attachments within the article content	Adorn
R4.4	Allow users to edit existing articles	Evident
R4.5	Allow users to view and search articles	Evident
R4.6	Provide users with the ability to view previous versions of articles	Evident
R4.7	Enable users to revert to a previous version of an article	Evident

Non-repudiation module ensures the authenticity and integrity of the published articles, its functions are described on Table 4.5.

Table 4.5: Non-Repudiation Functions

Ref	Description	Category
R5.1	Have a non-repudiation mechanism that activates on article publication and editing	Invisible

System Attributes

The system attributes, or non-functional requirements, represent the characteristics exhibited by the system. Similar to functions, they should also be categorized, with the following categories:

- **Mandatory -** the system must include it, typically a boundary constraint;
- **Desirable -** the system should be prepared to implement it.

The system attributes are represented in Table 4.6.

Table 4.6: System Attributes

Attribute	Detail / Boundary Constraint	Category
Platform Compatibility	The system should be able to function on the Windows and Linux operating systems.	Mandatory
Performance	The system should offer quick response times when loading and navigating through articles.	Mandatory
Scalability	The system ought to be able to handle an increasing number of users and articles without significantly degrading its performance.	Mandatory
Security	The system should implement strong security measures to prevent unauthorized access, ensure the accuracy of the content and protect user information.	Mandatory
Usability	The system ought to have a user-friendly interface that makes it simple for users to browse, create, edit and manage articles.	Mandatory
Reliability	The system must be extremely reliable in order to reduce downtime and guarantee that the platform is always available for users.	Mandatory
Language Support	The system should support multiple languages to serve a diverse user base.	Desirable
Support	The system should offer user support to help users use the platform effectively.	Desirable

Functions and System Attributes Relationship

The relationship established between the attributes and functions of the system is that an attribute can be implemented through a function or set of functions. This relationship is represented in Table B.1.

4.1.3 Use Cases

Each Use Case (UC) describes an utilization process and system functionalities, with a particular focus on identifying the actors involved and delineating their interactions with the system.

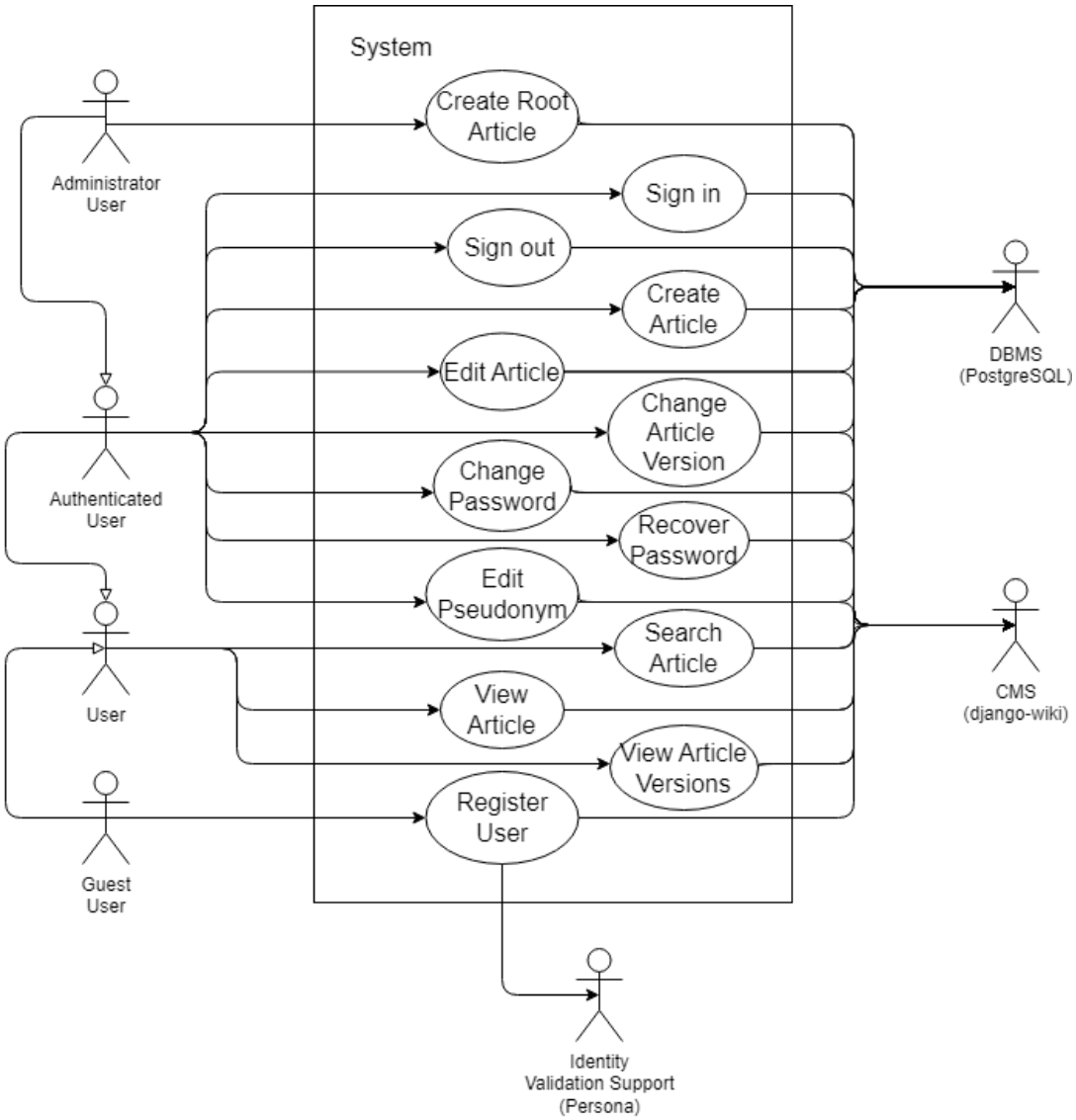

Figure 4.1: Use Case Diagram

As depicted in Figure 4.1, there are four distinct actors that can trigger an use case:

- **User -** This is an abstract representation and doesn't correspond to an actual entity within the system;
- **Guest User -** Extends from the base User and incorporates the additional functionality of registration;
- **Authenticated User -** Builds upon the User base and has access to a broad spectrum of use cases like editing and creating articles;
- **Administrator User -** Extends from Authenticated User, possesses access to all of its use cases and the additional capability to create root articles.

Also, the system depends on three actors:

- **CMS (Content Management System)** - Essential to all use cases, the system employs django-wiki [11] for efficient content management.
- **DBMS (Database Management System)** - A cornerstone in all scenarios, the system utilizes PostgreSQL for robust database management.
- **Identity Validation Support** - Exclusively required for the Register User use case, the system relies on Persona [29] for secure identity validation.

Tables 4.7 and 4.8 succinctly present the two pivotal use cases along with their associated system functions. For a detailed workflow of these two use cases, refer to Section 4.2.2. The remaining use cases are detailed in Attachment D.

Table 4.7: Use Case - Register User

UC1	
Name	Register User
Summary	First, the user is required to submit proof of his identity. Once the provided proof is successfully validated, he is then redirected to a new page where he can make a choice regarding the download of the non-repudiation app. If he opts to download and install it on his computer, it signifies that his publications will be fortified with his personal signature. Following the decision to download or not, he proceeds to a form page where he is prompted to provide his information. This information encompasses his email, pseudonym (optional), username, password and other details. Provided that all fields are duly validated, the registration process is considered complete.
References	R1.1, R2.1, R3.1, R5.1

Table 4.8: Use Case - Create Article

UC2	
Name	Create Article
Summary	The user initiates the article creation process by pressing the $< AddArticle >$ button. Subsequently, the user fills in the title and content fields, providing the necessary information for the article. Optionally, the user can attach images and incorporate them into the content. Once all the desired content is prepared, the user clicks on the $< Publish >$ button to complete the process, making the article available to others.
References	R4.1, R4.2, R5.1

4.1.4 Main Scenario

All systems have a main scenario that represents the typical flow of events. The main scenario of this system is depicted in Table 4.9.

Table 4.9: Main Scenario

Main Scenario			
Actor's Action		**System's Response**	
1	The Use Case begins when the user opens the website in the browser.		
2	The user starts by signing in by filling the log in form.		
		3	The system validates all fields in the submitted form and provides feedback on any errors.
4	The user is directed to the main page, clicks on $< AddArticle >$ fills in the desired fields, and clicks on the $< Publish >$ button.		
		5	The system performs field validation. If all fields are valid, the user's local machine is prompted to provide a signature. Whether the signature is received or not, the article is published.
6	The user sees his newly published article.		

4.2 Methodology

This section encompasses the system architecture (Section 4.2.1) and application workflow (Section 4.2.2), delving into the technical aspects that drive this project's development.

4.2.1 System Architecture

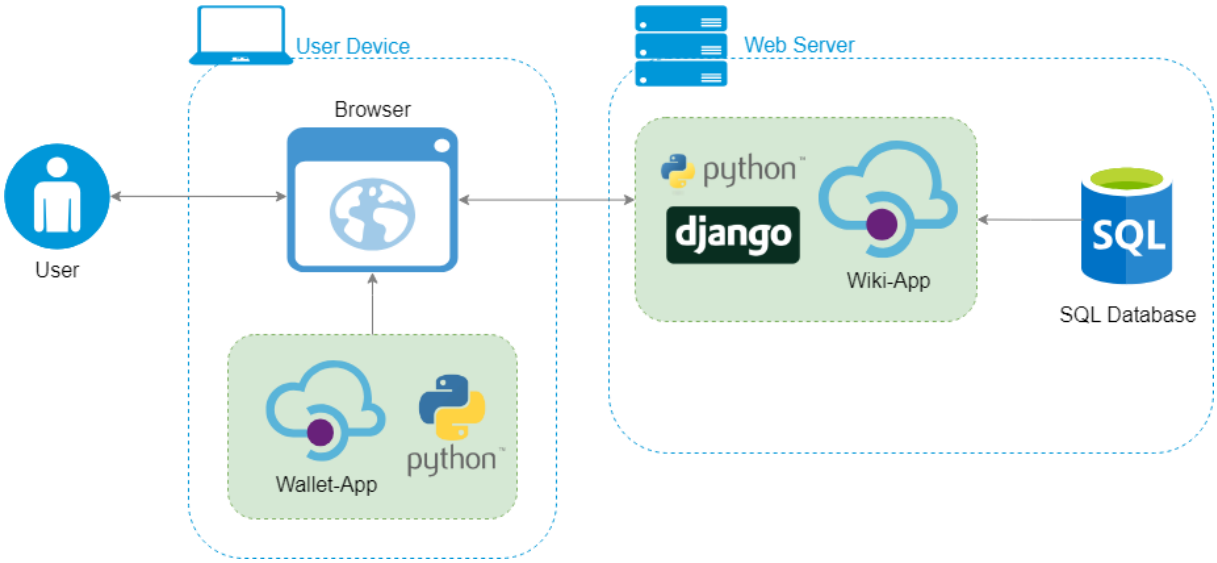

Figure 4.2: System Architecture

The system's architecture is visualized in Figure 4.2. It follows the client-server model, where users interact with the system through a web browser, employing the Hypertext Transfer Protocol (HTTP) protocol for communication.

The Server and the Client The server component serves as the central hub, housing the primary application known as Wiki-App. This application handles client requests and orchestrates the core logic of the platform. The server is equipped with a database, serving as the repository for storing article and user data. This setup enables the server to retrieve and persist data as necessary to support the application's functionality.

Wallet Application In addition to the client-server interaction, a critical component resides on the user's local machine—an application known as Wallet-App. This application plays a pivotal role in ensuring non-repudiation. Whenever users create or modify articles, the content is digitally signed using this application. This process guarantees the authenticity and integrity of user contributions within the system.

4.2.2 Application Workflow

The ADDA platform comprises several key components that work together to provide users with a seamless and secure experience. To visualize the interactions and processes within the system, Unified Modeling Language (UML) sequence diagrams are employed. UML sequence diagrams are a powerful tool for illustrating the dynamic behavior of the system, showcasing how various components communicate and collaborate during specific processes.

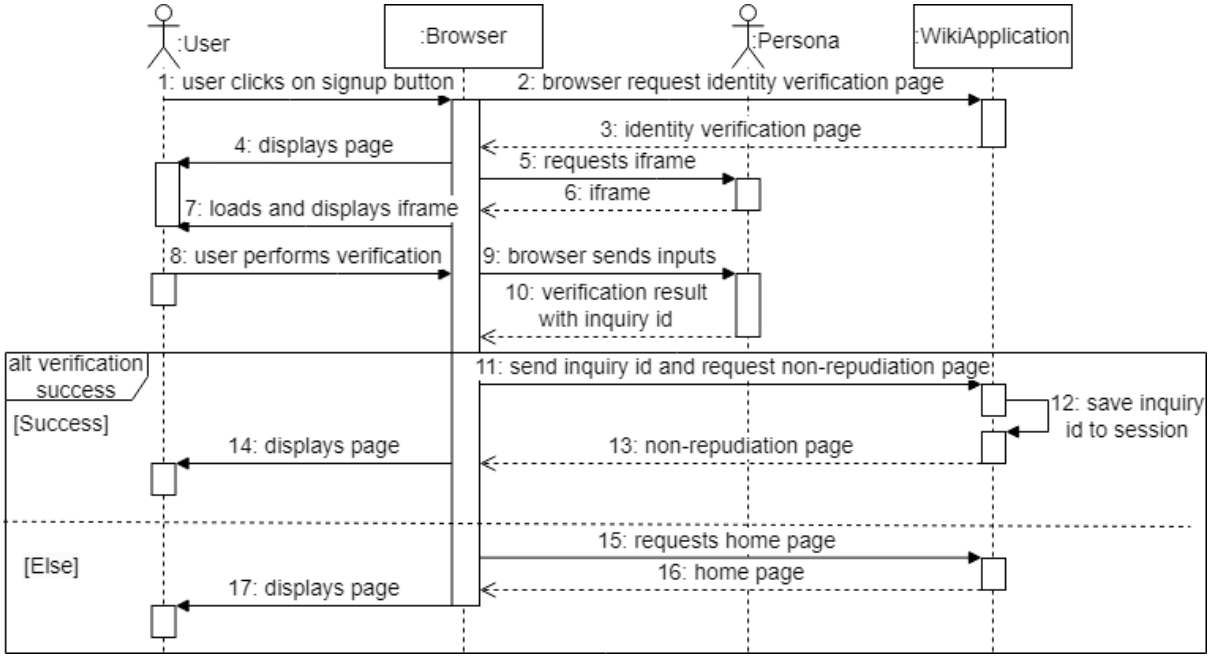

Figure 4.3: UML Sequence Diagram - Identity Verification

Identity Verification Depicted in Figure 4.3, is a visual narrative of the initial step within the user signup process: identity verification. This diagram illuminates the journey of users as they load the Persona iframe, which diligently guides them through the necessary steps to securely confirm their identity. Notably, during this process, the inquiry ID is received from Persona and thoughtfully preserved in the session for subsequent use and database storage.

Non-Repudiation within Signup Figure 4.4 presents the critical realm of non-repudiation within the signup process. This diagram unveils the multifaceted steps involved in downloading and installing the Wallet-App. Upon installation, it takes the initiative to generate a unique key pair, with the private key securely stored locally. Furthermore,

this diagram highlights the meticulous handling of signup form details, including the reception and storage of the public key furnished by the Wallet-App.

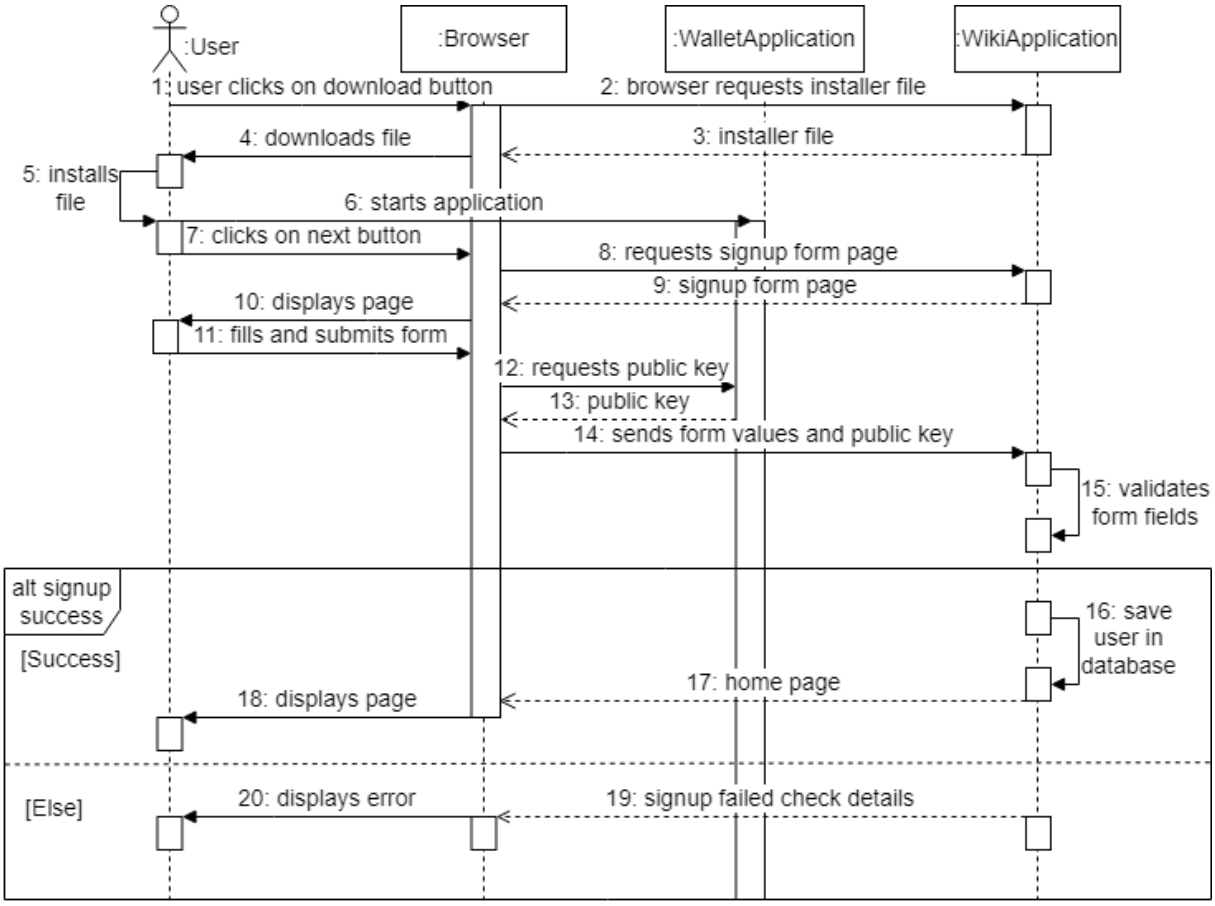

Figure 4.4: UML Sequence Diagram - Non-Repudiation

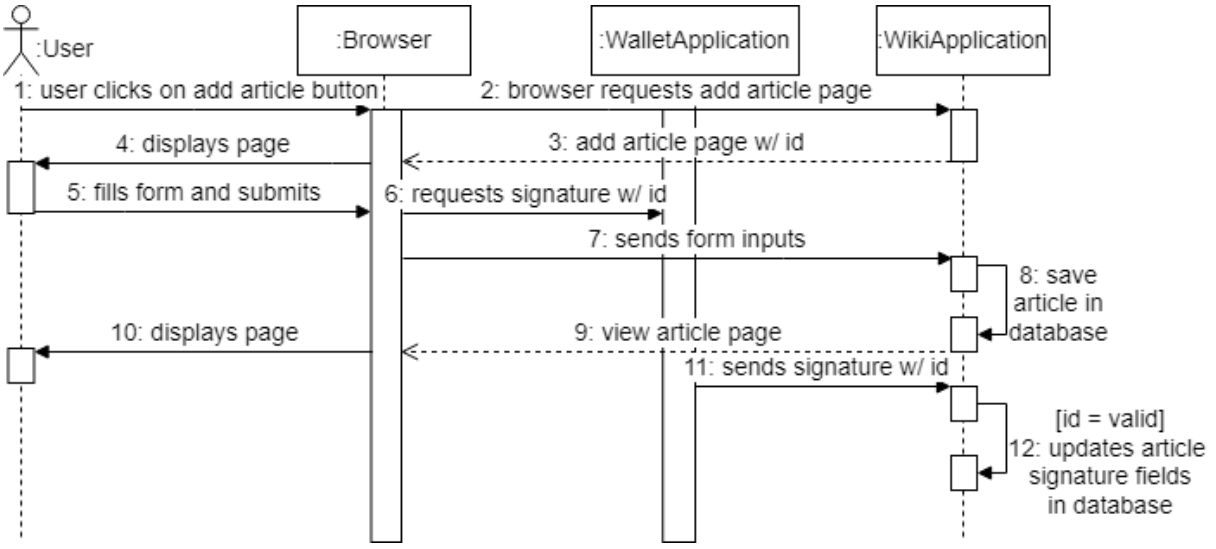

Figure 4.5: UML Sequence Diagram - Create Article

Article Creation Figure 4.5 offers a comprehensive overview of the article creation process. It illustrates how a user initiates the creation of a new article through his web browser, which then triggers the Wallet-App to request a digital signature. The diagram adeptly showcases the asynchronous nature of this workflow. While the user submits the article form and the Wiki-App saves the article, the Wallet-App generates and sends the signature to the Wiki-App. If the signature is received before the article is created, a retry system is employed. Sending the signature directly to the Wiki-App, rather than the browser, provides an added layer of protection against potential signature analysis attacks, bolstering the platform's security measures.

5

Implementation

This chapter describes the implementation process of the solution presented in Chapter 4. It explains the implementation of the key components of the application, highlighting their functionality and operation. Lastly, the technological dependencies will be introduced.

5.1 Wiki-App

The Wiki-App leverages the power of django-wiki as its foundation, it enhances the CMS with additional desired features and customization options. Two options were considered when integrating django-wiki into the application:

1. Create a fork from the django-wiki codebase for direct modifications;
2. Implement a middleware to intercept and process requests, which allows the injection of additional functionalities.

After careful consideration, the item 2 above was chosen. This approach was chosen for several reasons. Firstly, by using a middleware, the existing functionality of django-wiki could be leveraged while extending it with our customizations. Which means that any future improvements or bug fixes made to django-wiki would be automatically applied to the application. However, it should be noted that any potential

issues or bugs introduced in future updates of django-wiki or Django could also affect our application.

Another advantage of the middleware approach is that it provides a separation of concerns. By encapsulating the application specific customizations in a middleware layer, they'll be isolated from the core django-wiki codebase. This separation facilitates easier maintenance and upgrades, as the development team can focus on developing and maintaining only the middleware without directly modifying the django-wiki source code.

Table 5.1 provides an overview of the implementation of the UCs using the CMS, indicating which UCs were seamlessly integrated from the package and which required custom implementation to meet our specific requirements. It is important to highlight that identity verification, non-repudiation, pseudonym support and password change and recuperation all required custom implementation, as they demanded custom solutions not found in the default package.

Table 5.1: Implementation Types for Use Cases using the CMS

UC	**Implementation Type**
UC1, UC4, UC5, UC6, UC8, UC11, UC12, UC13	Custom Implementation
UC2, UC3, UC7, UC9, UC10	Package Default

5.1.1 Architecture

The Wiki-App follows the MTV architectural pattern (cf. Section 3.2.3). This architecture combined with the extension of the django-wiki framework empowers the application with scalability, maintainability and security.

Model Component In the MTV architecture of the Wiki-App, the Model component defines the structure and behavior of the entities that the application interacts with, such as articles, user accounts, revisions and more.

The Model component is directly related to the database schema through an Object-Relational Mapping (ORM) system. In the context of Django, the ORM allows developers to define Python classes that directly map to database tables. These classes, known as models, include attributes that map to the columns in the corresponding database tables and methods that encapsulate the business logic associated with those entities.

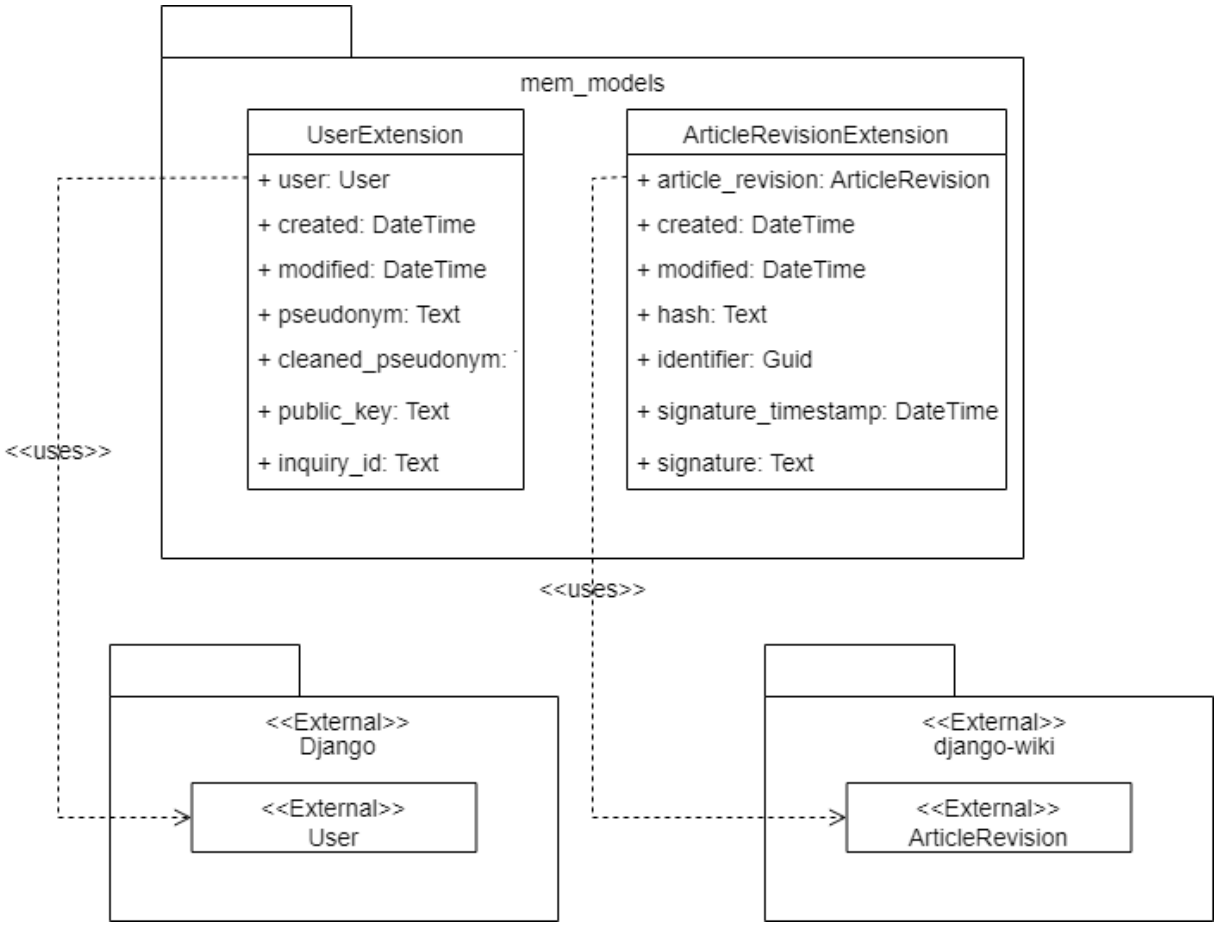

Figure 5.1: Class Diagram - Model Component

As seen in Figure 5.1, to extend the functionality of the existing django-wiki application, two new classes have been added to the Model component: *ArticleRevisionExtension* and *UserExtension*.

- **ArticleRevisionExtension -** Extends *ArticleRevision* entity and serves as a model for storing additional information related to article revisions. The *ArticleRevision Extension* class includes attributes such as the *hash*, which is used to store a generated hash in text format. The *signature* attribute stores the corresponding signature and the *signature_timestamp* the time in which the signature was created. Lastly, the *identifier* is a GUID that is used to identify the signature when received from the customer.

- **UserExtension -** Extends *User* entity and is used to store additional information related to user accounts. The *UserExtension* class includes attributes such as the *pseudonym* which stores the user-defined pseudonym, allowing various characters. To enforce uniqueness of pseudonyms, the *cleaned_pseudonym* attribute stores a canonical (lowercase, stripped accents and white spaces) version of the pseudonym. This canonical version allows for easy comparison and filtering to check for any existing pseudonyms that match. Additionally, the *public_key* attribute stores the user's public key in text format and the *inquiry_id* stores the Persona identification for identity check.

Template Component For the Wiki-App, new templates were created to accommodate the additional features and functionalities introduced, Table 5.2.

Table 5.2: Created Templates

Name	Description
Signup	Customized to include the pseudonym field and to send the public key request to the Wallet-App.
Create and Edit	Modified to request the signature information from the Wallet-App, Figure 5.2.
History	Shown in Figure 5.3, displays flags indicating if the revision is valid. It also shows the pseudonym (if it exists) instead of the username.
Identity_ Verification	Displays the Persona iframe which is responsible for verifying the user identity.
Non_ Repudiation	Allows users to download the Wallet-App for verified publications or skip (unverified posts in the future). Shown in Figure 5.4.
Password_ Reset	For the password recovery five new templates were created, including one for the email.

Figure 5.2: Template for Article Creation

Figure 5.3: Template for Article History

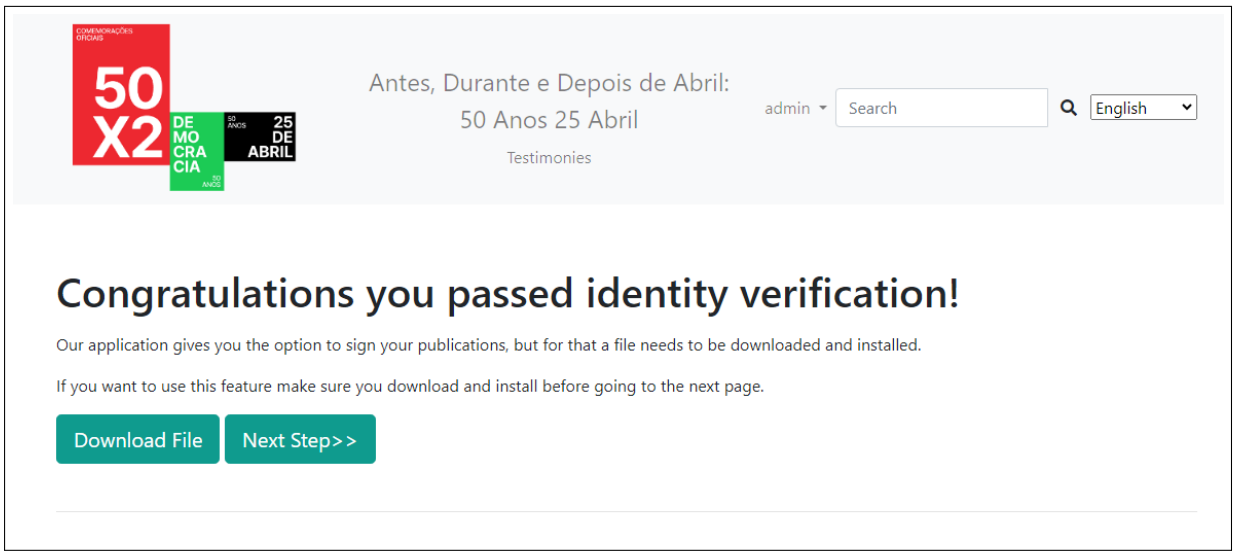

Figure 5.4: Template for Wallet-App Download

All of these templates used translation tags to make sure the content will be adapted to the user language choice.

View Component Within the Wiki-App's View component, a series of custom views have been strategically implemented to handle additional features, enhancing the application's functionality and security. These custom views, with the prefix "Mem", are depicted in Figure 5.5 and Table 5.3.

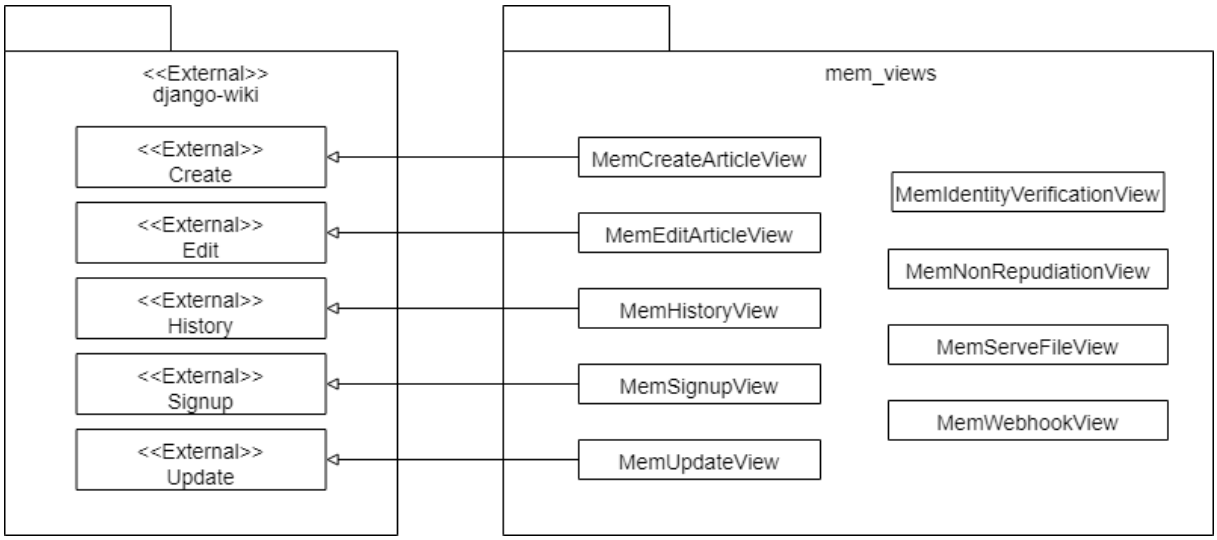

Figure 5.5: Class Diagram - View Component

In addition to the views, two distinct forms, as illustrated in Figure 5.6, have been developed. These forms serve a dual purpose, enabling users to both create and update

their pseudonyms while also facilitating the retrieval of their public keys upon signing up.

Table 5.3: Created Views

Name	Description
MemCreate-ArticleView	Generates an *ArticleRevisionExtension* with an unique identifier that will be sent to the Wallet-App along with the article content.
MemCreate-RootArticle-View	Similar to the *MemCreateArticleView*, with the only difference that this refers to the root article.
MemEdit-ArticleView	Creates an entry in *ArticleRevisionExtension* model when an article is edited. Links the revision to a unique identifier for the same purpose as in the *Create* views.
MemHistory-View	Each revision undergoes hash verification for content alignment, and the signature is authenticated using the editor's public key.
MemIdentity-Verification-View	Loads the correct template.
MemNon-Repudiation-View	This view saves the inquiry ID, initially passed through the URL, in the session for future use in the *Signup* view. It also loads the non-repudiation template.
MemServe-FileView	Serves as the endpoint to download the Wallet-App file.
MemSignup-View	Used to store fields: pseudonym, clean pseudonym, public key and inquiry ID within the *UserExtension* model.
MemWeb-hookView	Webhook for incoming user signature data requests. Verifies the received identifier's existence, if found, updates *ArticleRevisionExtension* signature information.

URL Component The URL configuration is a fundamental component in the Django web framework, responsible for directing incoming requests to specific views and functions within the application. With the addition of new custom views, it became essential to map these views to their corresponding URLs effectively. In the urls.py file, new entries were introduced to achieve this mapping. One such entry is exemplified below:

```
re_path(r"^_accounts/sign-up/$",
MemSignupView.as_view(), name="signup")
```

In this particular URL pattern, the endpoint $\_accounts/sign-up/$ is mapped to the MemSignupView view. Whenever a user visits this URL, the view takes charge of managing the signup process and rendering the appropriate template for user registration.

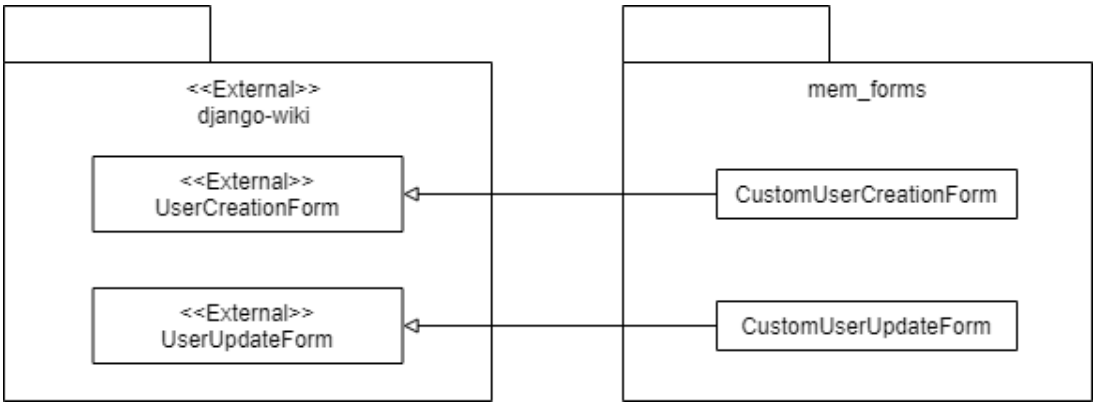

Figure 5.6: Class Diagram - Forms

By configuring these URL patterns, the application ensures that incoming requests are appropriately directed to the desired functionality.

5.1.2 Multilingual

The multilingual implementation in the Wiki-App is achieved using Django's powerful internationalization (i18n) framework [19]. The process involves identifying translatable strings, such as new fields like *pseudonym* and marking them for translation using Django's *gettext_lazy* function.

Once the translatable strings are marked, the next step is to create a .Portable Object (po) file. The .po file serves as a repository for the original strings and their corresponding translations. After marking the strings for translations, the .po file is generated by running the command:

```
python manage.py makemessages -l <language_code>
```

After the .po file is prepared with all the necessary translations, it is compiled into a binary format known as the .Machine Object (mo) file. The compilation process is done using Django's management command:

```
python manage.py compilemessages
```

The .mo file contains the translations in a format that is optimized for faster processing by Django when rendering the templates in the respective language.

By using the .po and .mo files, the Wiki-App can dynamically switch between different languages, providing a seamless multilingual experience to its users. The translations

are fetched from the appropriate .mo file based on the user's language preference, ensuring that they can interact with the application in their preferred language.

To ensure the whole system was multilingual, efforts were made beyond interface translation. The Wallet-App installer was also generated as multilingual. By doing so, the language barriers are addressed not only within the application itself but also in the installation process.

5.1.3 Persona

The identity verification process, using Persona [29], is integrated into the signup process as its initial step. After the Persona *iframe* is loaded, users are guided through the verification steps, including the validation of a legal document and camera-based validation (which can be performed on a different device by sending an email or text message with a link). Figure 5.7 provides a visual overview of some of these steps.

The integration of Persona into this project involved exploring different implementation possibilities, including API requests, SDK integration and JavaScript embedding. Given the absence of specific requirements regarding the visual aspects of identity verification, the best option was to use the embedded JavaScript approach. This decision eliminates the necessity for a separate frontend implementation, ensuring efficiency and saving valuable development time. The code snippet included in the Identity_Verification.html to load the Persona *iframe* is:

```
<script>
const client = new Persona.Client({

    templateId: "<templateId>",//Persona template Id
    environmentId: "<environmentId>", //Persona account secret key
    environment: "sandbox",
    onReady: () => client.open(),
    onCancel: ({ inquiryId, sessionToken }) =>
{ window.location.href = "/" },
    onError: (error) => { window.location.href = "/"; },
    onEvent: (name, metadata) => {
      if (name === 'start') {
        // Collect and save the inquiry ID for future use
        inquiryId = metadata["inquiryId"]
      }
```

```
    },
    onComplete: ({ inquiryId, status, fields }) => {
      // Inquiry completed.
      // Redirect to next page after inquiry is completed
      window.location.href=
        `/_accounts/non-repudiation/${inquiryId}`;
    },
});
</script>
```

As demonstrated in the code snippet, in case of verification error or cancellation, users are redirected to the homepage. Conversely, successful verification lets users proceed with the signup process. Upon successful verification, Persona generates an inquiry ID, that is passed to the next view to be securely stored in the session. This ID is later transferred to the *UserExtension* database table during user creation. This stored data serves as a valuable resource for future verification within the Persona dashboard.

5.1.4 Deployment

As depicted in Figure 5.8, the Wiki-App is hosted on a server with the TrueNAS SCALE, an open-source HCI [41]. TrueNAS provides a secure and reliable environment for data storage and management.

The server is located in a data center managed by the portuguese DSTI \ IGeFE (Departamento de Sistemas e Tecnologias de Informação \ Instituto de Gestão Financeira da Educação) [12], under the scope of a robust (regarding security, fault tolerance and maintenance) infrastructure.

To enhance flexibility and resource management, the Wiki-App is deployed within a virtualized environment with Alma Linux [3] Operating System (OS).

It's worth noting that the Wiki-App is already available for access [1], although initially, it offers a limited set of features. As development progresses, the contributions (features) herein described will be gradually introduced.

5.2 Wallet-App

The Wallet-App must execute locally (i.e., on the user's machine) to ensure that only the user has access to his private key. By keeping the private key local, the user retains

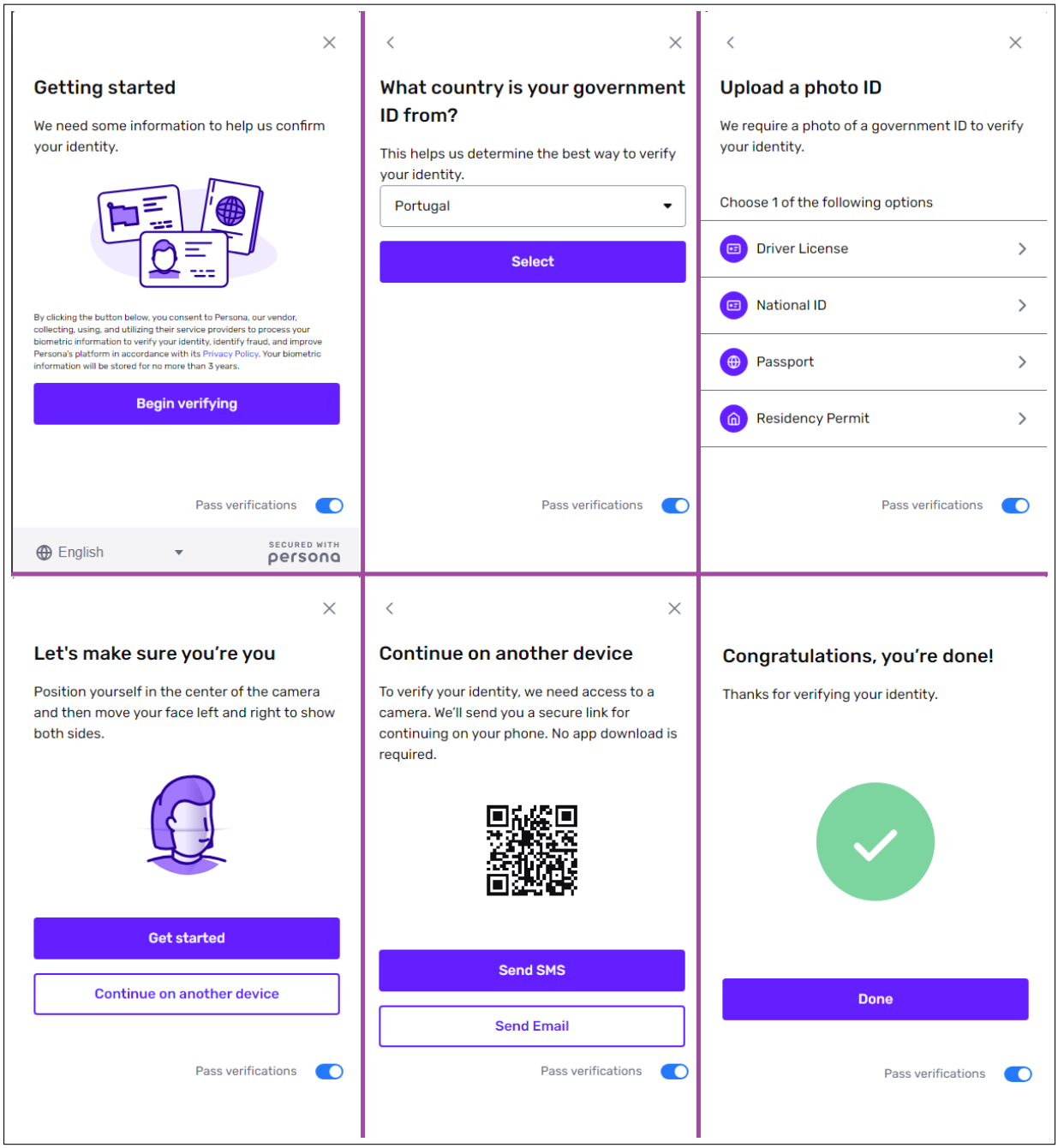

Figure 5.7: Persona IFrame Steps

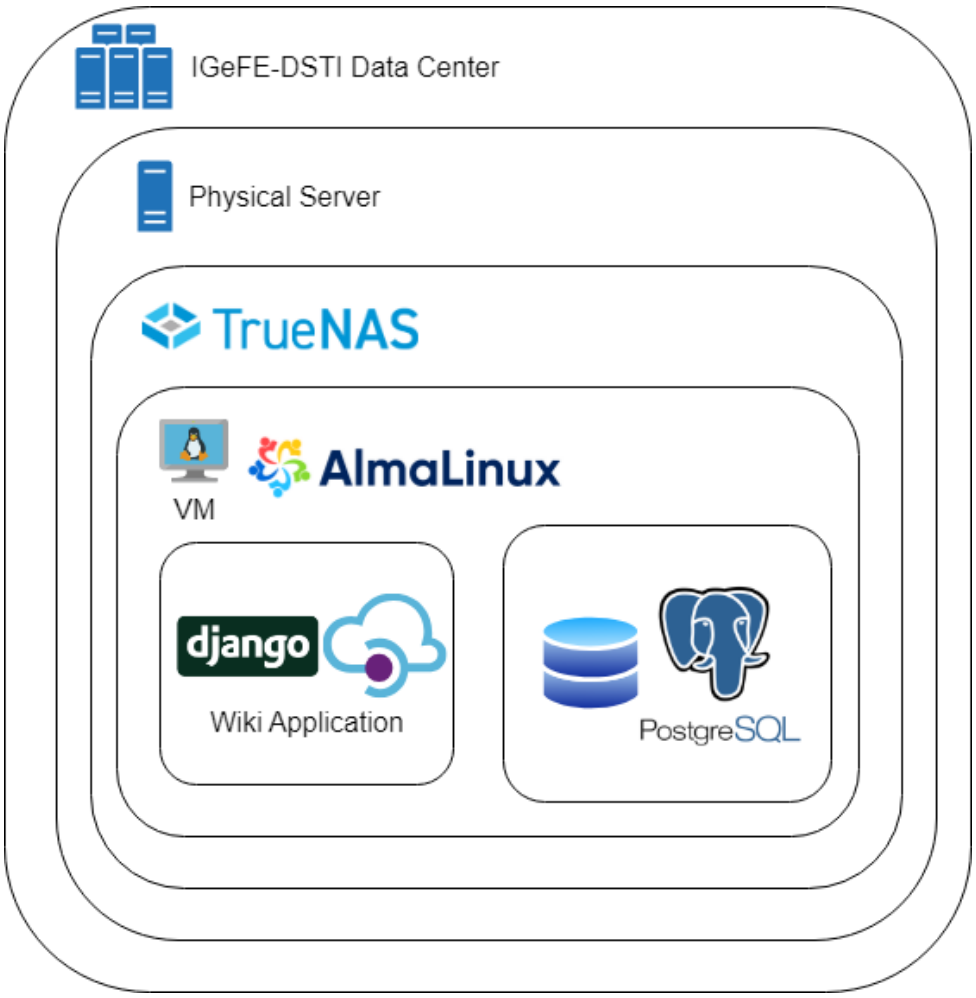

Figure 5.8: Infrastructure Diagram

full control over his cryptographic identity. This ensures the security of user-generated content and prevents unauthorized access to the private key, which is essential for signing article revisions and ensuring data integrity.

Communication Both web-socket and plain HTTP were contemplated as possibilities. However, given the sporadic need for communication, maintaining a WebSocket connection continuously open would be inefficient. WebSockets are more suitable for scenarios requiring continuous data flow. As the need here is occasional, HTTP communication was deemed the right choice. JavaScript functions embedded in the Wiki-App's templates interact with the Wallet-App's endpoints to perform cryptographic operations.

When a user creates or edits an article, the *GetSignature* endpoint is called to generate and obtain the digital signature for the content. Similarly, the *GetPublicKey* endpoint is called in the signup process so the public key can be saved for later signature verification.

GetPublicKey Endpoint Generates asymmetric key pairs if they don't already exist and returns the public key in string format. It takes two parameters: one for specifying the encryption algorithm (RSA or ECDSA) and another for determining whether to force the generation of new keys or return existing ones. In the key generation it's important to note that the RSA algorithm generates a key of 2048 bits, while ECDSA generates a key of 256 bits using the SECP256R1 curve.

GetSignature Endpoint It's triggered by a HTTP POST request, this endpoint receives a JSON-encoded body containing a string identifying the algorithm to be used, an identifier, article content and title. Upon receiving this data, the Wallet-App performs two operations. Firstly, it creates a cryptographic hash of the article content and a timestamp, using the secure SHA-256 hashing algorithm. If it's RSA a PKCS#1 v1.5 padding is also added in this step. Secondly, the Wallet-App employs the user's private key to generate a digital signature for the article. The resulting signature data is then transmitted directly to the server address instead of the browser. This approach mitigates potential attacks where malicious websites could exploit vulnerabilities by repeatedly requesting multiple signatures and studying response patterns.

Installation The Wallet-App is installed on the user's computer during the signup process. To create the installer, two tools were utilized: PyInstaller [33] and Inno Setup

[21]. PyInstaller is a multi-platform tool used to convert Python (.py) files into standalone executable files, ensuring compatibility across different operating systems without requiring prior installation of dependencies. Inno Setup, on the other hand, was employed to create the installer itself. It allows for multilingual support and includes conditions for running on computer startup, placing the executable file in the startup folder.

To ensure continuous protection and a streamlined user experience, the application is configured to run automatically in the background each time the computer starts. Operating silently in the background, the Wallet-App remains readily available to respond to requests from the main Wiki-App, facilitating seamless and secure cryptographic operations for each article created or edited by the user.

User education plays a great role, ensuring users understand the significance of their private key and its role in securing their contributions to the ADDA platform.

5.3 Database

As mentioned in Chapter 4, it was decided to use a PostgreSQL RDBMS. Three steps need to be followed to create the database:

1. **Drawing the Entity-Relationship (ER) Diagram -** In this step, a conceptual diagram that structures and organizes the data of the information system is created. The ER diagram represents the entities in the system, their attributes and the relationships between them.
2. **Establishing the Relational Model -** Based on the conceptual model, the logical relational model is established. This involves adapting the conceptual model into a tabular representation, establishing primary keys, foreign keys and relationships between the tables.
3. **Creating and Applying the Physical Model -** In this step, the physical organization of the data is carried out, which is dependent on the DBMS used, in this case PostgreSQL. The physical model defines how the data will be stored and accessed, taking into account the specific characteristics of the PosgtreSQL DBMS.

5.3.1 Conceptual Data-Model

The ER diagram is used to represent the entities (concepts) and the associations among those entities. It provides a visual representation of the database schema and shows

how different entities are related (regarding multiplicity and connectivity) to each other.

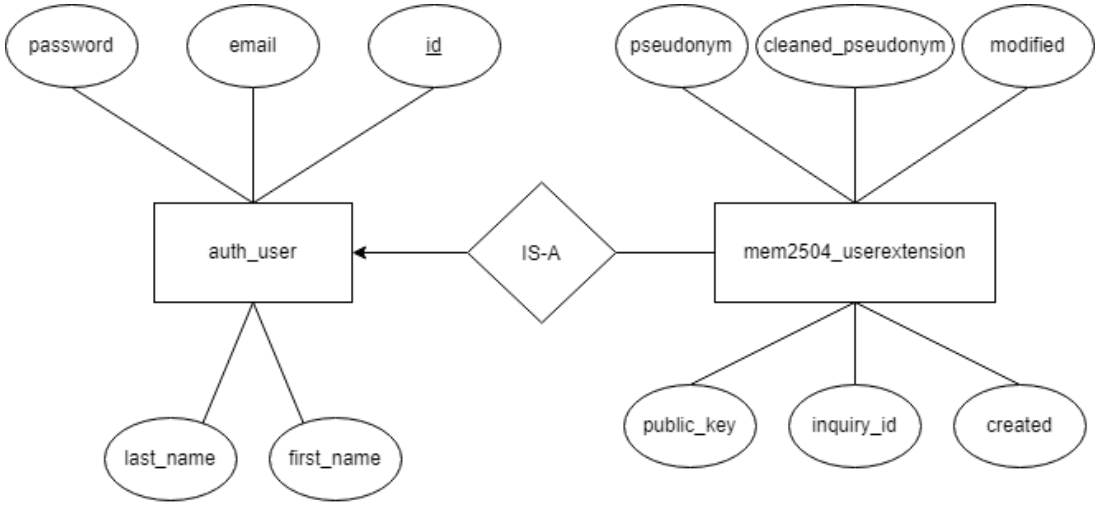

Figure 5.9: ER Diagram - User related Entities

The *auth_user* entity, inherited from Django's authentication system, stores user-related data such as passwords, usernames, first names and other authentication-related information. One of the two entities created for this specific project is related to it, Figure 5.9.

The *mem*2504*_user_extension* entity extends the *auth_user* entity and includes additional attributes such as pseudonym, clean pseudonym, public key and inquiry id. This entity is directly related to the class *UserExtension*.

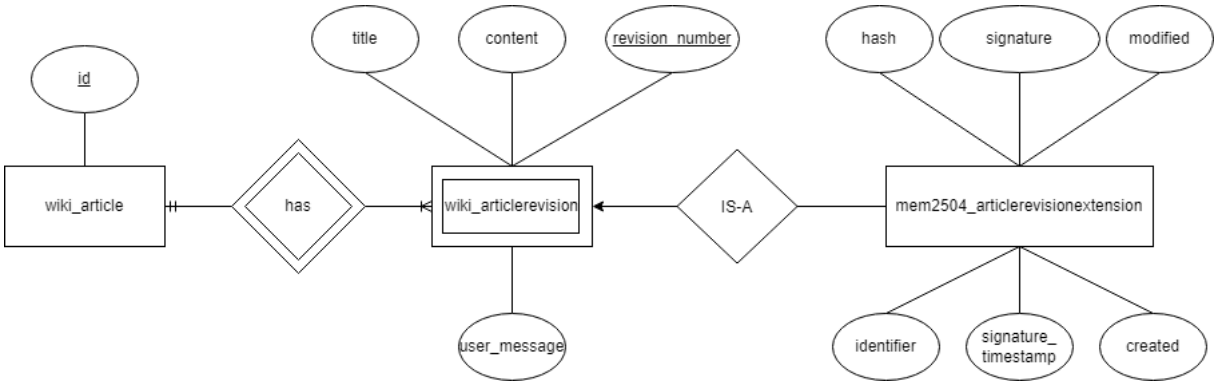

Figure 5.10: ER Diagram - Article Related Entities

The second created entity in the project scope was *mem*2504*_articlerevisionextension* which is related to the entities, inherited from django-wiki, *wiki_article* and *wiki_articlere vision*, Figure 5.10. This new entity is directly related to the class *ArticleRevisionExtension* mentioned in Section 5.1.1.

The *wiki_article* entity serves as the repository for storing articles. It establishes a one-to-many relationship with the *wiki_articlerevision* entity, dedicated to preserving various versions of articles, encompassing titles, content and additional revision details. The adoption of a one-to-many relationship stems from the fact that upon the creation of an article in the system, a corresponding article revision is automatically generated. So there's always at least one *wiki_articlerevision* associated with each *wiki_article*.

The *mem*2504*_articlerevisionextension* entity extends the *wiki_articlerevision* entity. It adds the fields associated with the signature: identifier, signature, hash and signature timestamp.

5.3.2 Logical Data-Model

The conceptual model is transposed to the relational data-model, a widely used conceptual framework for database design. This model ensures integrity relational constraints among tables, which are established through primary keys and foreign keys.

The tables were created with different prefixes based on their origin: "nyt" for tables related to django-nyt module which is used for notifications, "auth" for tables inherited from Django's authentication system, "wiki" for tables inherited from django-wiki, and "mem2504" for tables specific to this project.

Figures C.1 and C.2 depict the relational model for the added tables, while Figure C.3 shows the entire database relational model.

The *mem*2504*_user_extension* table extends the functionality of the *auth_user* table, as mentioned earlier. The primary key in *mem*2504*_user_extension* is the *user_id* which uniquely identifies each record. This *user_id* attribute also serves as a foreign key, establishing a connection to the corresponding *id* column in the *auth_user* table. This relationship ensures that each *mem*2504*_user_extension* record is associated with one and only one *auth_user* record.

Similarly, the *mem*2504*_articlerevisionextension* table extends the functionality of the *wiki_articlerevision* table. The primary key in *mem*2504*_articlerevisionextension* is the *article_revision_id*. This *article_revision_id* attribute also acts as a foreign key, linking to the corresponding *id* column in the *wiki_articlerevision* table. Once again, this relationship ensures that each *mem*2504*_articlerevisionextension* record is associated with one and only one *wiki_articlerevision* record.

5.3.3 Physical Data-Model

The physical model encompasses various considerations tailored to the chosen DBMS, PostgreSQL, including data types, storage structures, indexing strategies, partitioning schemes, caching mechanisms and other optimizations. Designing the physical model involves defining appropriate data types for each attribute, leveraging PostgreSQL's data type system. It also entails configuring table storage parameters, such as tablespaces and file locations, to ensure efficient storage allocation and management.

Looking at Figures C.1 and C.2, we can examine the added tables, their columns and respective data types.

In the *mem*2504_*user_extension* table, the *user_id* column serves as the primary key, uniquely identifying each record. The *created* and *modified* columns store timestamps indicating the entry's creation and last modification. The remaining columns (*pseudonym*, *clean_pseudonym*, *public_key* and *inquiry_id*) are all saved in text format.

For the *mem*2504_*articlerevisionextension* table, the primary key is the *article_revision_id* column, uniquely identifying each record. The *created* and *modified* columns have the same purpose as in the previous table. The *hash* column stores a generated hash in text format, while the *signature* column stores the corresponding signature also in text format. The *identifier* is a UUID field and the *signature_timestamp* as the name suggests is a datetime field.

Indexing is pivotal for enhancing query performance, and in the ADDA platform, a decision was made to create a single index for the *cleaned_pseudonym* column. This choice stems from the frequent need to search this field during registrations when the pseudonym is filled. While this indexing approach might slow down the write operations to this column, the trade-off proves beneficial considering the regularity of searches performed on this particular field. It's worth noting that other searches in the platform leverage primary keys, which come with default indexes.

PostgreSQL also provides caching mechanisms, such as shared buffers and query caches, which can be configured in the physical model to enhance overall system performance by minimizing disk I/O. However, due to frequent data updates in this project, it was determined that implementing a cache system was unnecessary.

5.4 Technological Dependencies

The ADDA platform relies on the functionality provided by SendGrid, Persona and django-wiki, and by association on Django.

Django-wiki Using the CMS with a middleware approach introduces a level of dependency on the internal workings of django-wiki. If significant changes are made to the django-wiki architecture or functionality, there may be the need to adjust the middleware accordingly to ensure compatibility and seamless integration. This aspect requires diligent monitoring and proactive development efforts on the Wiki-App part.

Persona Persona serves as a direct technological dependency, as our implementation and updates are directly influenced by its capabilities and enhancements. Regularly monitoring Persona's updates and maintaining compatibility with the Wiki-App is crucial to ensure optimal performance and security.

SendGrid Moreover, SendGrid plays a critical role in our project as a technological dependency. It is utilized for sending emails, particularly for processes like password recovery. As with other dependencies, it's essential to keep an eye on SendGrid's updates and maintain compatibility to guarantee efficient email communication within the Wiki-App application.

6

Evaluation

This chapter evaluates the performance and user experience of the application. The outcomes of two critical assessments: usability testing and speed tests of RSA and ECDSA algorithms are presented. The usability tests shed light on user interactions, uncovering strengths and areas for improvement. Additionally, the speed tests explore the efficiency of RSA and ECDSA in different scenarios.

6.1 Usability Testing

Usability is a critical aspect of any software application's success. A series of usability tests were conducted to ensure the application aligns with user expectations and provides a seamless experience. This section presents the methodology, results and insights derived from these tests.

6.1.1 Methodology

A series of usability tests were conducted, adhering to the guidelines put forth by Jakob Nielsen [25]. A focus group comprised of five participants were enlisted for these assessments. While testing with just 5 users may seem limited, studies indicate that it effectively reveals approximately 85% of usability problems [25]. Nielsen recommends distributing the testing budget across multiple small tests, enabling iterative improvements after each cycle [25]. This approach proves valuable in refining the application's

usability through continuous feedback and enhancement.

Participants were asked to complete all use cases, excluding the exclusive *CreateRootArticle* intended solely for administrators. Prior to starting the testing, participants were requested to provide personal information, including their date of birth, gender, occupation and their preferred language for using the application.

For each test task, participants were diligently guided through a structured set of inquiries, which encompassed the following key aspects:

1. Did you successfully complete the task?
2. Did you face any challenges?
3. Did you encounter error messages or unexpected behaviors?
4. Do you have any additional comments or feedback related to this task?

After completing all tasks, participants were invited to share their overall impressions of the application, its usability and any suggestions for improvement.

6.1.2 Results

This section unveils the results obtained from the conducted usability tests. Given that every participant successfully completed all the use cases, the initial question was excluded from the analysis. Figure 6.1 reveals the amount of participants responding with a "Yes" to the subsequent questions. The labels in the chart correspond to the numbers assigned to each question above.

Participants uncovered minor bugs and suggested areas for improvement. The bugs were promptly fixed and recommendations included:

- **Signup -** improve clarity in the text about downloading the Wallet-App and display a warning if the application is not installed and running;
- **Create Article -** add a button to return to the main page and change "Summary" to "Comment" for clearer labeling;
- **Edit Article -** place the image add button closer to the form for better visibility;
- **Switch Version -** enhance the clarity of the user interface to switch between versions and add a button to return from the history to the corresponding article;

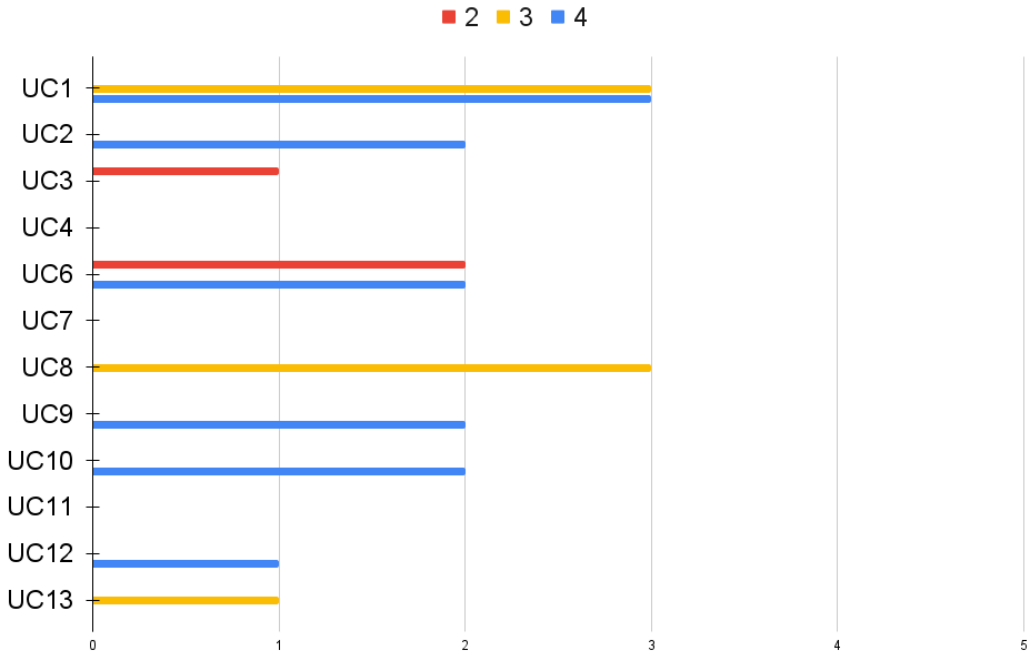

Figure 6.1: Usability Results Chart

- **Search Article -** expand search functionality to include similar words, not just exact matches and remove the redundant second search bar that appears.

The general impressions indicated a desire for a feature enabling them to access their personal article history, encompassing both creations and updates.

6.1.3 Discussion

The feedback obtained from participants collectively highlighted a favorable response to the application's design, with a particularly strong appreciation for functionalities related to article history viewing and version switching. These attributes were praised for their intuitive nature and contribution to a seamless user experience. The usability tests proved instrumental in identifying multiple areas for enhancement and improvement, ensuring that the platform aligns with user expectations.

6.2 Performance Tests for RSA and ECDSA

In this section, are presented the results of speed tests conducted to measure the performance of RSA and ECDSA cryptographic operations. The tests were designed to

assess the efficiency of these algorithms in various scenarios, including key generation, signature generation and signature verification.

6.2.1 Methodology

The speed tests were conducted using Postman [32], a popular API testing tool, to simulate real-world HTTP requests to the Wallet-App. Each test scenario captures the average execution time-span of 1000 requests for the cryptographic operations. The key aspects evaluated include:

- **Key Generation -** The time required to generate cryptographic keys for both RSA and ECDSA algorithms.
- **Signature Creation -** The time taken to generate signatures for messages of varying lengths using both RSA and ECDSA.
- **Signature Verification -** The time taken to verify signatures for the same message lengths as in the signature generation phase.
- **Video Conversion to Base64 -** Converting a video file (around 4 minute, 53MB .mp4 format) to Base64, resulting in a data size around 74 million characters. The signing and verification times were measured for this large dataset, with 5 requests for each operation.

6.2.2 Results

Tables 6.1 and 6.2, along with Figures 6.2 and 6.3, present a comprehensive overview of the test results.

Table 6.1: Key Generation Average Speed

Algorithm	Average Time (ms)
RSA	2310.11
ECDSA	135.88

6.2.3 Discussion

The results of the speed tests provide valuable insights into the efficiency of RSA and ECDSA cryptographic operations. Key takeaways from the tests include:

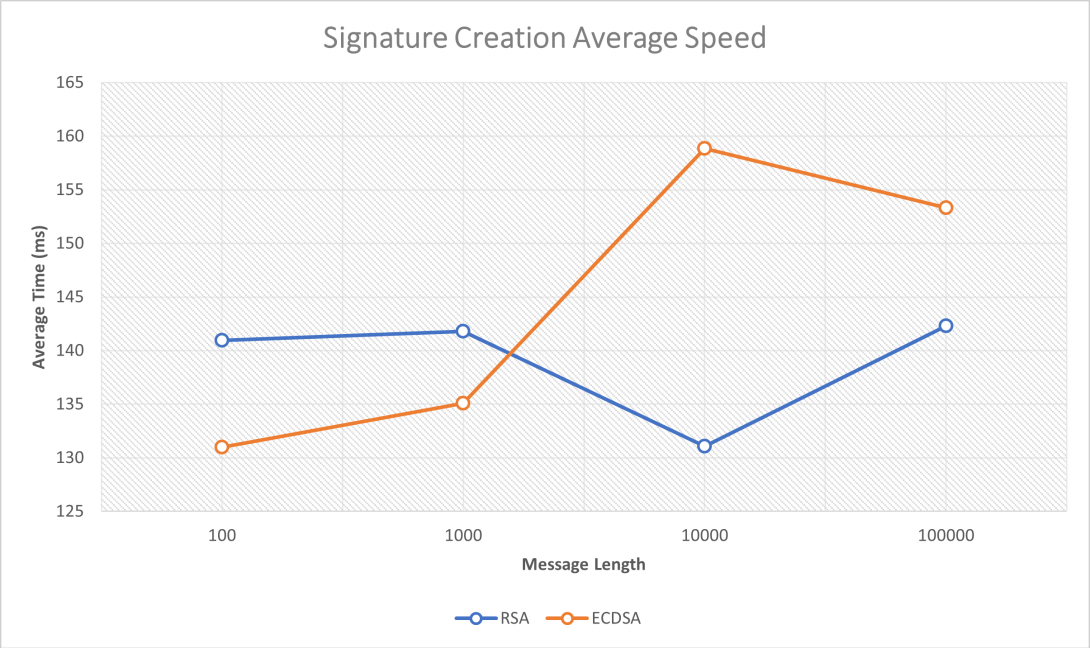

Figure 6.2: Signature Creation Average Speed

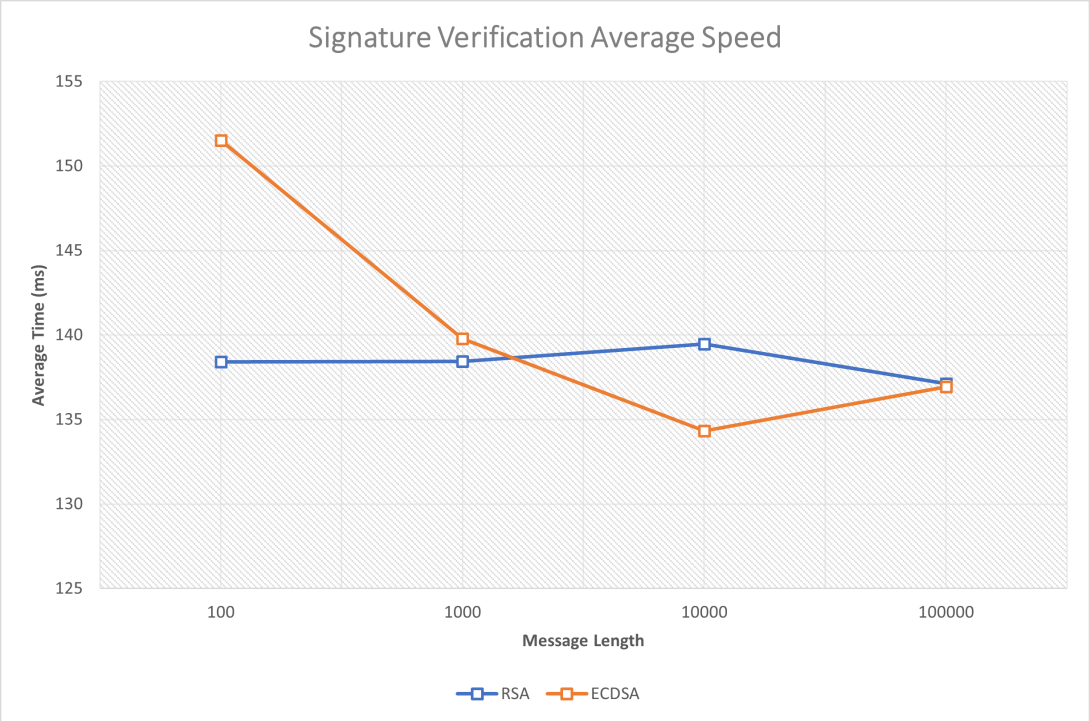

Figure 6.3: Signature Verification Average Speed

Table 6.2: Average Speed of Operations Applied to Video in Base64 Format

Operation	**Algorithm**	**Time (ms)**
Sign	RSA	1102
Sign	ECDSA	881
Verify Signature	RSA	963
Verify Signature	ECDSA	150

- **Key Generation -** (cf. Table 6.1) ECDSA significantly outperforms RSA in key generation, taking only a fraction of the time required by RSA.
- **Signature Generation and Verification -** (cf. Fig. 6.2 and 6.3) The performance of RSA and ECDSA varies depending on the message length. For shorter messages, ECDSA demonstrates slightly faster signature generation, while RSA exhibits slightly faster signature verification. However, the speed difference observed with messages until 100,000 characters is negligible.
- **Video Conversion -** (cf. Table 6.2) ECDSA significantly outperforms RSA in both signing and verifying signatures for large datasets, such as video conversions to Base64.

As expected, RSA signature processes were slower for larger messages in the tests. This can be attributed to the computational process known as exponentiation, which is a fundamental step in RSA signature generation. As the size of the input message grows, the exponentiation operation becomes more computationally intensive, resulting in increased processing time. In contrast, ECDSA, being based on elliptic curve cryptography, does not exhibit the same performance decline with larger messages, as it operates on a fixed-size elliptic curve and does not involve the same level of exponentiation complexity.

7

Conclusions and Future Work

The integration of identity verification and non-repudiation features has bolstered the security of the platform, as they are the foundations for robust mechanisms to verify user identities and ensure non-repudiable actions within the system.

However, the introduction of these advanced security measures has not been without its considerations. The additional steps for identity verification and non-repudiation may introduce some complexities and may potentially affect the overall user experience. Striking the right balance between robust security and user-friendly interactions has been an ongoing challenge.

Initial user feedback, from user testing, has been positive, with users expressing their satisfaction with the user interface, but with room for improvement. It's worth noting that some features of our platform have already been made available to the public.

The performance tests conducted show that the ECDSA outperforms the RSA algorithm in terms of speed and efficiency. This has implications for future security enhancements within the ADDA platform and warrants closer examination in subsequent development phases.

Future Work In the context of future developments for the application, several areas have been identified for potential enhancement.

To bolster system resilience, measures should be taken to prevent potential DoS attacks, ensuring the platform remains accessible and stable even during high-demand scenarios.

User testing has provided valuable insights into the application's usability and functionality. While the overall feedback from participants was positive, the testing process uncovered valuable recommendations for improvement within the system. These identified enhancements include improving the signup process to make it more intuitive and user-friendly, and refining the user experience during article creation and editing. These insights will guide future development into creating a better user experience.

Furthermore, with the successful testing of video-to-Base64 conversion, there's a potential to incorporate file encryption features into the application. This would enable users to securely publish audios, images and videos, ensuring their integrity and authenticity. Considering the inclusion of these new file types, future work involves enhancing the WYSIWYG editor to ensure advanced editing capabilities for multimedia content.

References

[1] Comissão Comemorativa 50 Anos 25 de Abril. "Adda50". (2023), [Online]. Available: https://adda50.cnedu.pt/.

[2] Comissão Comemorativa 50 Anos 25 de Abril. "As comemorações dos 50 anos do 25 de abril". (2022), [Online]. Available: https://www.50anos25abril.pt/estrutura-missao.

[3] AlmaLinux OS Foundation. "Almalinux". (2023), [Online]. Available: https://almalinux.org/.

[4] IP Agência para a Modernização Administrativa. "Autenticação gov". (2023), [Online]. Available: https://www.autenticacao.gov.pt/.

[5] David Birch, *Identity is the New Money*. London Publishing Partnership, 2014, ISBN: 978-1-907994-35-5.

[6] IP Agência para a Modernização Administrativa. "Chave móvel digital". (2023), [Online]. Available: https://www.autenticacao.gov.pt/a-chave-movel-digital.

[7] Committee on National Security Systems, *National information assurance (ia) glossary, CNSS Instruction No. 4009*, Apr. 2010. [Online]. Available: https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf.

[8] INSTICC. "10th international conference on information systems security and privacy". (2023), [Online]. Available: https://icissp.scitevents.org/.

[9] Phichayaphak Phiphitphatphaisit and Kulthida Tuamsuk, "Corporate social network design for knowledge sharing within university's operation", in *2016 5th International Conference on Computer Science and Network Technology (ICCSNT)*, 2016, pages 862–866. DOI: 10.1109/ICCSNT.2016.8070283.

[10] Jason Chia, Ji-Jian Chin, and Sook-Chin Yip, "Digital signature schemes with strong existential unforgeability", *F1000Research*, 2021. [Online]. Available: https://f1000research.com/articles/10-931.

[11] django wiki. "Django-wiki". (2023), [Online]. Available: https://github.com/django-wiki/django-wiki.

[12] Instituto de Gestão Financeira da Educação I.P. "Dsti - departamento de sistemas e tecnologias de informação". (2023), [Online]. Available: https://www.igefe.mec.pt/Page/Index/19?csrt=2356725830516656642.

[13] Matthew Green. "Euf-cma and suf-cma". A Few Thoughts on Cryptographic Engineering. (2023), [Online]. Available: https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/.

[14] Meta. "Facebook". (2023), [Online]. Available: https://www.facebook.com/.

[15] Anna Cinzia Squicciarini, Christopher Griffin, and Smitha Sundareswaran, "Towards a game theoretical model for identity validation in social network sites", in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pages 1081–1088. DOI: 10.1109/PASSAT/SocialCom.2011.208.

[16] Git. "Git". (2023), [Online]. Available: https://git-scm.com/.

[17] Inc. GitHub. "Github". (2023), [Online]. Available: https://github.com/.

[18] Google. "Gmail". (2023), [Online]. Available: https://mail.google.com/.

[19] Django Software Foundation and individual contributors. "Django - translation". (2023), [Online]. Available: https://docs.djangoproject.com/en/4.2/topics/i18n/translation/.

[20] W. Caelli, D. Longley, and M. Shain, *Information Security Handbook*. 1991.

[21] Jordan Russel. "Inno setup". (2023), [Online]. Available: https://jrsoftware.org/isinfo.php.

[22] J. Efrim Boritz, *International Journal of Accounting Information Systems*. 2005.

[23] MediaWiki. "Mediawiki". (2023), [Online]. Available: https://www.mediawiki.org/.

[24] Whitfield Diffie and Martin E. Hellman, "New directions in cryptography", 1976.

[25] Jakob Nielsen and Thomas K. Landauer, "A mathematical model of the finding of usability problems", in *Proceedings of ACM INTERCHI'93 Conference*, Amsterdam, The Netherlands, 1993, pages 206–213.

[26] Microsoft. "Outlook". (2023), [Online]. Available: https://outlook.office.com/mail/.

[27] Christof Paar and Jan Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, 2010.

[28] Kristian Beckers, *Pattern and Security Requirements: Engineering-Based Establishment of Security Standards*. 2015.

[29] Persona. "Persona". (2023), [Online]. Available: https://withpersona.com/.

[30] Phillip J. Windley, *Digital Identity*, 1st ed. O'Reilly Media, Aug. 2005, ISBN: 0-596-00878-3.

[31] Tibor Jager, Saqib A. Kakvi, and Alexander May, "On the security of the pkcs#1 v1.5 signature scheme", Sep. 2018. [Online]. Available: https://eprint.iacr.org/2018/855.pdf.

[32] Inc. Postman. "Postman". (2023), [Online]. Available: https://www.postman.com/.

[33] PyInstaller. "Pyinstaller". (2023), [Online]. Available: https://pyinstaller.org/.

[34] Python Software Foundation. "Python". (2023), [Online]. Available: https://www.python.org/.

[35] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, pages 120–126, 1978. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/359340.359342.

[36] The OpenSSL Project Authors. "Rsa-pss". (2023), [Online]. Available: https://www.openssl.org/docs/man3.0/man7/RSA-PSS.html.

[37] SendGrid. "Sendgrid". (2023), [Online]. Available: https://sendgrid.com/.

[38] Golden. "Sha-256". (2023), [Online]. Available: https://golden.com/wiki/SHA-256-XKEJ8AB.

[39] William Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson, 2017.

[40] Trello Enterprise. "Trello". (2023), [Online]. Available: https://trello.com/.

[41] Inc. iXsystems. "Truenas". (2023), [Online]. Available: https://www.truenas.com/.

[42] Sam Vaknin, *The Six Sins of Wikipedia*. CreateSpace Independent Publishing Platform, 2007, ISBN: 978-1430312444. [Online]. Available: https://samvak.tripod.com/wikipedia.html.

[43] "Video from spie - the international society for optics and photonics". (2021), [Online]. Available: https://doi.org/10.1117%2F12.2266326.5459349132001.

[44] Microsoft. "Microsoft visual studio code". (2023), [Online]. Available: https://code.visualstudio.com/.

[45] Deane Barker, *Web Content Management: Systems, Features, and Best Practices*. O'Reilly, 2015.

[46] Wiki.js. "Wiki.js". (2023), [Online]. Available: https://js.wiki/.

[47] Wikipedia contributors. "Wikipedia". (2023), [Online]. Available: https://en.wikipedia.org/wiki/Main_Page.

[48] Wikiwand. "Wikiscanner". (2023), [Online]. Available: https://www.wikiwand.com/en/WikiScanner.

[49] Randall Mikkelsen, "Cia and fbi computers used for wikipedia edits", *Reuters*, Aug. 2007. [Online]. Available: https://www.wikiwand.com/en/WikiScanner.

[50] X Corp. "X". (2023), [Online]. Available: https://twitter.com/.

Version Control

Version control is an essential aspect of any software development project, providing a systematic way to track changes, collaborate with team members, and ensure the integrity of the codebase. For this project, version control plays a crucial role in managing the development of the ADDA platform and facilitating efficient collaboration among contributors.

To fulfill the version control requirements, the project team has chosen Git [16] as the preferred source control system. Git is a widely adopted distributed version control system known for its speed, scalability and flexibility. It offers a comprehensive set of features that enable efficient branching and merging, facilitating concurrent development and parallel workflows.

In the context of creating the ADDA platform, Git's branching capabilities are particularly valuable. The team can create separate branches to work on specific features or modifications tailored to the unique requirements of the ADDA platform. This approach allows for isolated development and testing of these custom features without impacting the main codebase.

In addition to Git, the team has opted to use GitHub [17] as the interface for managing the project's repositories. GitHub provides a web-based platform that enhances collaboration, code review and issue tracking. It offers a seamless integration with Git, allowing team members to contribute, review and manage code changes in a centralized and accessible manner.

B

Functions and System Attributes' Relationship

Table B.1: Functions and System Attributes Relationship

Attribute	**Functions**
Platform Compatibility	R1.1, R1.2, R1.3, R1.4, R2.1, R3.1, R3.2, R3.3, R3.4, R4.1, R4.2, R4.3, R4.4, R4.5, R4.6, R4.7, R5.1
Performance	R4.5, R4.6, R4.7
Scalability	R1.1, R1.4, R4.1, R4.2, R4.5, R4.6, R4.7
Security	R1.1, R1.2, R1.3, R1.4, R2.1, R3.3, R5.1
Usability	R1.1, R1.2, R1.3, R2.1, R3.1, R3.2, R3.3, R3.4, R4.1, R4.2, R4.3, R4.4, R4.5, R4.6, R4.7
Reliability	R1.1, R1.2, R1.3, R4.1, R4.2, R4.4, R4.5, R4.6, R4.7
Language Support	R1.1, R1.2, R1.3, R2.1, R3.1, R3.2, R3.3, R3.4, R4.1, R4.2, R4.3, R4.4, R4.5, R4.6, R4.7
Support	R3.3, R3.4

C

Relational Model Diagrams

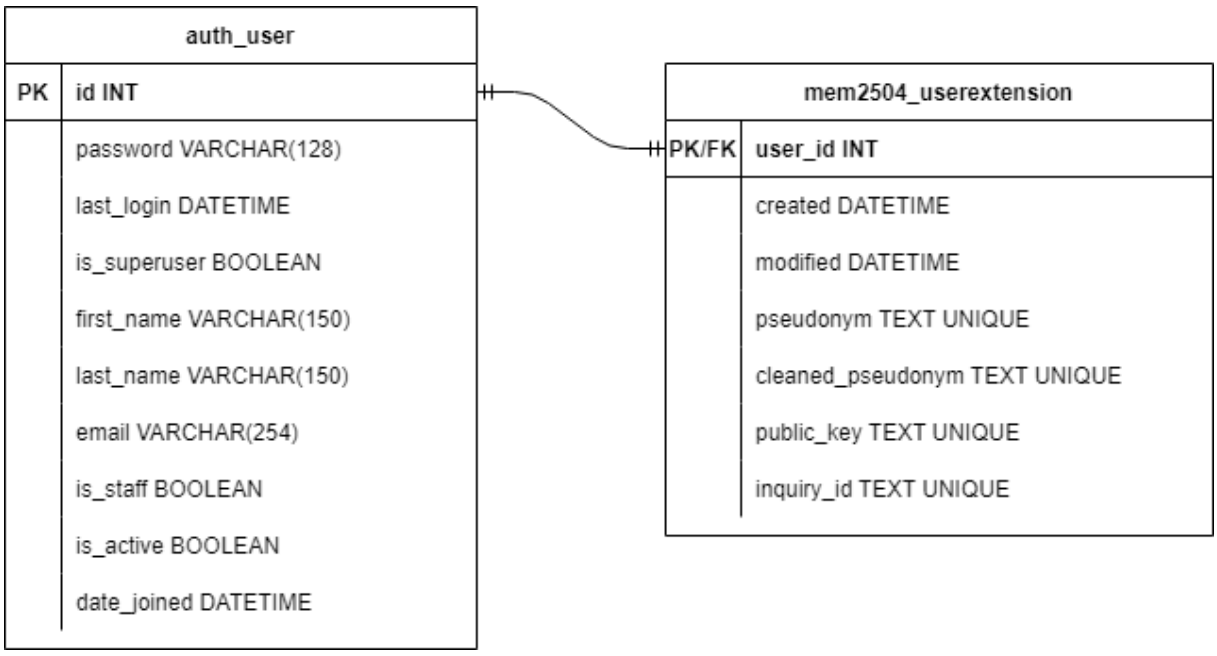

Figure C.1: Relation Model Diagram - User related Tables

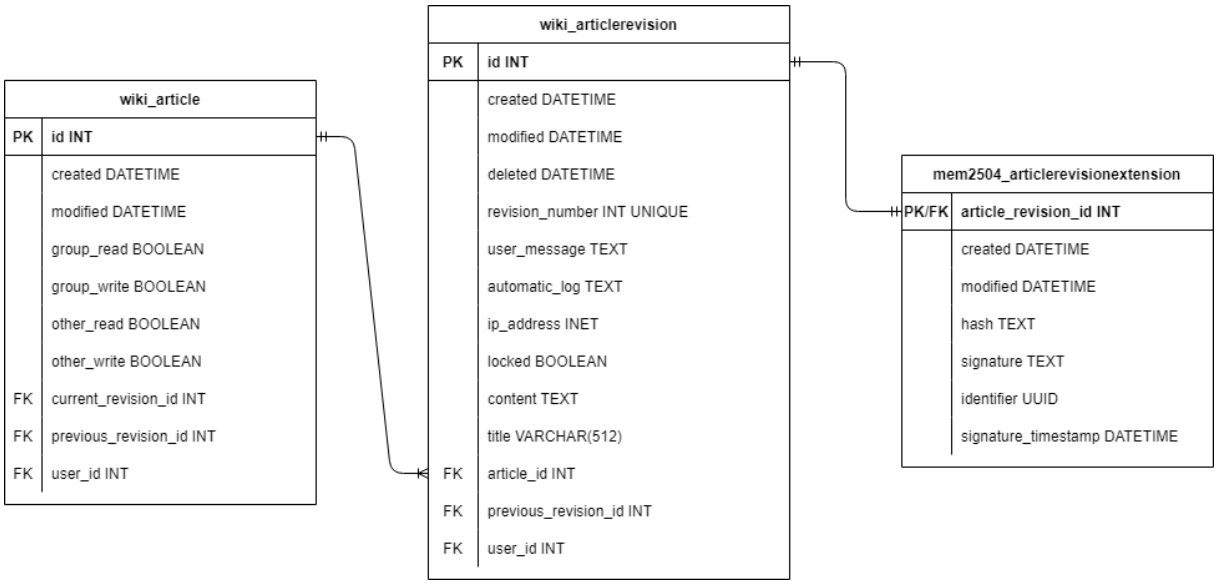

Figure C.2: Relation Model Diagram - Article related Tables

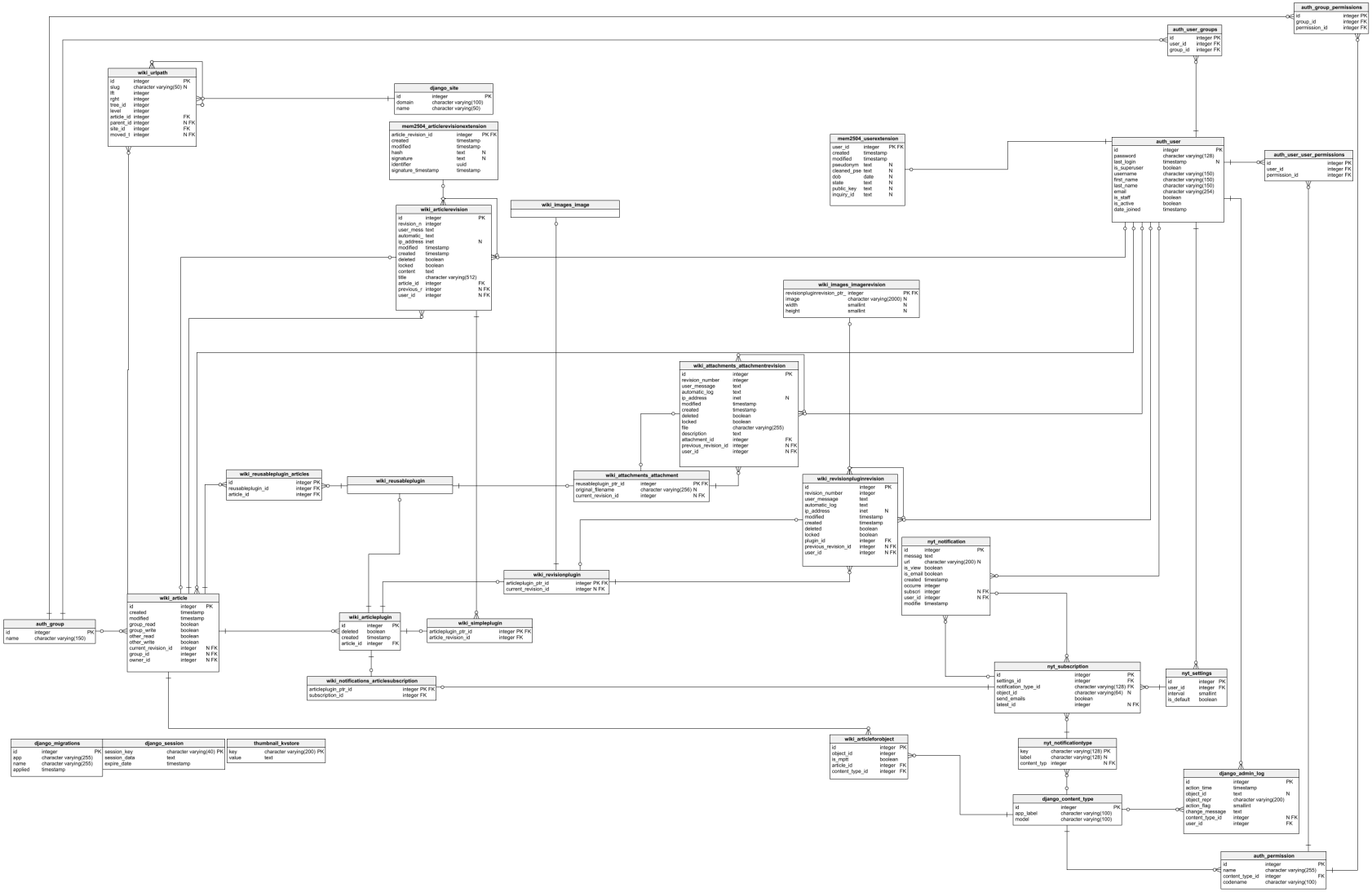

Figure C.3: Relational Model Diagram - All Tables

D

Use Cases

Table D.1: Use Case - Sign In

UC3	
Name	Sign In
Summary	The user provides their credentials, e-mail and password, if both fields are valid, the user is then directed to the home page.
References	R1.3, R1.4

Table D.2: Use Case - Sign Out

UC4	
Name	Sign Out
Summary	The user initiates the sign out process by pressing the $< SignOut >$ button. The system performs the necessary actions to end the user session. Upon successful sign out, the user is redirected to the home page as an anonymous user.
References	R1.2, R1.4

Table D.3: Use Case - Create Root Article

UC5	
Name	Create Root Article
Summary	The root article, serving as the primary wiki article and acting as the homepage, follows a creation process very similar to that of UC2 (Table 4.8). The only difference is that creating a root article is restricted to administrator users.
References	R4.1, R4.2, R5.1

Table D.4: Use Case - Edit Article

UC6	
Name	Edit Article
Summary	The user navigates to the desired article and presses the $< EditArticle >$ button. The system allows the user to modify the existing content, including the title, text and any attached images. The user can make the desired changes and then clicks on the $< Save >$ button to update the article. The updated version of the article is then made available to others.
References	R4.4, R5.1

Table D.5: Use Case - View Article

UC7	
Name	View Article
Summary	The user navigates to the desired article and clicks on it to view its content. The system retrieves and displays the article's title and content, including any incorporated images. The user can scroll through the article to read its content and view any multimedia elements.
References	R4.5

Table D.6: Use Case - View Article Versions

UC8	
Name	View Article Versions
Summary	The user navigates to the desired article page and clicks on the $< History >$ button. The system redirects the user to the History page, displaying a list of all the previous versions of the article. The user can select any version from the list to view its title and content. By choosing a specific version, the user can compare it with the current version or review the article's past content.
References	R4.6

Table D.7: Use Case - Change Article Version

UC9	
Name	Change Article Version
Summary	The user follows the steps of the $< ViewArticleVersions >$ use case by navigating to the history page and selecting the desired version. Upon selecting a version from the list, the user clicks on the version to access its details. On the version details page, the user can click on the $< ChooseThisVersion >$ button to replace the current version of the article with the selected version. This action updates the article's content and reverts it to the chosen version.
References	R4.7

Table D.8: Use Case - Search Article

UC10	
Name	Search Article
Summary	The user can utilize the system's search bar by entering keywords or phrases. The search will include both the article titles and their content. After the search is completed the system will present a list o results that match the search criteria, allowing the user to browse and access the relevant articles.
References	R4.5

Table D.9: Use Case - Change Password

UC11	
Name	Change Password
Summary	The user can access the user configuration page, where he has the option to edit his information. On the user configuration page, he can find the $< ChangePassword >$ section. After filling in the *NewPassword* and *OldPassword* fields and submitting the form, the system will perform a validation. If everything is correct, the user can log in using the newly specified password.
References	R3.3

Table D.10: Use Case - Recover Password

UC12	
Name	Recover Password
Summary	When a user forgets his password, he can initiate the password recovery process by visiting the login page and clicking on the $< ForgotYourPassword >$ button. Subsequently, he will be directed to a screen where he is prompted to enter his registered email address. If the provided email corresponds to an existing account, an email containing further instructions will be sent to that address. Upon receiving the email, the user can click on the provided link, which will redirect him to a dedicated form. In this form, he will have the opportunity to define a new password. After submitting the new password, the system will perform a validation check. If the entered password meets the necessary criteria the password will be successfully updated.
References	R3.4

Table D.11: Use Case - Edit Pseudonym

UC13	
Name	Edit Pseudonym
Summary	Upon going to the user configuration page, the user can locate the pseudonym field and make the necessary changes. After editing the pseudonym, the user can save the changes, and the updated pseudonym will be reflected in any published/edited articles.
References	R3.2