ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΈΤΗ, ΑΝΆΛΥΣΗ ΚΑΙ ΥΛΟΠΟΊΗΣΗ BLOCKCHAIN ΣΥΣΤΉΜΑΤΟΣ ΔΙΑΧΕΊΡΙΣΗΣ ΠΡΌΣΒΑΣΗΣ ΧΡΗΣΤΏΝ ΓΙΑ ΤΗΝ ΠΡΟΣΤΑΣΊΑ ΙΔΙΩΤΙΚΌΤΗΤΑΣ ΜΙΚΡΏΝ ΠΑΙΔΙΏΝ ΚΑΙ ΚΛΙΝΙΚΏΝ ΕΠΑΓΓΕΛΜΑΤΙΏΝ

Γεώργιος Λιόντος

Επιβλέπων: Βασιλική Λιάγκου

Επίκουρη, καθηγήτρια

Άρτα, Οκτώβριος, 2024

STUDY, ANALYSIS, AND IMPLEMENTATION OF A BLOCKCHAIN-BASED USER ACCESS MANAGEMENT SYSTEM FOR THE PRIVACY PROTECTION OF YOUNG CHILDREN AND CLINICAL PROFESSIONALS

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

Βασιλική Λιάγκου,

2. Μέλος επιτροπής

Χρυσόστομος Στύλιος,

3. Μέλος επιτροπής

Πέτρος Καρβέλης,

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Λιόντος, Γεώργιος

Υπογραφή

1. Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη ενός συστήματος διαχείρισης πρόσβασης χρηστών, που χρησιμοποιεί την τεχνολογία blockchain για την προστασία της ιδιωτικότητας. Το σύστημα στοχεύει στη διασφάλιση των ευαίσθητων προσωπικών δεδομένων μικρών παιδιών και κλινικών επαγγελματιών.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν σύγχρονα εργαλεία και τεχνολογίες όπως το IntelliJ IDEA για την ανάπτυξη του κώδικα, το Docker για τη διαχείριση απομονωμένων περιβαλλοντων(containers), και το Apache Maven για τη διαχείριση εξαρτήσεων του έργου. Η εφαρμογή βασίζεται στην Java SDK 1.8 για τη σταθερότητα στις κρυπτογραφικές βιβλιοθήκες, ενώ το **Node.js** υποστηρίζει την ανάπτυξη υψηλής απόδοσης του διακομιστή.

Η καινοτομία του συστήματος έγκειται στη χρήση της τεχνολογίας blockchain, που προσφέρει αυξημένη ασφάλεια και αποκεντρωμένη διαχείριση των δεδομένων. Αυτό το πλαίσιο διασφαλίζει ότι τα διαπιστευτήρια των χρηστών και οι διαδικασίες ταυτοποίησης παραμένουν ασφαλείς, επιτρέποντας παράλληλα την αδιάλειπτη λειτουργία της εφαρμογής σε περιβάλλοντα που απαιτούν αυστηρή προστασία των δεδομένων.

Λέξεις-κλειδιά: Διαχείριση πρόσβασης χρηστών, Προστασία ιδιωτικότητας, Αποκεντρωμένη ασφάλεια

ABSTRACT

The present thesis focuses on the development of a user access management system that utilizes blockchain technology to protect privacy. The system aims to safeguard the sensitive personal data of young children and clinical professionals.

For the development of the application, modern tools and technologies were used, such as IntelliJ IDEA for code development, Docker for managing isolated environments (containers), and Apache Maven for project dependency management. The application is based on Java SDK 1.8 to ensure stability in cryptographic libraries, while Node.js supports high-performance server development.

The innovation of the system lies in the use of blockchain technology, which offers enhanced security and decentralized management of data. This framework ensures that user credentials and authentication processes remain secure, while allowing uninterrupted operation of the application in environments requiring strict data protection.

Keywords: Blockchain, User access management, Privacy protection, ADHD diagnosis, Decentralized security

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ/ΕΙΚΟΝΩΝ

1. Εισαγωγή

Η παρούσα εργασία επικεντρώνεται στην ανάπτυξη ενός συστήματος διαχείρισης ταυτοτήτων που βασίζεται στην τεχνολογία blockchain, με έμφαση στην προστασία της ιδιωτικότητας και της ασφάλειας των χρηστών. Η ανάγκη για ασφαλή διαχείριση προσωπικών δεδομένων και η διαρκής απειλή από παραβιάσεις δεδομένων καθιστούν απαραίτητη την ανάπτυξη καινοτόμων λύσεων που υπερβαίνουν τις παραδοσιακές μεθόδους διαχείρισης ταυτοτήτων.

Μέσω της υλοποίησης του συστήματος διαχείρισης πρόσβασης, το οποίο επικεντρώνεται στην προστασία των δεδομένων των παιδιών αλλά και των κλινικών επαγγελματιών, αξιοποιείται η αποκεντρωμένη φύση του blockchain για την προστασία της ιδιωτικότητας και τη διασφάλιση της ακεραιότητας των δεδομένων. Στην εφαρμογή, τα διάφορα συστατικά στοιχεία και οι τεχνολογίες συμβάλλουν στην επίτευξη των λειτουργικών στόχων και την ασφάλεια του συστήματος, με κάθε εργαλείο να διαδραματίζει έναν συγκεκριμένο ρόλο. Το Spring Boot χρησιμοποιείται για την ανάπτυξη του backend, παρέχοντας το περιβάλλον για τη διαχείριση αιτημάτων και την εκτέλεση της επιχειρηματικής λογικής.

Το Angular, από την πλευρά του frontend, δημιουργεί δυναμικές και διαδραστικές διεπαφές χρήστη, βελτιώνοντας την εμπειρία χρήστη. Το Docker επιτρέπει την εκτέλεση της εφαρμογής σε απομονωμένα περιβάλλοντα μέσω containerization, εξασφαλίζοντας συνέπεια μεταξύ ανάπτυξης και παραγωγής. Το IntelliJ IDEA λειτουργεί ως το κύριο περιβάλλον ανάπτυξης, προσφέροντας εργαλεία για προγραμματισμό, διαχείριση κώδικα και εντοπισμό σφαλμάτων, ενώ το Git και το GitHub διευκολύνουν τη συνεργασία και την παρακολούθηση αλλαγών στον κώδικα, διασφαλίζοντας την ομαλή διαχείριση των εκδόσεων της εφαρμογής. Αυτές οι τεχνολογίες συνεργάζονται για να υποστηρίξουν τη λειτουργία και την ανάπτυξη ενός ολοκληρωμένου, ασφαλούς και ευέλικτου συστήματος.

1.1 Αντικείμενο και Στόχοι της Εργασίας: Σύντομη περιγραφή του τι αφορά η εργασία και ποιοι είναι οι στόχοι της εφαρμογής.

Το αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη και η ανάλυση ενός συστήματος διαχείρισης ταυτοτήτων, με σκοπό την προστασία της ιδιωτικότητας σε εφαρμογές που διαχειρίζονται ευαίσθητα προσωπικά δεδομένα. Το σύστημα διαχείρισης ταυτότητας χρησιμοποιεί την τεχνολογία blockchain για να παρέχει μια ασφαλή και ανώνυμη πλατφόρμα για τη διαχείριση ευαίσθητων δεδομένων παιδιών, διασφαλίζοντας την ιδιωτικότητα και την ανωνυμία των χρηστών.

1.2 Περιγραφή της Εφαρμογής: Μια γενική περιγραφή της λειτουργικότητας της εφαρμογής που αναπτύσσεις.

Διασφάλιση της ιδιωτικότητας: Χρήση ανώνυμων διαπιστευτηρίων και αποκεντρωμένης διαχείρισης ταυτοτήτων, που διασφαλίζουν ότι τα προσωπικά δεδομένα των χρηστών δεν αποκαλύπτονται σε τρίτους χωρίς τη συγκατάθεση τους.

Ασφάλεια των δεδομένων: Ενσωμάτωση κρυπτογραφικών τεχνικών, όπως οι υπογραφές δακτυλίου και οι αποδείξεις μηδενικής γνώσης (zkSNARKs), για την προστασία της ταυτότητας και των δεδομένων των χρηστών.

Ενίσχυση της αξιοπιστίας του συστήματος: Χρήση έξυπνων συμβολαίων για την αυτόματη διαχείριση της πρόσβασης και των δικαιωμάτων των χρηστών, εξασφαλίζοντας μια ασφαλή και αξιόπιστη πλατφόρμα για την ανάλυση των δεδομένων.

Υποστήριξη λήψης αποφάσεων: Παροχή εργαλείων που διευκολύνουν την ανάλυση και την παρουσίαση των αποτελεσμάτων με κατανοητό τρόπο, προσφέροντας καθοδήγηση για την κατάλληλη υποστήριξη των παιδιών.

2. Συστήματα Διαχείρισης Ταυτότητας με σεβασμό στην ιδιωτικότητα και ασφάλεια του χρήστη.

Τα συστήματα διαχείρισης ταυτότητας που βασίζονται στην τεχνολογία blockchain έχουν αναδειχθεί ως ισχυρές λύσεις για τη προστασία της ιδιωτικότητας και της ασφάλειας των χρηστών, ξεπερνώντας τις αδυναμίες των παραδοσιακών κεντρικών συστημάτων. Η αποκέντρωση που προσφέρουν , σε συνδυασμό με την αδιάβλητη φύση τους, επιτρέπει την αποτελεσματική διαχείριση δεδομένων και τη διασφάλιση της εμπιστευτικότητας των πληροφοριών. Για παράδειγμα, σε συστήματα που χρησιμοποιούν βιομετρικά δεδομένα ασθενών, το blockchain μπορεί να ενισχύσει την ασφάλεια και την προστασία των δεδομένων, αντιμετωπίζοντας προκλήσεις όπως η πρόσβαση, η επεκτασιμότητα και η εμπιστευτικότητα (Masood et al., 2024).

Παράλληλα, τα συστήματα ψηφιακής ταυτότητας που βασίζονται στο Blockchain (BDIS) παρέχουν έναν καινοτόμο τρόπο διαχείρισης της ταυτότητας των χρηστών, διαχωρίζοντας τις λειτουργίες της επαλήθευσης και της έκδοσης διαπιστευτηρίων, γεγονός που μειώνει τον κίνδυνο διαρροής πληροφοριών. Εφαρμόζοντας κρυπτογραφικές τεχνικές όπως η υπογραφές δακτυλίου και οι αποδείξεις μηδενικής γνώσης, τα συστήματα αυτά διασφαλίζουν την ιδιωτικότητα, ενώ παράλληλα παρέχουν δυνατότητες ελέγχου και ακύρωσης ταυτότητας (Song et al., 2024).

Τέλος, η χρήση έξυπνων συμβολαίων για τη διαχείριση των παραμέτρων του συστήματος και την επαλήθευση των ταυτοτήτων προσφέρει αυξημένη ασφάλεια, διαφάνεια και ανθεκτικότητα, ενισχύοντας περαιτέρω την αξιοπιστία αυτών των συστημάτων(Song et al., 2024)

2.1 Συστήματα διαχείρισης ταυτότητας με ανώνυμα διαπιστευτήρια

Τα ανώνυμα συστήματα διαχείρισης ταυτότητας έχουν προσελκύσει το ενδιαφέρον λόγω της δυνατότητας τους να διατηρούν την ιδιωτικότητα των χρηστών, ενώ παράλληλα διασφαλίζουν την ακεραιότητα και την ασφάλεια των διαπιστευτηρίων. Ένα παράδειγμα αποτελεί το Αποκεντρωμένο Ανώνυμο Σύστημα Φήμης (DARS). Το οποίο επιτρέπει στους χρήστες να χρησιμοποιούν διαφορετικά ψευδώνυμα για την αλληλεπίδρασή τους, διατηρώντας τη φήμη τους όμως συνδεδεμένη με ένα κοινό διακριτικό (token), μέσω κρυπτογραφικών μεθόδων όπως τα zkSNARKs και τα Merkle trees. Αυτό το σύστημα αποτρέπει τους κακόβουλους χρήστες από το να ξεκινούν από την αρχή, διασφαλίζοντας την ανωνυμία και προστατεύοντας από επιθέσεις φήμης (Rabah et al.,2024).

Παράλληλα το σύστημα ALLOSAUR χρησιμοποιεί κρυπτογραφικούς συσσωρευτές (accumulators) για την αποτελεσματική διαχείριση και επικαιροποίηση των ανώνυμων διαπιστευτηρίων. Σε αντίθεση με παραδοσιακά συστήματα όπου η επικαιροποίηση των διαπιστευτηρίων είναι δαπανηρή, το ALLOSAUR επιτυγχάνει ανώνυμη ενημέρωση με

χαμηλό κόστος επικοινωνίας και υπολογιστικής ισχύος,βελτιώνοντας σημαντικά την ταχύτητα επεξεργασίας χωρίς να θυσιάζει την ανωνυμία (Jaques et al., 2024). Αυτές οι προσεγγίσεις προσφέρουν λύσεις που διασφαλίζουν την ανωνυμία και την ασφάλεια των χρηστών στα σύγχρονα αποκεντρωμένα συστήματα διαχείρισης ταυτότητας.

2.2 Αποκεντρωμένα συστήματα διαχείρισης ταυτότητας με βάση την τεχνολογία BlockChain

Τα αποκεντρωμένα συστήματα διαχείρισης ταυτότητας που βασίζονται στο blockchain αποτελούν μια καινοτόμο λύση για την προστασία των προσωπικών δεδομένων. Η βασική αρχή αυτών των συστημάτων είναι η απουσία ενός κεντρικού διαχειριστή, με τη διαχείριση των ταυτοτήτων να πραγματοποιείται από το ίδιο το δίκτυο blockchain. Αυτό προσφέρει αυξημένη ανθεκτικότητα σε επιθέσεις, καθώς η παραβίαση ενός κόμβου δεν θέτει σε κίνδυνο ολόκληρο το σύστημα (Masood et al., 2024). Σύμφωνα με τη μελέτη των Masood et al. (2024), η χρήση blockchain σε συστήματα διαχείρισης ταυτότητας εξασφαλίζει ότι οι ταυτότητες των χρηστών παραμένουν ασφαλείς και ανώνυμες.

Το blockchain αποθηκεύει τα δεδομένα σε ένα μη αλλοιώσιμο και διαφανές δίκτυο, καθιστώντας σχεδόν αδύνατη την παραποίηση των δεδομένων χωρίς να γίνει αντιληπτό. Αυτό επιτρέπει τη δημιουργία ανθεκτικών και διαφανών συστημάτων που παρέχουν επίσημες εγγυήσεις ασφάλειας. Τα έξυπνα συμβόλαια (smart contracts) που ενσωματώνονται σε αυτά τα συστήματα επιτρέπουν την αυτόματη διαχείριση των ταυτοτήτων και την εκτέλεση προκαθορισμένων ενεργειών όταν πληρούνται συγκεκριμένες προϋποθέσεις.

Για παράδειγμα, τα έξυπνα συμβόλαια μπορούν να διαχειρίζονται αυτόματα την πρόσβαση των χρηστών σε μια υπηρεσία, ελέγχοντας την εγκυρότητα των διαπιστευτηρίων τους χωρίς ανθρώπινη παρέμβαση (Moreno et al., 2020). Οι κρυπτογραφικές τεχνικές, όπως οι zkSNARKs, επιτρέπουν στους χρήστες να αποδεικνύουν συγκεκριμένες ιδιότητες των ταυτοτήτων τους χωρίς να αποκαλύπτουν την πλήρη ταυτότητά τους, προσφέροντας έτσι υψηλό επίπεδο ανωνυμίας και προστασίας των προσωπικών δεδομένων. Αυτές οι τεχνικές είναι ιδιαίτερα χρήσιμες σε εφαρμογές που απαιτούν την προστασία της ιδιωτικότητας, όπως οι υπηρεσίες υγείας και οι πλατφόρμες κοινωνικής δικτύωσης (Song et al., 2024).

2.3 Το σύστημα OLYMPUS

Το σύστημα OLYMPUS είναι μια προηγμένη λύση στη διαχείριση ταυτοτήτων, επικεντρωμένη στην προστασία της ιδιωτικότητας των χρηστών μέσω της τεχνολογίας blockchain. Στη σύγχρονη ψηφιακή εποχή, η ασφαλής και ιδιωτική διαχείριση ταυτοτήτων αποτελεί κρίσιμη ανάγκη, καθώς τα παραδοσιακά συστήματα διαχείρισης ταυτότητας συχνά αποτυγχάνουν να διασφαλίσουν την ιδιωτικότητα και την ασφάλεια των χρηστών. Το OLYMPUS χρησιμοποιεί μια αποκεντρωμένη προσέγγιση για την ασφαλή επαλήθευση ταυτοτήτων, ελαχιστοποιώντας την εξάρτηση από κεντρικούς διαχειριστές και παρέχοντας ανώνυμα διαπιστευτήρια, τα οποία επιτρέπουν στους χρήστες να αποδεικνύουν την ταυτότητά τους χωρίς να αποκαλύπτουν ευαίσθητες πληροφορίες (Bernabe et al., 2021).

Οι κρυπτογραφικές τεχνικές που χρησιμοποιεί το σύστημα περιλαμβάνουν υπογραφές δακτυλίου και αποδείξεις μηδενικής γνώσης (zkSNARKs), οι οποίες επιτρέπουν την αυθεντικοποίηση των δεδομένων χωρίς την αποκάλυψη της ταυτότητας του χρήστη. Οι υπογραφές δακτυλίου επιτρέπουν την επιβεβαίωση ενός μηνύματος από έναν εκ των συμμετεχόντων χωρίς να αποκαλύπτεται ποιος συγκεκριμένα το υπέγραψε, ενώ οι αποδείξεις μηδενικής γνώσης επιτρέπουν την επιβεβαίωση μιας πληροφορίας χωρίς να αποκαλύπτονται τα πραγματικά δεδομένα (Song et al., 2024).

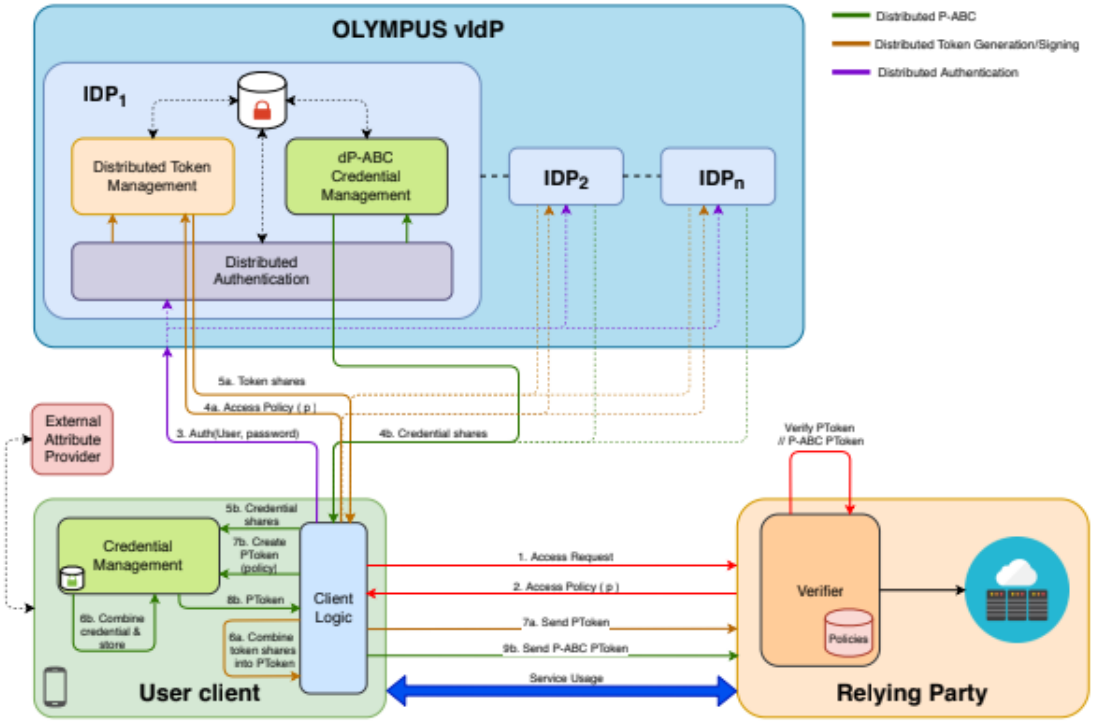
Αυτές οι τεχνικές προσφέρουν σημαντική προστασία της ιδιωτικότητας και της ασφάλειας στα σύγχρονα συστήματα διαχείρισης ταυτότητας. Επιπλέον, το σύστημα χρησιμοποιεί τεχνολογίες όπως οι κρυπτογραφικοί συσσωρευτές, οι οποίοι επιτρέπουν την ασφαλή και αποδοτική επικαιροποίηση των ανώνυμων διαπιστευτηρίων. Το σύστημα ALLOSAUR, για παράδειγμα, εφαρμόζει κρυπτογραφικούς συσσωρευτές για τη βελτίωση της ταχύτητας και της ανωνυμίας κατά τη διαχείριση των διαπιστευτηρίων (Jaques et al., 2024). Αυτή η προσέγγιση μειώνει τα κόστη επικοινωνίας και υπολογιστικής ισχύος, προσφέροντας έτσι μια πιο αποτελεσματική και ανώνυμη διαχείριση ταυτοτήτων.

2.4 Εφαρμογή του OLYMPUS σε εφαρμογή διαχείρισης πρόσβασης

Το σύστημα που στοχεύει στη διασφάλιση των ευαίσθητων προσωπικών δεδομένων μικρών παιδιών και κλινικών επαγγελματιών, αξιοποιεί τις δυνατότητες του OLYMPUS για την ασφαλή διαχείριση δεδομένων και ταυτοτήτων. Η πλατφόρμα επιτρέπει την ασφαλή καταγραφή και ανάλυση κλινικών δεδομένων μέσω ερωτηματολογίων και παρακολούθησης της συμπεριφοράς των παιδιών, χρησιμοποιώντας αλγόριθμους μηχανικής μάθησης για την ανάλυση των δεδομένων (Bernabe et al., 2021).

Το blockchain εξασφαλίζει ότι τα δεδομένα παραμένουν ασφαλή και αδιάβλητα, προστατεύοντας τα προσωπικά δεδομένα των χρηστών από πιθανές παραβιάσεις. Το σύστημα προσφέρει μια βελτιωμένη εμπειρία διαχείρισης ταυτοτήτων σε κλινικά περιβάλλοντα, ενισχύοντας την ασφάλεια και προστατεύοντας την ιδιωτικότητα των χρηστών, ειδικά όταν πρόκειται για ευαίσθητες ομάδες, όπως παιδιά και κλινικούς επαγγελματίες (Moreno et al., 2020).

Επιπλέον, οι γονείς και οι ειδικοί μπορούν να διαχειρίζονται τα δεδομένα των παιδιών μέσω της πλατφόρμας, που συμμορφώνεται πλήρως με τους κανονισμούς GDPR και προσφέρει επιλογές για τη διαχείριση των cookies και άλλων παραμέτρων ιδιωτικότητας.



Εικόνα 1 : Αρχιτεκτονική Olympus

2.5 Συστήματα Διαχείρισης Ταυτότητας και Εφαρμογές ή Ηλεκτρονικές Υπηρεσίες

Τα συστήματα διαχείρισης ταυτότητας (Identity Management Systems - IdM) αποτελούν βασικό εργαλείο για την ασφάλεια και τη διαχείριση των ταυτοτήτων των χρηστών σε σύγχρονες ηλεκτρονικές εφαρμογές και υπηρεσίες. Αυτά τα συστήματα παρέχουν τις μεθόδους και τις διαδικασίες για την επαλήθευση της ταυτότητας των χρηστών, την εξουσιοδότηση πρόσβασης σε πληροφορίες και την προστασία των προσωπικών δεδομένων τους. Με την αυξημένη χρήση διαδικτυακών υπηρεσιών, όπως οι τραπεζικές συναλλαγές, οι πλατφόρμες κοινωνικής δικτύωσης, οι υπηρεσίες υγείας και οι εφαρμογές e-commerce, η διαχείριση ταυτότητας γίνεται ολοένα και πιο κρίσιμη.

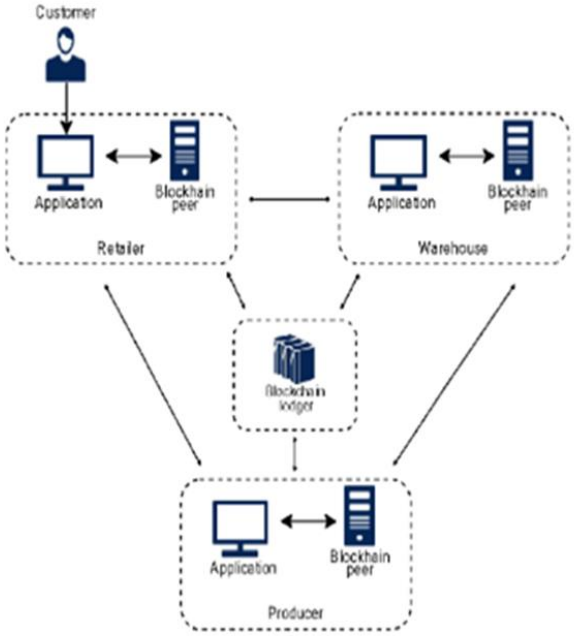
Οι κύριες λειτουργίες ενός συστήματος διαχείρισης ταυτότητας περιλαμβάνουν την καταγραφή, την ταυτοποίηση και την εξουσιοδότηση των χρηστών. Αυτό επιτρέπει τη διαχείριση των διαπιστευτηρίων, όπως κωδικοί πρόσβασης, βιομετρικά δεδομένα ή άλλες μορφές ταυτοποίησης, ώστε να εξασφαλιστεί ότι μόνο οι εξουσιοδοτημένοι χρήστες έχουν πρόσβαση σε συγκεκριμένους πόρους (Masood et al., 2024). Αυτά τα συστήματα προστατεύουν τα δεδομένα των χρηστών από μη εξουσιοδοτημένη πρόσβαση, παραβιάσεις και υποκλοπές, εξασφαλίζοντας την ιδιωτικότητα και την ακεραιότητα των πληροφοριών.

2.5.1 Βασική αρχιτεκτονική συστήματος για την παροχή ηλεκτρονικών υπηρεσιών.

Η αρχιτεκτονική ενός συστήματος διαχείρισης ταυτότητας περιλαμβάνει μια σειρά από στοιχεία που συνεργάζονται για την ασφαλή διαχείριση των ταυτοτήτων και των προσβάσεων των χρηστών. Τα κύρια στοιχεία αυτής της αρχιτεκτονικής είναι:

- **Πάροχοι Υπηρεσιών Ταυτότητας (IdSPs):** Αυτοί οι πάροχοι είναι υπεύθυνοι για την επαλήθευση της ταυτότητας των χρηστών και την έκδοση διαπιστευτηρίων. Οι IdSPs διαδραματίζουν κεντρικό ρόλο στη διασφάλιση της ακεραιότητας των ταυτοτήτων και στην προστασία των δεδομένων από μη εξουσιοδοτημένες προσβάσεις (Bernabe et al., 2021). Οι πάροχοι αυτοί διαχειρίζονται τη διαδικασία ταυτοποίησης των χρηστών και εξασφαλίζουν ότι τα διαπιστευτήρια που εκδίδονται είναι αξιόπιστα και ασφαλή.
- **Αποκεντρωμένοι Πάροχοι Ταυτότητας (vIdPs):** Σε αντίθεση με τους παραδοσιακούς κεντρικούς παρόχους, οι αποκεντρωμένοι πάροχοι χρησιμοποιούν τεχνολογίες blockchain για την ασφαλή διαχείριση των ταυτοτήτων χωρίς την ανάγκη για έναν κεντρικό διαχειριστή. Αυτή η προσέγγιση μειώνει τους κινδύνους από παραβιάσεις δεδομένων και παρέχει αυξημένη ιδιωτικότητα και ασφάλεια στους χρήστες, καθώς τα δεδομένα τους δεν συγκεντρώνονται σε ένα μοναδικό σημείο αποθήκευσης (Moreno et al., 2020).
- **Κρυπτογραφικές Τεχνικές:** Οι κρυπτογραφικές μέθοδοι, όπως οι αποδείξεις μηδενικής γνώσης (zkSNARKs) και οι κρυπτογραφικοί συσσωρευτές, εξασφαλίζουν ότι οι χρήστες μπορούν να αποδεικνύουν την ταυτότητά τους χωρίς να αποκαλύπτουν ευαίσθητες πληροφορίες. Αυτές οι τεχνικές αυξάνουν την ασφάλεια και την ιδιωτικότητα, επιτρέποντας την ασφαλή επικαιροποίηση των διαπιστευτηρίων χωρίς να εκτίθεται η πραγματική ταυτότητα των χρηστών (Masood et al., 2024).
- **Έξυπνα Συμβόλαια (Smart Contracts):** Τα έξυπνα συμβόλαια αυτοματοποιούν τη διαχείριση των πολιτικών πρόσβασης και τη διαδικασία επαλήθευσης ταυτοτήτων. Επιτρέπουν την άμεση και αξιόπιστη διαχείριση των χρηστών και των προσβάσεων, μειώνοντας την ανάγκη για ανθρώπινη παρέμβαση και ελαχιστοποιώντας τα σφάλματα. Αυτή η αυτοματοποίηση αυξάνει την απόδοση και την ασφάλεια του συστήματος, κάνοντάς το πιο ανθεκτικό σε επιθέσεις και απειλές (Song et al., 2024).

Η αρχιτεκτονική ενός συστήματος διαχείρισης ταυτότητας είναι σχεδιασμένη για να εξασφαλίζει την ασφάλεια και την ιδιωτικότητα των χρηστών, δημιουργώντας ένα περιβάλλον που προστατεύει τις προσωπικές πληροφορίες και επιτρέπει την ασφαλή χρήση των ηλεκτρονικών υπηρεσιών.



Εικόνα 2 : Αρχιτεκτονική Συστήματος Διαχείρισης Πρόσβασης με Χρήση Blockchain

3. Τεχνολογίες και Πλαίσια για την Ανάπτυξη Διαδικτυακών Εφαρμογών

Οι τεχνολογίες και τα πλαίσια ανάπτυξης διαδικτυακών εφαρμογών παίζουν καθοριστικό ρόλο στη σύγχρονη ανάπτυξη λογισμικού, παρέχοντας στους προγραμματιστές εργαλεία και βιβλιοθήκες που διευκολύνουν την κατασκευή, τη συντήρηση και την κλιμάκωση εφαρμογών. Η επιλογή των κατάλληλων τεχνολογιών εξαρτάται από τις ανάγκες κάθε έργου, συμπεριλαμβάνοντας την απόδοση, την ασφάλεια, την εμπειρία χρήστη και την ευκολία ανάπτυξης.

Τα δημοφιλή πλαίσια ανάπτυξης(web frameworks), όπως το Spring Boot, το Angular, το React και το Django, ενσωματώνουν σύγχρονες αρχιτεκτονικές πρακτικές και προτυποποιημένες λύσεις που επιταχύνουν την ανάπτυξη και βελτιώνουν την ποιότητα των εφαρμογών. Με την εξέλιξη των απαιτήσεων της αγοράς και των τεχνολογιών, τα πλαίσια αυτά συνεχίζουν να προσαρμόζονται, προσφέροντας συνεχώς νέες δυνατότητες για την αντιμετώπιση των προκλήσεων της ανάπτυξης διαδικτυακών εφαρμογών.

3.1 Πλαίσια Ανάπτυξης Διαδικτυακών Εφαρμογών

Οι διαδικτυακές εφαρμογές είναι λογισμικά που εκτελούνται μέσω του διαδικτύου και παρέχουν υπηρεσίες στους χρήστες μέσω προγραμμάτων περιήγησης. Σε αντίθεση με τις παραδοσιακές εφαρμογές, οι διαδικτυακές εφαρμογές δεν απαιτούν εγκατάσταση στη συσκευή του χρήστη και μπορούν να προσφέρουν διαδραστικότητα και πρόσβαση από οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο.

Για την ανάπτυξη αυτών των εφαρμογών, χρησιμοποιούνται **πλαίσια ανάπτυξης (web frameworks)**, τα οποία προσφέρουν μια οργανωμένη προσέγγιση στην κατασκευή εφαρμογών ιστού. Τα δημοφιλή πλαίσια ανάπτυξης επιτρέπουν στους προγραμματιστές να αυτοματοποιούν πολλές διαδικασίες, όπως η διαχείριση δεδομένων, η επικοινωνία με τη βάση δεδομένων, η ασφάλεια και η δρομολόγηση αιτημάτων.

Οι διαδικτυακές εφαρμογές και τα δημοφιλή πλαίσια περιγράφονται παρακάτω (Django, Angular, React, Ruby on Rails, Spring Boot κ.ά.)

3.2 Σχεδιαστικές επιλογές

1) Django

Ένα ισχυρό πλαίσιο γραμμένο σε Python που προσφέρει πλήρη υποστήριξη για τη διαχείριση δεδομένων και την ασφάλεια των εφαρμογών. Είναι γνωστό για την ταχύτητά του στην ανάπτυξη εφαρμογών και την υποστήριξη που προσφέρει για την προστασία δεδομένων μέσω ενσωματωμένων μηχανισμών(MDPI).

2) Angular

Ένα πλαίσιο JavaScript που έχει αναπτυχθεί από την Google, το οποίο χρησιμοποιείται για τη δημιουργία δυναμικών και διαδραστικών εφαρμογών ιστού. Το Angular βασίζεται στην αρχιτεκτονική MVC και προσφέρει δυνατότητες όπως **two-way data binding** και **dependency injection**, γεγονός που το καθιστά ιδανικό για μεγάλες και σύνθετες εφαρμογές (Technology Rivers).

3) React

Μια βιβλιοθήκη JavaScript που αναπτύχθηκε από τη Facebook για τη δημιουργία δυναμικών περιβαλλόντων χρήστη (UIs). Η βασική φιλοσοφία του React είναι η δημιουργία components, δηλαδή ανεξάρτητων μονάδων κώδικα που μπορούν να επαναχρησιμοποιηθούν σε διαφορετικά σημεία της εφαρμογής (SpringerLink) (Technology Rivers).

4) Ruby on Rails

Ένα πλαίσιο που χρησιμοποιεί τη γλώσσα Ruby και προωθεί τη φιλοσοφία του "Convention over Configuration". Είναι γνωστό για την ταχύτητα και την απλότητα στην ανάπτυξη, καθώς και για τη δυνατότητα εύκολης επαναχρησιμοποίησης κώδικα (MDPI)(Technology Rivers).

5) Spring Boot

Βασισμένο στη γλώσσα Java, το Spring Boot είναι ιδανικό για την ανάπτυξη μεγάλων εφαρμογών και υπηρεσιών, όπως μικροϋπηρεσίες. Παρέχει αυτοματοποίηση και εργαλεία για τη γρήγορη κατασκευή και ανάπτυξη εφαρμογών και είναι ιδιαίτερα δημοφιλές για εφαρμογές μεγάλης κλίμακας που απαιτούν υψηλή ασφάλεια και απόδοση (SpringerLink)(SpringerLink).

Η επιλογή του κατάλληλου πλαισίου εξαρτάται από τις ανάγκες του εκάστοτε έργου και την κλίμακα της εφαρμογής. Τα παραπάνω πλαίσια προσφέρουν σημαντικές δυνατότητες για την κατασκευή ασφαλών, κλιμακούμενων και αξιόπιστων εφαρμογών ιστού.

3.3 Πλεονεκτήματα και Επιλογή του Spring Boot

Πλεονεκτήματα του Spring Boot

Το Spring Boot είναι ένα εξαιρετικά δημοφιλές πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών που προσφέρει σημαντικά πλεονεκτήματα για την ανάπτυξη λογισμικού, ιδιαίτερα σε εφαρμογές μεγάλης κλίμακας και σύνθετης αρχιτεκτονικής. Μερικά από τα κύρια πλεονεκτήματά του είναι τα εξής:

- **Απλοποίηση της Ανάπτυξης**: Το Spring Boot απλοποιεί τη διαμόρφωση των εφαρμογών, προσφέροντας προεπιλεγμένες ρυθμίσεις και πρότυπα (convention over configuration). Με αυτό τον τρόπο, οι προγραμματιστές μπορούν να επικεντρωθούν περισσότερο στη λογική της εφαρμογής και λιγότερο στις ρυθμίσεις του περιβάλλοντος ανάπτυξης (SpringerLink)(SpringerLink).
- **Αυτοματοποιημένο Build και Δοκιμές**: Με το Spring Boot, η αυτοματοποίηση των διαδικασιών build και test είναι πιο απλή. Οι δυνατότητες ενσωμάτωσης με εργαλεία όπως το Maven και το Gradle επιτρέπουν στους προγραμματιστές να διαχειρίζονται εξαρτήσεις και να εκτελούν δοκιμές εύκολα (MDPI)(Technology Rivers).
- **Υποστήριξη Μικροϋπηρεσιών (Microservices)**: Το Spring Boot είναι ιδιαίτερα κατάλληλο για την ανάπτυξη αρχιτεκτονικών μικροϋπηρεσιών, όπου διαφορετικές μονάδες μιας εφαρμογής εκτελούνται ανεξάρτητα. Η αρχιτεκτονική αυτή βελτιώνει την κλιμακωσιμότητα και την απόδοση των εφαρμογών, ειδικά σε μεγάλης κλίμακας συστήματα (SpringerLink) (SpringerLink).
- **Ευελιξία και Επεκτασιμότητα**: Το Spring Boot παρέχει εργαλεία και ενσωματωμένες λειτουργίες που επιτρέπουν την εύκολη ενσωμάτωση με άλλες τεχνολογίες, όπως βάσεις δεδομένων, συστήματα διαχείρισης ασφάλειας και εξωτερικά API. Επιπλέον, το Spring Boot μπορεί να επεκταθεί εύκολα ανάλογα με τις ανάγκες της εφαρμογής, χωρίς να απαιτείται σημαντική ανασύνταξη του κώδικα (MDPI).
- **Ενσωματωμένη Υποστήριξη για Ασφάλεια**: Το πλαίσιο Spring Security, το οποίο είναι ενσωματωμένο στο Spring Boot, παρέχει ισχυρές δυνατότητες για την προστασία των εφαρμογών, όπως η διαχείριση ταυτοποίησης και πρόσβασης χρηστών, και η προστασία από επιθέσεις CSRF (Cross-Site Request Forgery) και XSS (Cross-Site Scripting) (SpringerLink).

3.4 Γιατί επιλέχθηκε το Spring Boot για την παρούσα εργασία

Η επιλογή του Spring Boot για την παρούσα εργασία έγινε με βάση τα συγκεκριμένα πλεονεκτήματα που προσφέρει για την ανάπτυξη μεγάλων, κατανεμημένων και ασφαλών εφαρμογών. Το Spring Boot επιλέχθηκε για τους εξής λόγους:

- **Ευκολία Στην Ανάπτυξη**: Η απλοποιημένη διαδικασία διαμόρφωσης και η ελαχιστοποίηση των χειροκίνητων ρυθμίσεων που προσφέρει το Spring Boot επιτρέπουν την ταχύτερη και πιο αποδοτική ανάπτυξη της εφαρμογής. Οι προεπιλεγμένες ρυθμίσεις του πλαισίου καθιστούν τη δημιουργία νέων έργων απλή και γρήγορη, γεγονός που μειώνει το κόστος και τον χρόνο ανάπτυξης.
- **Υποστήριξη Μικροϋπηρεσιών**: Η εφαρμογή που αναπτύσσεται στο πλαίσιο αυτής της εργασίας απαιτεί τη διαχείριση μικροϋπηρεσιών, μια αρχιτεκτονική που υποστηρίζεται πλήρως από το Spring Boot. Αυτό επιτρέπει την εύκολη κλιμάκωση της εφαρμογής και τη διαχείριση μεμονωμένων υπηρεσιών ανεξάρτητα.
- **Ενσωμάτωση με Βάσεις Δεδομένων και Εξωτερικές Υπηρεσίες**: Το Spring Boot διευκολύνει την ενσωμάτωση με βάσεις δεδομένων και άλλες υπηρεσίες τρίτων. Αυτή η ευελιξία είναι ζωτικής σημασίας για την παρούσα εργασία, όπου απαιτείται επικοινωνία με πολλαπλές εξωτερικές πηγές δεδομένων και συστήματα.
- **Ασφάλεια**: Το Spring Boot παρέχει ενσωματωμένα εργαλεία για την ενίσχυση της ασφάλειας των εφαρμογών, όπως η ταυτοποίηση χρηστών και η προστασία από ευρέως γνωστές επιθέσεις. Αυτό ήταν ένας καθοριστικός παράγοντας για την επιλογή του, καθώς η ασφάλεια των δεδομένων των χρηστών αποτελεί προτεραιότητα στην παρούσα εργασία.

Η επιλογή του Spring Boot ήταν επίσης αποτέλεσμα της ενεργής κοινότητας υποστήριξης και της διαρκούς εξέλιξής του, που εξασφαλίζει ότι το πλαίσιο παραμένει σύγχρονο και αξιόπιστο.

3.5 Άλλα Πλαίσια και Τεχνολογίες

Υπάρχουν πολλά δημοφιλή πλαίσια και τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη διαδικτυακών εφαρμογών, εκτός από το Spring Boot. Τα πλαίσια αυτά προσφέρουν διαφορετικές δυνατότητες και είναι κατάλληλα για συγκεκριμένους τύπους έργων ανάλογα με τις απαιτήσεις της εφαρμογής και τη γλώσσα προγραμματισμού που χρησιμοποιείται.

Άλλα δημοφιλή πλαίσια και τεχνολογίες για ανάπτυξη εφαρμογών (Node.js, Flask, κ.ά.)

1) **Node.js**

Το Node.js είναι μια πλατφόρμα ανάπτυξης που βασίζεται στη γλώσσα JavaScript και χρησιμοποιείται κυρίως για την κατασκευή διακομιστών (back-end) και δικτυακών εφαρμογών. Το Node.js εκτελεί JavaScript στον διακομιστή και είναι ιδιαίτερα δημοφιλές για την ανάπτυξη εφαρμογών σε πραγματικό χρόνο, όπως τα chat και τα εργαλεία συνεργασίας. Το πλεονέκτημά του είναι η μη μπλοκαριστική αρχιτεκτονική I/O, που το καθιστά πολύ αποτελεσματικό για εφαρμογές με υψηλές απαιτήσεις απόδοσης (MDPI)(Technology Rivers).

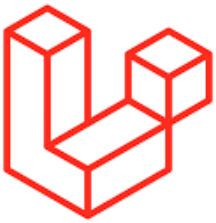
2) **Flask**

Το **Flask** είναι ένα μικρο-πλαίσιο για τη γλώσσα προγραμματισμού Python, το οποίο χρησιμοποιείται για την ανάπτυξη ελαφρών και εύκολα διαχειριζόμενων διαδικτυακών εφαρμογών. Παρέχει μεγάλη ευελιξία και ελευθερία στους προγραμματιστές, καθώς δεν επιβάλλει την υιοθέτηση συγκεκριμένων αρχιτεκτονικών ή εργαλείων. Το Flask είναι δημοφιλές για μικρές και μεσαίες εφαρμογές και θεωρείται εύκολο στη χρήση για νέους προγραμματιστές (SpringerLink)(SpringerLink).

3) **Express.js**

Το **Express.js** είναι ένα ελαφρύ πλαίσιο για τον Node.js και χρησιμοποιείται κυρίως για την κατασκευή εφαρμογών διαδικτύου και API. Το Express.js είναι γνωστό για την απλότητά του και τη δυνατότητά του να δημιουργεί πολύ γρήγορα RESTful υπηρεσίες. Εξαιτίας της αρχιτεκτονικής του, οι προγραμματιστές έχουν μεγάλη ελευθερία στον τρόπο που διαμορφώνουν την εφαρμογή τους, αλλά το πλαίσιο προσφέρει παράλληλα βασικά εργαλεία που καθιστούν τη διαδικασία ανάπτυξης αποδοτική (MDPI).

4) **Laravel**



Το **Laravel** είναι ένα πλαίσιο για τη γλώσσα PHP, που προσφέρει μια σειρά από ενσωματωμένα χαρακτηριστικά για τη διαχείριση των δεδομένων, την ασφαλή πρόσβαση χρηστών και τη σύνδεση με βάσεις δεδομένων. Χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών που βασίζονται σε διακομιστή, όπως ιστολόγια, ιστότοποι ηλεκτρονικού εμπορίου και άλλες εφαρμογές με βάση δεδομένων (Technology Rivers).

5) Vue.js

Το **Vue.js** είναι ένα άλλο JavaScript πλαίσιο που χρησιμοποιείται για την ανάπτυξη διαδραστικών διεπαφών χρήστη. Το Vue.js είναι γνωστό για την απλότητα και την ευελιξία του και επιτρέπει την ανάπτυξη επεκτάσιμων εφαρμογών που μπορούν να προσαρμοστούν στις ανάγκες του χρήστη. Σε αντίθεση με τα μεγαλύτερα πλαίσια, όπως το Angular, το Vue.js επικεντρώνεται αποκλειστικά στο front-end και ενσωματώνεται εύκολα με άλλες τεχνολογίες (Technology Rivers).

Η επιλογή του κατάλληλου πλαισίου εξαρτάται από τις ανάγκες της εφαρμογής, την κλίμακα του έργου και τη γλώσσα προγραμματισμού που προτιμάται. Οι σύγχρονες τεχνολογίες προσφέρουν ένα ευρύ φάσμα εργαλείων που διευκολύνουν την ανάπτυξη δυναμικών και ασφαλών εφαρμογών.

4. Τεχνολογίες Ανάπτυξης και Εργαλεία

4.1 Docker

Το Docker αποτελεί βασικό εργαλείο στην ανάπτυξη και διαχείριση σύγχρονων εφαρμογών, προσφέροντας δυνατότητες containerization, δηλαδή δημιουργία απομονωμένων περιβαλλόντων που περιέχουν όλες τις εξαρτήσεις της εφαρμογής. Αυτό επιτρέπει την εκτέλεση των εφαρμογών ανεξάρτητα από το υποκείμενο λειτουργικό σύστημα, βελτιώνοντας τη φορητότητα και την επαναληψιμότητα των περιβαλλόντων.

Τι είναι το Docker και πώς χρησιμοποιείται για την containerization:

Το Docker χρησιμοποιείται για τη δημιουργία, απομόνωση και διαχείριση containers, που είναι ελαφριά και φορητά πακέτα λογισμικού. Το Docker εκτελεί εφαρμογές με συνέπεια σε διαφορετικά περιβάλλοντα, από έναν τοπικό υπολογιστή ανάπτυξης έως έναν cloud server. Η χρήση των Dockerfiles αυτοματοποιεί τη διαδικασία κατασκευής containers, προσφέροντας μεγαλύτερη συνέπεια και μειώνοντας τα σφάλματα που προκύπτουν από ανθρώπινη παρέμβαση (Merkel, 2024). Η δυνατότητα εκτέλεσης πολλαπλών containers στον ίδιο server μειώνει την κατανάλωση πόρων σε σύγκριση με τα παραδοσιακά virtual machines, καθώς τα containers χρησιμοποιούν τον ίδιο πυρήνα του λειτουργικού συστήματος, κάτι που επιτρέπει ταχύτερη εκκίνηση και αποδοτικότερη διαχείριση πόρων (Pahl & Lee, 2024).

Πλεονεκτήματα του Docker στη διαχείριση εφαρμογών σε διαφορετικά περιβάλλοντα:

- **Κλιμάκωση και Αποδοτικότητα Πόρων:** Το Docker επιτρέπει την εύκολη προσαρμογή των εφαρμογών σε αυξημένες ή μειωμένες απαιτήσεις μέσω της προσθήκης ή αφαίρεσης containers, επιτρέποντας τη δυναμική κλιμάκωση χωρίς σημαντικές αλλαγές στην αρχιτεκτονική του συστήματος (Docker Inc., 2024).
- **Ενοποίηση με CI/CD και DevOps:** Το Docker είναι πλήρως ενσωματωμένο σε πρακτικές DevOps, διευκολύνοντας τη συνεχή ανάπτυξη, ολοκλήρωση και παράδοση λογισμικού. Η αυτοματοποίηση των διαδικασιών ανάπτυξης μειώνει τους χρόνους διάθεσης και διασφαλίζει τη συνεπή εκτέλεση των εφαρμογών από το στάδιο της ανάπτυξης έως την παραγωγή (Merkel, 2024).
- **Ασφάλεια και Απομόνωση:** Τα containers παρέχουν απομόνωση των εφαρμογών, περιορίζοντας την πρόσβαση στους πόρους του συστήματος και αυξάνοντας την ασφάλεια των δεδομένων. Η απομόνωση μειώνει την πιθανότητα παραβιάσεων και προστατεύει τις εφαρμογές από εξωτερικές επιθέσεις, κάτι που είναι κρίσιμο για πολυενοικιακά (multi-tenant) περιβάλλοντα, όπως αυτά που συναντώνται σε cloud υπηρεσίες (Docker Inc., 2024).
- **Φορητότητα και Επαναληψιμότητα:** Το Docker επιτρέπει τη μεταφορά containers μεταξύ διαφόρων πλατφορμών και περιβαλλόντων χωρίς αλλαγές, κάτι που διευκολύνει την ανάπτυξη σε πολλαπλές τοποθεσίες και βελτιώνει την επαναληψιμότητα των περιβαλλόντων ανάπτυξης και παραγωγής (Pahl & Lee, 2024).
- **Αυτοματοποιημένη Διαχείριση Ενημερώσεων και Αναπτύξεων:** Η χρήση εργαλείων όπως Docker Compose και Docker Swarm διευκολύνει τη διαχείριση πολλαπλών containers, επιτρέποντας την αυτοματοποιημένη ενημέρωση και ανάπτυξη εφαρμογών, μειώνοντας την ανάγκη για χειροκίνητη παρέμβαση και αυξάνοντας την αποτελεσματικότητα (Merkel, 2024).

4.2 Maven

4.2.1 Τι είναι το Maven και πώς χρησιμοποιείται για τη διαχείριση εξαρτήσεων και το χτίσιμο του project.

Το **Apache Maven** είναι ένα εργαλείο διαχείρισης έργων και αυτοματοποίησης build, που χρησιμοποιείται κυρίως στην ανάπτυξη εφαρμογών Java. Το Maven παρέχει μια οργανωμένη και δομημένη προσέγγιση για την ανάπτυξη λογισμικού, προσφέροντας ταυτόχρονα εργαλεία για τη διαχείριση εξαρτήσεων, την εκτέλεση τεστ, τη συσκευασία (packaging) και την ανάπτυξη (deployment). Στόχος του είναι η αυτοματοποίηση της διαδικασίας κατασκευής (build) και η μείωση των χειροκίνητων παρεμβάσεων από τους προγραμματιστές.

4.2.2 Διαχείριση Εξαρτήσεων με Maven

Η διαχείριση εξαρτήσεων είναι μια από τις κύριες λειτουργίες του Maven και βασίζεται στο `pom.xml` (Project Object Model), το οποίο περιέχει όλες τις πληροφορίες σχετικά με τις εξαρτήσεις που απαιτεί το έργο. Μια εξάρτηση είναι συνήθως μια βιβλιοθήκη (JAR file) που απαιτείται για τη λειτουργία του project. Όταν ένας προγραμματιστής προσθέτει μια εξάρτηση στο `pom.xml`, το Maven αναλαμβάνει αυτόματα να κατεβάσει τις απαραίτητες βιβλιοθήκες από αποθετήρια, όπως το **Maven Central**, και να τις ενσωματώσει στο έργο.

Το Maven διαχειρίζεται τρεις βασικούς τύπους αποθετηρίων:

- **Τοπικό Αποθετήριο (Local Repository)**: Βρίσκεται στον υπολογιστή του προγραμματιστή και αποθηκεύει τις βιβλιοθήκες που έχουν ήδη χρησιμοποιηθεί στο παρελθόν.
- **Κεντρικό Αποθετήριο (Central Repository)**: Είναι το προεπιλεγμένο αποθετήριο του Maven και περιέχει μια μεγάλη συλλογή από ανοιχτού κώδικα βιβλιοθήκες.
- **Απομακρυσμένα Αποθετήρια (Remote Repositories)**: Αποθετήρια που διαχειρίζονται τρίτα μέρη και είναι διαθέσιμα στο διαδίκτυο για τη λήψη πρόσθετων εξαρτήσεων (Apache Maven)(DEV Community).

Μέσω της χρήσης του **dependency management** στο `pom.xml`, το Maven διαχειρίζεται αυτόματα τις εξαρτήσεις και τις ενημερώνει αν χρειάζεται, χωρίς ο προγραμματιστής να ανησυχεί για χειροκίνητη λήψη και ενσωμάτωση. Επίσης, το Maven χρησιμοποιεί **transitive dependencies**, δηλαδή αν μια εξάρτηση χρειάζεται άλλες εξαρτήσεις, αυτές θα ενσωματωθούν αυτόματα στο έργο (ar5iv).

4.2.3 Διαδικασία Build με το Maven

Το Maven οργανώνει τη διαδικασία κατασκευής των έργων μέσα από μια σειρά προκαθορισμένων φάσεων που ανήκουν σε έναν κύκλο ζωής (build lifecycle). Ο κύκλος ζωής περιλαμβάνει τα εξής στάδια:

1. **compile**: Μεταγλώττιση του κώδικα.
2. **test**: Εκτέλεση των τεστ που έχουν οριστεί στο project.
3. **package**: Δημιουργία των εκτελέσιμων αρχείων (JAR ή WAR).
4. **install**: Εγκατάσταση του project στο τοπικό αποθετήριο.
5. **deploy**: Ανάπτυξη του έργου σε απομακρυσμένο αποθετήριο ή παραγωγικό περιβάλλον (Apache Maven).

Κάθε φάση μπορεί να αντιστοιχηθεί σε συγκεκριμένες ενέργειες μέσω **plugins**. Το Maven χρησιμοποιεί ένα αρχιτεκτονικό μοντέλο που βασίζεται σε **plugins**, με τα οποία επεκτείνεται η λειτουργικότητά του. Για παράδειγμα, με τη χρήση του **Surefire Plugin**, το Maven μπορεί να εκτελεί τεστ κατά τη διαδικασία build, και με το **Jar Plugin**, να συσκευάζει τον κώδικα σε εκτελέσιμα αρχεία (Proactive OSS Support).

4.2.4 Πλεονεκτήματα της αυτοματοποίησης που παρέχει το Maven.

1. Απλοποίηση της Διαχείρισης Εξαρτήσεων

Ένα από τα κύρια πλεονεκτήματα του Maven είναι η αυτοματοποίηση της διαχείρισης των εξαρτήσεων. Οι προγραμματιστές δεν χρειάζεται να αναζητούν χειροκίνητα βιβλιοθήκες και να ανησυχούν για προβλήματα συμβατότητας. Το Maven διασφαλίζει ότι όλες οι εξαρτήσεις και οι αλληλεξαρτήσεις (transitive dependencies) επιλύονται και ενσωματώνονται σωστά στο project (ar5iv)(Apache Maven).

2. Αυτοματοποίηση Επαναλαμβανόμενων Εργασιών

Το Maven διευκολύνει τις καθημερινές εργασίες ανάπτυξης λογισμικού μέσω της αυτοματοποίησης. Η διαδικασία build εκτελείται αυτόματα, χωρίς να απαιτείται χειροκίνητη παρέμβαση, εξοικονομώντας χρόνο και μειώνοντας την πιθανότητα σφαλμάτων. Με μία εντολή, όπως `mvn install`, μπορείς να διαχειριστείς την κατασκευή, τη σύνταξη και τις δοκιμές του έργου (Proactive OSS Support).

3. Συνέπεια και Επαναχρησιμοποίηση

Το Maven προωθεί τη χρήση δοκιμασμένων προτύπων, γεγονός που εξασφαλίζει τη συνέπεια ανάμεσα σε έργα που χρησιμοποιούν το ίδιο αρχείο `pom.xml`. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλες ομάδες προγραμματιστών, καθώς επιτρέπει τη γρήγορη δημιουργία έργων με παρόμοια αρχιτεκτονική (Apache Maven).

4. Ενσωμάτωση με CI/CD

Το Maven υποστηρίζει πλήρως τις διαδικασίες συνεχούς ενσωμάτωσης (CI) και συνεχούς παράδοσης (CD), επιτρέποντας την εύκολη ενσωμάτωση σε εργαλεία όπως το **Jenkins** ή το **GitLab CI**. Αυτό σημαίνει ότι οι διαδικασίες build, testing και deployment μπορούν να αυτοματοποιηθούν εξ ολοκλήρου, προσφέροντας ταχύτερη ανάπτυξη και παράδοση λογισμικού (Proactive OSS Support)(DEV Community).

5. Ευκολία στη Χρήση και Επέκταση

Το Maven ακολουθεί τη φιλοσοφία του **Convention over Configuration**, προσφέροντας προκαθορισμένες ρυθμίσεις και δομές που κάνουν την ανάπτυξη ευκολότερη. Παράλληλα, μπορεί να προσαρμοστεί στις ανάγκες κάθε έργου μέσω των plugins, επιτρέποντας την εύκολη ενσωμάτωση τρίτων εργαλείων και πλατφορμών (Apache Maven)(Proactive OSS Support).

4.3 IntelliJ IDEA

Τι είναι το IntelliJ και πώς διευκολύνει την ανάπτυξη εφαρμογών με Java.

Το **IntelliJ IDEA** είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που αναπτύχθηκε από την JetBrains και θεωρείται ένα από τα πιο ισχυρά εργαλεία για την ανάπτυξη εφαρμογών με Java. Προσφέρει μια σειρά από χαρακτηριστικά που ενισχύουν την παραγωγικότητα των προγραμματιστών, όπως έξυπνη βοήθεια στον κώδικα, αυτόματη συμπλήρωση (auto-completion), εντοπισμό λαθών σε πραγματικό χρόνο και προτάσεις για βελτίωση του κώδικα. Αυτά τα χαρακτηριστικά επιτρέπουν στους προγραμματιστές να εργάζονται πιο γρήγορα και με λιγότερα λάθη, ενώ διατηρούν την ποιότητα του κώδικα υψηλή.

Ένα από τα μεγαλύτερα πλεονεκτήματα του IntelliJ IDEA είναι η **ενσωμάτωση με εργαλεία διαχείρισης εξαρτήσεων**, όπως το Maven και το Gradle. Το IntelliJ επιτρέπει την αυτόματη διαχείριση των εξαρτήσεων και των διαδικασιών build, κάτι που βοηθά τους προγραμματιστές να επικεντρωθούν στη συγγραφή κώδικα και όχι στη διαμόρφωση των έργων (DZone)(Cyberogism). Επίσης, το IntelliJ προσφέρει ισχυρά εργαλεία για την **πλοήγηση στον κώδικα** και την ανάλυση της ροής δεδομένων, διευκολύνοντας τους προγραμματιστές να κατανοήσουν μεγάλους κώδικες και σύνθετα έργα (Cyberogism).

Πλεονεκτήματα του IntelliJ ως ολοκληρωμένο περιβάλλον ανάπτυξης (IDE).

- **Έξυπνη Συμπλήρωση Κώδικα και Ανάλυση**: Το IntelliJ IDEA προσφέρει δυνατότητες έξυπνης συμπλήρωσης κώδικα, που βοηθούν τους προγραμματιστές να γράφουν κώδικα πιο γρήγορα και με λιγότερα σφάλματα. Επίσης, παρέχει συνεχή ανάλυση του κώδικα σε πραγματικό χρόνο και εντοπίζει πιθανά σφάλματα, προτείνοντας λύσεις πριν την εκτέλεση (Cyberogism)(DZone).
- **Ενσωμάτωση με Εργαλεία Build**: Το IntelliJ υποστηρίζει πλήρως εργαλεία build όπως το Maven και το Gradle, προσφέροντας ενσωματωμένα εργαλεία διαχείρισης εξαρτήσεων και αυτοματοποίησης build. Αυτό διευκολύνει την ανάπτυξη μεγάλων έργων και την αποφυγή λαθών κατά τη διαχείριση των εξαρτήσεων (Cyberogism).
- **Δοκιμαστικά Εργαλεία**: Το IntelliJ προσφέρει ενσωματωμένη υποστήριξη για πλαίσια δοκιμών όπως το JUnit και το TestNG, καθιστώντας εύκολη την εκτέλεση δοκιμών μέσα από το ίδιο το IDE, με τη δυνατότητα να βλέπεις τα αποτελέσματα σε πραγματικό χρόνο (CompareCamp.com).
- **Πλοήγηση στον Κώδικα**: Το IntelliJ IDEA διευκολύνει την πλοήγηση σε μεγάλους κώδικες με εργαλεία όπως το "Search Everywhere", που επιτρέπει στους προγραμματιστές να βρίσκουν γρήγορα κλάσεις, μεθόδους και μεταβλητές (Cyberogism).

- **Υποστήριξη για CI/CD**: Η ενσωμάτωση του IntelliJ με εργαλεία συνεχούς ενσωμάτωσης (CI) και συνεχούς παράδοσης (CD) διευκολύνει την αυτοματοποίηση των διαδικασιών build και deployment, καθιστώντας το ιδανικό για περιβάλλοντα DevOps (CompareCamp.com).

4.4 Cookies και Διαχείριση Δεδομένων Χρήστη

Τα cookies αποτελούν ένα σημαντικό εργαλείο στις διαδικτυακές εφαρμογές για τη διαχείριση των δεδομένων του χρήστη και την παροχή εξατομικευμένων υπηρεσιών. Είναι μικρά αρχεία κειμένου που αποθηκεύονται στη συσκευή του χρήστη από τον ιστότοπο που επισκέπτεται, με σκοπό την αποθήκευση πληροφοριών όπως οι προτιμήσεις, οι συνεδρίες σύνδεσης και άλλες παραμέτρους που διευκολύνουν τη διαδραστικότητα του χρήστη με την εφαρμογή.

Ωστόσο, εγείρονται ανησυχίες σχετικά με την ιδιωτικότητα και την ασφάλεια των δεδομένων που αποθηκεύονται στα cookies. Σύμφωνα με τον GDPR, οι ιστότοποι είναι υποχρεωμένοι να εξασφαλίζουν τη ρητή συγκατάθεση των χρηστών πριν την αποθήκευση cookies στις συσκευές τους.

Αυτό απαιτεί τη χρήση διαφανών μηχανισμών συγκατάθεσης και την παροχή πλήρους πληροφόρησης σχετικά με τα δεδομένα που συλλέγονται και τον τρόπο χρήσης τους (Pantelic et al., 2022). Επιπλέον, η ασφάλεια των cookies μπορεί να βελτιωθεί μέσω κρυπτογραφικών τεχνικών, όπως η χρήση HTTPS, για τη διασφάλιση ότι τα cookies δεν είναι ευάλωτα σε επιθέσεις όπως το session hijacking, εξασφαλίζοντας ότι οι πληροφορίες δεν θα είναι προσβάσιμες από κακόβουλους χρήστες (van Lieshout & Kolkman, 2020).

Τι είναι τα cookies και πώς χρησιμοποιούνται στις διαδικτυακές εφαρμογές

Τα **cookies** είναι αρχεία κειμένου που αποθηκεύονται στον υπολογιστή ή τη συσκευή του χρήστη όταν επισκέπτεται έναν ιστότοπο. Αυτά τα αρχεία αποθηκεύουν δεδομένα που χρησιμοποιούνται για την αναγνώριση του χρήστη σε μελλοντικές επισκέψεις και για τη βελτίωση της εμπειρίας περιήγησης. Τα cookies επιτρέπουν στους ιστότοπους να θυμούνται προτιμήσεις, όπως η γλώσσα, οι ρυθμίσεις της ιστοσελίδας, ή ακόμη και το περιεχόμενο του καλαθιού αγορών σε ένα ηλεκτρονικό κατάστημα. Η χρήση των cookies καθιστά δυνατή την εξατομίκευση της εμπειρίας του χρήστη, κάνοντάς την πιο ευχάριστη και λειτουργική (MDPI).

Στις διαδικτυακές εφαρμογές, τα cookies χρησιμοποιούνται για διάφορους λόγους, όπως:

Αποθήκευση των προτιμήσεων του χρήστη: Τα cookies επιτρέπουν στους ιστότοπους να θυμούνται τις ρυθμίσεις που έχει κάνει ο χρήστης κατά την επίσκεψή του, όπως η γλώσσα ή τα φίλτρα αναζήτησης.

Διαχείριση συνεδριών (sessions): Τα cookies διατηρούν τον χρήστη συνδεδεμένο κατά τη διάρκεια της περιήγησής του, επιτρέποντας συνεδρίες σύνδεσης να παραμένουν ενεργές ακόμα και αν ο χρήστης επιστρέψει αργότερα στον ιστότοπο.

Εξατομικευμένες διαφημίσεις: Πολλές εταιρείες χρησιμοποιούν cookies για να παρακολουθούν τη συμπεριφορά των χρηστών στο διαδίκτυο και να προβάλλουν διαφημίσεις που ανταποκρίνονται στα ενδιαφέροντά τους (SpringerLink)(ReadkonG).

Υπάρχουν δύο τύποι cookies:

Session cookies: Χρησιμοποιούνται για τη διάρκεια της περιήγησης και διαγράφονται μόλις κλείσει ο περιηγητής.

Persistent cookies: Αυτά παραμένουν στη συσκευή του χρήστη για μεγαλύτερο χρονικό διάστημα και χρησιμοποιούνται για την παρακολούθηση των προτιμήσεων σε επόμενες επισκέψεις.

Διαχείριση cookies και ασφάλεια προσωπικών δεδομένων

Η διαχείριση των cookies συνδέεται άμεσα με την ασφάλεια των προσωπικών δεδομένων των χρηστών. Τα cookies μπορούν να αποθηκεύσουν πληροφορίες που σχετίζονται με την ταυτότητα των χρηστών, τις προτιμήσεις τους και τη συμπεριφορά τους στον ιστότοπο. Αυτές οι πληροφορίες, εάν δεν διαχειρίζονται σωστά, μπορούν να αποτελέσουν κίνδυνο για την ιδιωτικότητα του χρήστη.
Η διαχείριση των cookies περιλαμβάνει την επιλογή και έλεγχο των τύπων cookies που επιτρέπεται να αποθηκευτούν στη συσκευή του χρήστη. Πολλοί ιστότοποι εφαρμόζουν **διαφανείς μηχανισμούς συγκατάθεσης**, όπου ο χρήστης μπορεί να επιλέξει αν θα αποδεχθεί ή θα απορρίψει τα cookies. Αυτή η διαδικασία, γνωστή ως **cookies consent**, απαιτεί τη σαφή ενημέρωση του χρήστη για το πώς χρησιμοποιούνται τα cookies και ποια δεδομένα συλλέγονται.
Η διασφάλιση της ασφάλειας των cookies απαιτεί την εφαρμογή **τεχνικών κρυπτογράφησης** για την αποτροπή μη εξουσιοδοτημένης πρόσβασης σε προσωπικά δεδομένα. Επίσης, οι προγραμματιστές μπορούν να εφαρμόσουν μεθόδους που περιορίζουν την πρόσβαση σε cookies μόνο από συγκεκριμένες εφαρμογές ή περιόδους λειτουργίας, εξασφαλίζοντας ότι τα δεδομένα παραμένουν ασφαλή και ιδιωτικά (SpringerLink)(MDPI).

Συμβατότητα με κανονισμούς όπως το GDPR και οι βέλτιστες πρακτικές

Ο **Γενικός Κανονισμός για την Προστασία Δεδομένων (GDPR)**, ο οποίος τέθηκε σε εφαρμογή στην Ευρωπαϊκή Ένωση το 2018, επέβαλε νέους κανόνες για τη χρήση των cookies και την προστασία των προσωπικών δεδομένων των χρηστών. Σύμφωνα με τον GDPR, οι ιστότοποι οφείλουν να εξασφαλίζουν την **ενημερωμένη συγκατάθεση** των χρηστών για την αποθήκευση cookies.

Οι ιστότοποι πρέπει να παρέχουν στον χρήστη σαφή και κατανοητή ενημέρωση για το πώς χρησιμοποιούνται τα cookies και ποια δεδομένα συλλέγονται. Επιπλέον, ο χρήστης έχει το δικαίωμα να απορρίψει ή να αποσύρει τη συγκατάθεσή του ανά πάσα στιγμή (MDPI)(ReadkonG). Οι **βέλτιστες πρακτικές** για τη χρήση cookies περιλαμβάνουν την ελαχιστοποίηση της συλλογής δεδομένων, τη χρήση cookies μόνο όταν είναι απαραίτητο και την αποθήκευση των δεδομένων για το ελάχιστο δυνατό διάστημα.

Επίσης, οι ιστότοποι πρέπει να διασφαλίζουν ότι τα δεδομένα που συλλέγονται μέσω cookies είναι επαρκώς προστατευμένα από μη εξουσιοδοτημένη πρόσβαση και δεν κοινοποιούνται σε τρίτα μέρη χωρίς τη συγκατάθεση του χρήστη. Συνοψίζοντας, η σωστή διαχείριση των cookies είναι απαραίτητη τόσο για τη βελτίωση της εμπειρίας χρήστη όσο και για την προστασία των προσωπικών δεδομένων. Η συμμόρφωση με κανονισμούς όπως το GDPR και η εφαρμογή βέλτιστων πρακτικών προστασίας της ιδιωτικότητας είναι κρίσιμη για τη δημιουργία ενός ασφαλούς και διαφανούς περιβάλλοντος για τους χρήστες του διαδικτύου (MDPI)(ReadkonG).

5. Σχεδιαστικές Επιλογές

Στην παρούσα ενότητα περιγράφονται οι σχεδιαστικές επιλογές που χρειάζονται για την υλοποίηση του συστήματος Διαχείρισης Πρόσβασης. Δημιουργήθηκε μια εφαρμογή για επιτραπέζιους υπολογιστές που αξιοποιεί το πλαίσιο Olympus για την παροχή αυθεντικοποίησης, βασισμένης στα διαπιστευτήρια βάσει χαρακτηριστικών (Privacy-preserving Attribute-Based Credentials).

Επίσης αξιοποιώντας την κρυπτοβιβλιοθήκη OLYMPUS, προχωρήσαμε σε τροποποίηση του πελάτη (Client) που συνεργαζόταν με την κλάση PestoIdPRESTConnection IdP, μετατρέποντάς τον σε PabcIdPRESTConnection IdP. Με αυτή την αλλαγή, καταφέραμε να προσαρμόσουμε και να λειτουργήσουμε τον Διαχειριστή Πιστοποιητικών, ώστε να μπορεί να αποθηκεύει τα διαπιστευτήρια στον πελάτη (Client). Κατά τη διαδικασία εγγραφής του χρήστη, λαμβάνει έναν κωδικό μιας χρήσης (OTP), ο οποίος του επιτρέπει να αποκτήσει πρόσβαση στο σύστημα.

Βήματα Υλοποίησης Συστήματος Διαχείρισης Πρόσβασης:

1. **Ανάπτυξη Εφαρμογής:**
    - Δημιουργήθηκε εφαρμογή για επιτραπέζιους υπολογιστές με στόχο την αυθεντικοποίηση.
2. **Εφαρμογή Privacy-Preserving Attribute-Based Credentials (PABCs):**
    - Υιοθετήθηκε το πλαίσιο Olympus για την αυθεντικοποίηση με προστασία της ιδιωτικότητας.
3. **Τροποποίηση Κρυπτογραφικής Βιβλιοθήκης:**
    - Ο πελάτης (Client) τροποποιήθηκε ώστε να αντικατασταθεί η συνεργασία με την κλάση **PestoIdPRESTConnection IdP** από την **PabcIdPRESTConnection IdP**.
4. **Προσαρμογή Διαχειριστή Πιστοποιητικών:**
    - Ο Διαχειριστής Πιστοποιητικών αναπροσαρμόστηκε ώστε να αποθηκεύει διαπιστευτήρια στον πελάτη (Client).
5. **Υλοποίηση Διαδικασίας Εγγραφής:**
    - Ο χρήστης λαμβάνει έναν κωδικό μιας χρήσης (OTP) κατά την εγγραφή, τον οποίο χρησιμοποιεί για πρόσβαση στο σύστημα.

Για την ορθή λειτουργία της εφαρμογής, ακολουθήσαμε τις μεθόδους που προορίζονται για τη ρύθμιση του pABC IdP και είναι σχεδιασμένες να καλούνται από τον αντίστοιχο πελάτη.

- **Perform OPRF (Username, Nonce, Cryptographic value, Token, TokenType):** εκτελεί μια κρυπτογραφική διαδικασία OPRF σε μια τιμή που

βασίζεται σε έναν ψευδοκωδικό (ο οποίος δημιουργείται με τη χρήση του κωδικού πρόσβασης του χρήστη). Αν υπάρχει ήδη ένας λογαριασμός για τον χρήστη με το συγκεκριμένο όνομα χρήστη, τότε το προαιρετικό Token του τύπου TokenType επικυρώνεται πριν την εκτέλεση της λειτουργίας OPRF. Ο χρήστης στη συνέχεια θα αξιοποιήσει την έξοδο αυτής της διαδικασίας για να (επαν)δημιουργήσει ένα ζεύγος κλειδιών υπογραφής, αποτελούμενο από δημόσιο και ιδιωτικό κλειδί.

- **Finish Registration (Username, Cookie, Public Key, Signature, Nonce, idProof)**: ολοκληρώνει την εγγραφή ενός χρήστη αποθηκεύοντας το δημόσιο κλειδί που σχετίζεται με το όνομα χρήστη και προσθέτοντας χαρακτηριστικά στο προαιρετικό idProof, εφόσον αυτά μπορούν να επαληθευτούν. Πριν από αυτό, η υπογραφή στο Nonce, το idProof και το Username πρέπει να επαληθευτεί χρησιμοποιώντας το δημόσιο κλειδί. Επιπλέον, είναι απαραίτητο να επιβεβαιωθεί ότι το Cookie είναι έγκυρο, δηλαδή ότι η φάση εγγραφής του χρήστη έχει ξεκινήσει. Το δημόσιο κλειδί θα συσχετιστεί με το όνομα χρήστη και θα χρησιμοποιείται για την επαλήθευση μελλοντικών προσπαθειών ταυτοποίησης.
- **AddAttributes (Username, Cookie, Nonce, Signature, idProof):** πιστοποιεί έναν ήδη υπάρχοντα χρήστη με βάση το όνομα χρήστη, ένα cookie συνεδρίας, και μια υπογραφή στο όνομα χρήστη, το Nonce και το idProof. Εάν ο έλεγχος ταυτότητας είναι επιτυχής, τότε τα χαρακτηριστικά που περιλαμβάνονται στο idProof προστίθενται στον λογαριασμό του χρήστη, υπό την προϋπόθεση ότι αυτά τα χαρακτηριστικά μπορούν να επαληθευτούν.
- **DeleteAttributes (Username, Cookie, Nonce, Signature, Attributes)**: πραγματοποιεί έλεγχο ταυτότητας για έναν ήδη υπάρχοντα χρήστη, βασιζόμενη στο όνομα χρήστη, ένα cookie συνεδρίας, και μια υπογραφή που περιλαμβάνει το όνομα χρήστη, το Nonce και τα συγκεκριμένα χαρακτηριστικά (Attributes). Αφού επιβεβαιωθεί η αυθεντικότητα του χρήστη, τα χαρακτηριστικά που έχουν καθοριστεί διαγράφονται από τον λογαριασμό του. Με αυτόν τον τρόπο, εξασφαλίζεται ότι μόνο επαληθευμένοι χρήστες μπορούν να τροποποιήσουν τα δεδομένα τους.
- **RefreshCookie (Cookie)** επικυρώνει ότι το παρεχόμενο cookie είναι ακόμα έγκυρο και, εφόσον ισχύει, επιστρέφει ένα νέο cookie με ανανεωμένο χρόνο ισχύος. Αυτό διασφαλίζει ότι η συνεδρία του χρήστη παραμένει ενεργή για μεγαλύτερο χρονικό διάστημα, χωρίς να απαιτείται εκ νέου σύνδεση.
- **Authenticate (Username, Cookie, Nonce, Signature, Policy):** πραγματοποιεί έλεγχο ταυτότητας ενός ήδη υπάρχοντος χρήστη, χρησιμοποιώντας το όνομα χρήστη, ένα cookie συνεδρίας, και μια υπογραφή που περιλαμβάνει το όνομα χρήστη, το Nonce και την καθορισμένη πολιτική (Policy). Η υπογραφή επαληθεύεται με το δημόσιο κλειδί που σχετίζεται με το συγκεκριμένο όνομα χρήστη. Αν η διαδικασία ελέγχου ταυτότητας ολοκληρωθεί επιτυχώς, κατασκευάζεται ένα διακριτικό (token), το οποίο επιστρέφεται στον χρήστη. Το

διακριτικό αυτό περιλαμβάνει τα αποθηκευμένα χαρακτηριστικά του χρήστη, σύμφωνα με τα κριτήρια που ορίζει η παρεχόμενη πολιτική. Η πολιτική καθορίζει ποια χαρακτηριστικά θα ενσωματωθούν στο διακριτικό, εάν υπάρχουν διαθέσιμα και επαληθευμένα.

Εκτός από τις παραπάνω μεθόδους, ο PABC IdP παρέχει επιπλέον λειτουργίες που συνέβαλαν σημαντικά στην υλοποίηση του προγράμματος. Αυτές οι μέθοδοι περιγράφονται παρακάτω:

- **Get dp-ABC Public Parameters:** Επιστρέφει τις δημόσιες παραμέτρους dp-ABC των διακομιστών, οι οποίες είναι απαραίτητες για τη διαδικασία αυθεντικοποίησης και τη χρήση των διαπιστευτηρίων.
- **GetCredential (Username, Cookie, Nonce, Signature):** Ελέγχει την ταυτότητα ενός ήδη υπάρχοντος χρήστη, χρησιμοποιώντας το όνομα χρήστη, ένα cookie συνεδρίας, και μια υπογραφή που περιλαμβάνει το όνομα χρήστη, το Nonce και την πολιτική. Η υπογραφή επαληθεύεται με το δημόσιο κλειδί που σχετίζεται με το όνομα χρήστη. Αν η διαδικασία ταυτοποίησης ολοκληρωθεί επιτυχώς, κατασκευάζεται ένα διαπιστευτήριο με βάση όλα τα αποθηκευμένα χαρακτηριστικά του χρήστη, το οποίο επιστρέφεται σε αυτόν.

5.1 Κύριες αρχιτεκτονικές διαδικασίες για την εφαρμογή διαχείρισης χρηστών

Το σύστημα μας περιλαμβάνει κατά κύριο λόγο τρεις φάσεις. Η φάση εγγραφής , η φάση παραγωγής(generation phase), κατά την οποία ένας εγγεγραμμένος χρήστης μπορεί να λάβει διακριτικά ή διαπιστευτήρια και τη φάση επαλήθευσης.

α) Η φάση Εγγραφής:

Η φάση εγγραφής επιτρέπει στον χρήστη να δημιουργήσει έναν νέο λογαριασμό στον κατανεμημένο πάροχο ταυτότητας (vIdP), χρησιμοποιώντας ένα όνομα χρήστη και έναν κωδικό πρόσβασης για την προστασία του λογαριασμού. Η διαδικασία αυτή είναι διαδραστική και περιλαμβάνει τη συνεργασία με όλους τους παρόχους ταυτότητας που απαρτίζουν τον vIdP. Κατά την εγγραφή, ο χρήστης μπορεί να υποβάλει προαιρετικά ένα σύνολο χαρακτηριστικών που έχει λάβει από εξωτερικό πάροχο, τα οποία ο vIdP επαληθεύει πριν τα αποδεχθεί και τα αποθηκεύσει. Με την ολοκλήρωση της διαδικασίας, ο χρήστης λαμβάνει ένα μήνυμα επιβεβαίωσης που επισημαίνει την επιτυχή δημιουργία του λογαριασμού.

β)Η φάση παραγωγής(generation phase):

Η φάση παραγωγής στο σύστημά μας ξεκινά όταν ο χρήστης επιθυμεί να χρησιμοποιήσει την υπηρεσία που παρέχεται από ένα τρίτο μέρος. Αυτή η διαδικασία πραγματοποιείται εκτός του πλαισίου της κρυπτοβιβλιοθήκης και έχει ως κύριο στόχο την κατανεμημένη παραγωγή και παρουσίαση της πιστοποίησης. Ο έλεγχος ταυτότητας ξεκινά όταν ο χρήστης εισάγει το όνομα χρήστη και τον κωδικό πρόσβασης στον vIdP, και μετά την επιτυχή ταυτοποίηση από τους παρόχους ταυτότητας, εκτελείται το κατανεμημένο πρωτόκολλο P-ABC, το οποίο συνδυάζει κρυπτογράφηση βασισμένη σε χαρακτηριστικά και τεχνολογίες προστασίας ιδιωτικότητας. Στη συνέχεια, ο χρήστης λαμβάνει ένα διαπιστευτήριο P-ABC που παράγεται κατανεμημένα, επιτρέποντάς του να αποκτήσει ένα διακριτικό πρόσβασης για τη χρήση μιας υπηρεσίας με συγκεκριμένη πολιτική πρόσβασης. Ο χρήστης μπορεί να διατηρήσει το διαπιστευτήριο ασφαλώς στο σύστημά του και να δημιουργήσει το διακριτικό πρόσβασης χωρίς την ανάγκη επικοινωνίας με τον vIdP, παρακάμπτοντας έτσι τα ενδιάμεσα βήματα.

γ)Η φάση Επαλήθευσης:

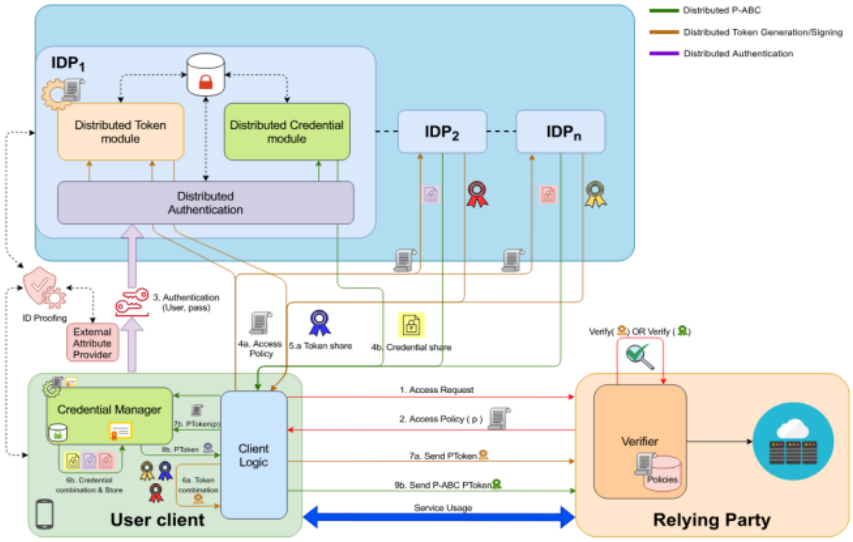
Η φάση επαλήθευσης είναι υπεύθυνη για τον έλεγχο της εγκυρότητας των υποβαλλόμενων διακριτικών πρόσβασης, περιλαμβάνοντας την επαλήθευση των υπογραφών και την αξιολόγηση της καταλληλότητάς τους για μια συγκεκριμένη πολιτική πρόσβασης. Ο επαληθευτής μπορεί να εφαρμόσει τα κατανεμημένα P-ABC, τα οποία προέρχονται από τα διαπιστευτήρια του χρήστη, ώστε να εξασφαλίσει την ασφάλεια και την τήρηση των προϋποθέσεων πρόσβασης.

Στη συνέχεια, θα περιγραφούν οι σχεδιαστικές επιλογές για τα διαπιστευτήρια βάσει χαρακτηριστικών απορρήτου (P-ABC) και οι τρόποι με τους οποίους οι πάροχοι ταυτότητας (IdPs) συμβάλλουν στην εφαρμογή διαχείρισης χρηστών και την πιστοποίηση.

5.2 Βασικά συστήματα για την υποστήριξη της εφαρμογής διαχείρισης χρηστών

Το σχήμα dP-ABC (distributed partial attribute-based credential) είναι ένα κρυπτογραφικό πρωτόκολλο που επιτρέπει τη διανομή των διαπιστευτηρίων των χρηστών ανάμεσα σε πολλαπλούς παρόχους ταυτότητας (IdPs), διατηρώντας ταυτόχρονα το απόρρητο και την ασφάλεια. Κάθε IdP δημιουργεί το δικό του ζεύγος δημόσιου-ιδιωτικού κλειδιού και υπογράφει τα χαρακτηριστικά του χρήστη με το ιδιωτικό του κλειδί, παράγοντας ένα μερικό διαπιστευτήριο που περιέχει ένα υποσύνολο των χαρακτηριστικών του χρήστη. Στη συνέχεια, ο χρήστης συνδυάζει τα μερικά διαπιστευτήρια από όλους τους IdPs, δημιουργώντας ένα τελικό επαληθεύσιμο διαπιστευτήριο, χρησιμοποιώντας το δημόσιο κλειδί του vIdP.

Αυτό το σύστημα βασίζεται στην έννοια των διαπιστευτηρίων βάσει χαρακτηριστικών, όπου η πρόσβαση σε πόρους χορηγείται με βάση την εκπλήρωση συγκεκριμένων πολιτικών που βασίζονται σε χαρακτηριστικά.



Εικόνα 3 : Αρχιτεκτονική Συστήματος

Στο πλαίσιο της υλοποίησης της εφαρμογής σχεδιάστηκαν και υλοποιήθηκαν οι παρακάτω μέθοδοι ώστε η εφαρμογή να μπορεί να εκδίδει πιστοποιητικά για. Αυτά τα πιστοποιητικά είναι σχεδιασμένα ώστε να μπορούν να επαληθευτούν από την εφαρμογή που χρησιμοποιείται.

1) SignIdentityProof

Για την υλοποίηση της εφαρμογής δημιουργήσαμε την κλάση "SignIdentityProof", η οποία επεκτείνει μια υπάρχουσα κλάση με το ίδιο όνομα για την εφαρμογή υπογραφής βάσει χαρακτηριστικών. Η κλάση αυτή περιλαμβάνει δύο βασικές μεταβλητές: την υπογραφή (signature), που είναι μια συμβολοσειρά αναπαράστασης της ψηφιακής υπογραφής για την εξασφάλιση της εγκυρότητας των δεδομένων, και το data, ένα αντικείμενο της κλάσης UniAttributes, που περιέχει χαρακτηριστικά σχετικά με τα πανεπιστημιακά στοιχεία, όπως το όνομα του τίτλου και το ίδρυμα. Αυτή η δομή υποστηρίζει την ασφαλή και αξιόπιστη διαχείριση των πιστοποιητικών.

Ο εικονικός πάροχος ταυτότητας (vIdP) σχηματίζεται με την άθροιση των δημόσιων κλειδιών όλων των μερικών παρόχων ταυτότητας (IdPs), χρησιμοποιώντας ένα ασφαλές πρωτόκολλο υπολογισμού πολλαπλών μερών, το οποίο διασφαλίζει ότι το ιδιωτικό κλειδί του vIdP δεν αποκαλύπτεται σε κανέναν.

Αυτό επιτρέπει στον vIdP να επαληθεύει την αυθεντικότητα των διαπιστευτηρίων που παρέχονται από τον χρήστη, ο οποίος μπορεί να δημιουργεί επαληθεύσιμα διαπιστευτήρια που ενδέχεται να περιέχουν ευαίσθητα προσωπικά δεδομένα. Αντί να μοιράζεται απευθείας αυτά τα δεδομένα, ο χρήστης μπορεί να δημιουργήσει

κρυπτογραφημένα ή προστατευμένα τμήματα και να τα διανείμει στον vIdP ή σε άλλες οντότητες που συμμετέχουν στη διαδικασία δημιουργίας διαπιστευτηρίων.

Ο vIdP στη συνέχεια χρησιμοποιεί ένα συγκεντρωτικό δημόσιο κλειδί για να συνδυάσει τα δεδομένα αυτά και να δημιουργήσει ένα τελικό επαληθεύσιμο διαπιστευτήριο, το οποίο ο χρήστης μπορεί να παρουσιάσει για να αποδείξει την ταυτότητά του ή άλλα χαρακτηριστικά.

Αυτή η προσέγγιση επιτρέπει μια κατανεμημένη και ελαχιστοποιημένη σε επίπεδο εμπιστοσύνης διαδικασία, όπου καμία μεμονωμένη οντότητα δεν έχει πλήρη πρόσβαση στα προσωπικά δεδομένα του χρήστη.

Η κλάση "SignIdentityProof" περιέχει τρεις κατασκευαστές που διευκολύνουν τη δημιουργία νέων περιπτώσεων της κλάσης. Ο πρώτος είναι ένας προεπιλεγμένος κατασκευαστής χωρίς ορίσματα, ο οποίος δημιουργεί μια κενή περίπτωση της κλάσης χωρίς αρχικές τιμές για τις μεταβλητές signature και data.

Ο δεύτερος κατασκευαστής δέχεται ένα JSONObject ως παράμετρο, ανακτά την υπογραφή από το αντικείμενο JSON και προσπαθεί να δημιουργήσει ένα αντικείμενο UniAttributes με τα δεδομένα του, επιτρέποντας τη δημιουργία μιας περίπτωσης με πληροφορίες υπογραφής και δεδομένων. Ο τρίτος κατασκευαστής δέχεται απευθείας μια συμβολοσειρά υπογραφής και ένα αντικείμενο Attributes, εκχωρώντας τα στις αντίστοιχες μεταβλητές, και είναι χρήσιμος όταν υπάρχουν ήδη τα δεδομένα υπογραφής και χαρακτηριστικών. Όλοι οι κατασκευαστές συμβάλλουν στη δημιουργία διαπιστευτηρίων (tokens) με διαφορετικές προσεγγίσεις.

2) Attributes

Η κλάση "Attributes" υλοποιήθηκε για να ενσωματώνει μια απόδειξη ταυτότητας που υπογράφεται από την κλάση "SignIdentityProof" και περιλαμβάνει χαρακτηριστικά όπως το όνομα, η ημερομηνία γέννησης, το ηλεκτρονικο ταχυδρομιο , userid, ρόλος. Λειτουργεί ως αποδεικτικό ταυτότητας και μπορεί να χρησιμοποιηθεί για την επαλήθευση της ταυτότητας του ατόμου.

Περιλαμβάνει δύο κατασκευαστές: έναν προεπιλεγμένο χωρίς ορίσματα που δημιουργεί μια κενή περίπτωση της κλάσης με προεπιλεγμένες τιμές, και έναν παραμετροποιημένο που δέχεται τα χαρακτηριστικά name, dateOfBirth, email, userid και role για την αρχικοποίηση των αντίστοιχων πεδίων.

Η κλάση παρέχει επίσης τη μέθοδο "toString" για την αναπαράσταση των τρεχουσών τιμών των πεδίων σε μορφοποιημένη συμβολοσειρά, και τη μέθοδο "toJson" για τη μετατροπή του αντικειμένου σε μορφή JSON, διευκολύνοντας την παρουσίαση των δεδομένων ταυτότητας με κρυπτογραφικά διασφαλισμένο τρόπο.

3) SignIdentityProver

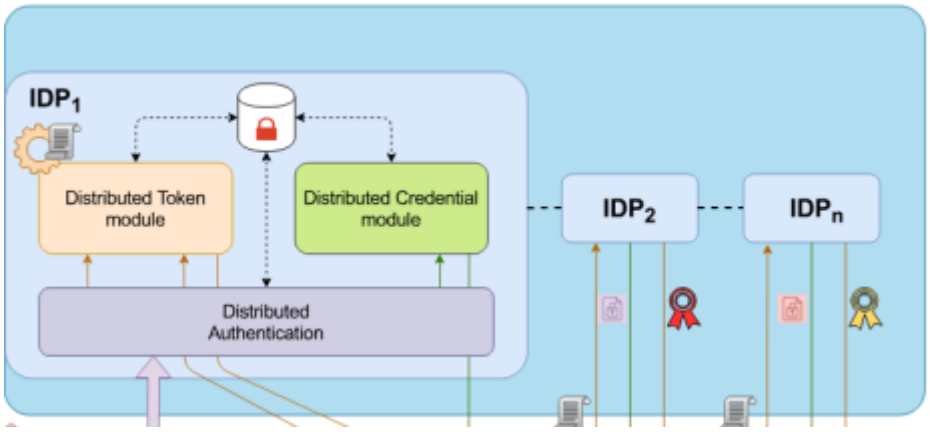
Η κλάση "SignIdentityProof" υλοποιεί τη διεπαφή IdentityProver και αναπαριστά ένα αντικείμενο απόδειξης ταυτότητας που περιέχει υπογραφή και σχετιζόμενα χαρακτηριστικά (Attributes). Είναι υπεύθυνη για την επαλήθευση της εγκυρότητας της απόδειξης ταυτότητας και την προσθήκη των σχετικών χαρακτηριστικών στο προφίλ του χρήστη, το οποίο αποθηκεύεται στο αντικείμενο Storage.

Ο κατασκευαστής της κλάσης δέχεται ένα αντικείμενο Storage, που χρησιμοποιείται για την αποθήκευση και ανάκτηση των χαρακτηριστικών του χρήστη. Η μέθοδος isValid επαληθεύει την εγκυρότητα του idProof και επιστρέφει true αν η υπογραφή είναι έγκυρη, ενώ τα χαρακτηριστικά αποθηκεύονται στο Storage για μελλοντική χρήση. Η μέθοδος addAttributes αναλύει το proof σε ένα αντικείμενο SignIdentityProof, εξάγει τα χαρακτηριστικά και τα αποθηκεύει σε έναν χάρτη (Map), με το όνομα του χαρακτηριστικού ως κλειδί. Η μέθοδος getProof χρησιμοποιεί τον ObjectMapper για να μετατρέψει τη συμβολοσειρά JSON σε αντικείμενο SignIdentityProof, επιστρέφοντας το ανάλυτο αντικείμενο εάν είναι επιτυχής, διαφορετικά επιστρέφει null.

4) Vidp(Εικονικός Πάροχος Ταυτότητας)

Στο σύστημα μας τροποποιήσαμε τους εικονικούς παρόχους ταυτότητας (vIdP) που χρησιμοποιεί ο διακομιστής, ο οποίος βασίζεται στο πρωτόκολλο P-ABC και επικοινωνεί με άλλους διακομιστές μέσω REST API. Κάθε vIdP περιλαμβάνει τρεις βασικές ενότητες: μία για τον έλεγχο ταυτότητας, η οποία επαληθεύει το όνομα χρήστη και τον κωδικό πρόσβασης, και δύο για την κατανεμημένη πιστοποίηση και έκδοση διαπιστευτηρίων. Η κατανεμημένη πιστοποίηση συνεργάζεται με άλλες μονάδες ελέγχου ταυτότητας, και αφού επιβεβαιωθεί η εγκυρότητα, ο χρήστης πιστοποιείται.

Επιπλέον, το σύστημα χρησιμοποιεί ένα σχήμα Pesto για τη δημιουργία παρόχου ταυτότητας (IdP), ο οποίος είναι υπεύθυνος για την έκδοση και επαλήθευση διαπιστευτηρίων, αποθηκεύοντας τα δεδομένα των χρηστών σε μια βάση δεδομένων στη μνήμη. Η κύρια μέθοδος διαβάζει το αρχείο ρυθμίσεων και δημιουργεί ένα αντικείμενο PABCIdPImpl, που χειρίζεται τις αιτήσεις P-ABC, το οποίο αρχικοποιείται με στοιχεία όπως IdentityProver, PestoDatabase, και ServerCryptoModule. Το PABCIdPImpl προωθείται σε ένα αντικείμενο RESTIdPServer, το οποίο εκκινεί τον διακομιστή REST API και χειρίζεται τις αιτήσεις, ενώ το αρχείο ρυθμίσεων περιλαμβάνει πληροφορίες για τη ρύθμιση των απαραίτητων παραμέτρων.



Εικόνα 4 : Σχηματικό vIDP

5.3 Αναλυτική περιγραφή Υλοποίησης

5.3.1 Επεξεργασία Βιβλιοθήκης

Για την υλοποίηση του έργου, ξεκινάμε λαμβάνοντας τον κώδικα από το αποθετήριο ανοιχτού κώδικα (https://bitbucket.alexandra.dk/projects/OL). Ο κώδικας αυτός περιλαμβάνει τον server της εφαρμογής που θα υλοποιήσουμε, καθώς και τον κώδικα για τον service-provider (https://bitbucket.alexandra.dk/projects/OL/repos/olympus-service-provider/browse).

Ακολουθούν τα βήματα:

- Κλωνοποίηση του αποθετηρίου: Ανοίγουμε το IntelliJ IDEA και κλωνοποιούμε το αποθετήριο. Εντοπίζουμε τον φάκελο olympus-identity-master, ο οποίος είναι ο βασικός φάκελος (parent) του project και περιέχει όλα τα modules που θα επεξεργαστούμε.
- Καθαρισμός του project: Στο τερματικό του IntelliJ εκτελούμε την εντολή mvn clean για να καθαρίσουμε το project, διαγράφοντας τον φάκελο target, όπου αποθηκεύονται τα παραγόμενα αρχεία.
- Κατασκευή και εγκατάσταση του project: Στη συνέχεια, εκτελούμε την εντολή mvn install σε όλο το tree του project, ώστε να κατασκευάσουμε και να εγκαταστήσουμε τα αρχεία στο τοπικό αποθετήριο της Maven. Πριν εκτελέσουμε αυτή την εντολή, βεβαιωνόμαστε ότι έχουμε εκκινήσει το Docker για να αποφύγουμε σφάλματα.
- Αλλαγή module: Αφού ολοκληρωθεί το install, ανοίγουμε ένα δεύτερο τερματικό και αλλάζουμε στο module oidc-demo-idp με την εντολή cd oidc-demo-idp.

- Εκκίνηση υπηρεσιών: Εκτελούμε την εντολή docker compose up για να δημιουργήσουμε και να εκκινήσουμε τις υπηρεσίες της εφαρμογής.
- Εκκίνηση front-end: Σε ένα τρίτο τερματικό, αλλάζουμε module με την εντολή cd front και εκτελούμε την εντολή mvn spring-boot:run για να εκκινήσουμε το front-end μέρος της εφαρμογής.
- Κλωνοποίηση του service-provider: Κατεβάζουμε τον φάκελο olympus-service-provider από το αποθετήριο και τον κλωνοποιούμε στο τοπικό μας directory.
- Εγκατάσταση εξαρτήσεων: Σε ένα τέταρτο τερματικό, πηγαίνουμε στον φάκελο olympus-service-provider με την εντολή cd olympus-service-provider και εκτελούμε npm install για να εγκαταστήσουμε όλες τις εξαρτήσεις από το αρχείο package.json.
- Εκκίνηση της εφαρμογής: Εκτελούμε την εντολή npm start και ελέγχουμε στον browser αν η εφαρμογή λειτουργεί σωστά στον localhost.

5.4 Τροποποίηση Γραφικού Περιβάλλοντος

Για την υλοποίηση του έργου, προχωράμε στις απαραίτητες προσαρμογές ώστε η εφαρμογή να πληροί τις νέες απαιτήσεις και να ενσωματώσει τις λειτουργίες που έχουμε ορίσει. Στόχος είναι η εφαρμογή να λειτουργεί ομαλά, με έμφαση στην αισθητική και τη λειτουργικότητα, καθώς και στην ενίσχυση της ασφάλειας των χρηστών. Αρχικά, επικεντρωνόμαστε στις αλλαγές του front-end, καθώς είναι ταχύτερες στην υλοποίηση και θα μας επιτρέψουν να δώσουμε άμεσα μορφή στην εφαρμογή.

Τροποποιήσεις με βάση το Σενάριο Χρήσης:

1. **Εκκίνηση Συστήματος και Σελίδα Σύνδεσης:**
    - Σχεδιάστηκε η αρχική σελίδα όπου ο χρήστης βλέπει την επιλογή "LOG IN".
    - Προστέθηκε λειτουργικότητα για την ανακατεύθυνση στη σελίδα "Verification System".
2. **Διαδικασία Εισαγωγής Νέου Χρήστη:**
    - Υλοποιήθηκε η σελίδα όπου ο διαχειριστής μπορεί να εισάγει τα στοιχεία των νέων χρηστών.
    - Προστέθηκαν φόρμες εισαγωγής για το όνομα, το επώνυμο, την ειδικότητα και άλλες λεπτομέρειες του κλινικού γιατρού.
3. **Προσαρμογές στη Σελίδα Credential Storage:**

- Προστέθηκαν επιλογές διαχείρισης credentials, όπως "Αποκάλυψη" ή "Απόκρυψη".
- Ενσωματώθηκε λειτουργικότητα για την προβολή επιλογών όπως "Home", "Personal Information", "Change Password", "Delete Account" και "Logout".

4. **Διαχείριση Pop-up Μηνυμάτων:**
    - Προστέθηκαν δυναμικά μηνύματα ενημέρωσης για την επιτυχή ολοκλήρωση ενεργειών, όπως καταχώρηση, επαλήθευση ή αποθήκευση credentials.
5. **Ενίσχυση της Ασφάλειας:**
    - Βελτιώθηκε η σελίδα "Check Access Credential" με ασφαλή επιβεβαίωση των credentials μέσω κουμπιού "Login Via Credential".
6. **Προσαρμογή User Flow:**
    - Διασφαλίστηκε η ομαλή μετάβαση ανάμεσα στις διάφορες σελίδες, με στόχο την καλύτερη εμπειρία χρήστη (UX).
    - Προστέθηκαν οδηγίες πλοήγησης στις σελίδες με ξεκάθαρη δομή, σύμφωνα με το σενάριο χρήσης.
7. **Τελική Σελίδα Επαλήθευσης:**
    - Σχεδιάστηκε μήνυμα επιβεβαίωσης που ενημερώνει τον χρήστη ότι είναι πλέον "Verified".

Με αυτές τις τροποποιήσεις, η εφαρμογή ανταποκρίνεται στις ανάγκες που περιγράφονται στο σενάριο χρήσης και παρέχει μια βελτιωμένη εμπειρία χρήστη, εστιάζοντας στην ευχρηστία και την ασφάλεια.

5.4.1. Αλλαγές στην απεικόνιση του γραφικού περιβάλλοντος

Οι πρώτες αλλαγές που υλοποιούμε αφορούν την αισθητική του front-end και την προσαρμογή του στις απαιτήσεις του έργου. Συγκεκριμένα:

- Αντικατάσταση του λογότυπου: Στο φάκελο src/main/resources/templates/login.html του service-provider, αντικαθιστούμε το λογότυπο της εφαρμογής με ένα νέο που αντιπροσωπεύει την ταυτότητα του οργανισμού.
- Προσαρμογή μηνυμάτων: Αλλάζουμε το μήνυμα που εμφανίζεται κατά την είσοδο στην εφαρμογή από "Login to university account" σε "Login to your clinician account", ώστε να αντικατοπτρίζει την κύρια χρήση της εφαρμογής, η

οποία αφορά κλινικούς επαγγελματίες. Επίσης, μετονομάζουμε το "University Secretariat" σε "Secretariat" για να ανταποκρίνεται στον οργανισμό που χρησιμοποιεί την εφαρμογή.

- Αναδυόμενο μήνυμα κατά την εγγραφή: Όταν ο χρήστης ολοκληρώνει την εγγραφή του, προσθέτουμε ένα popup μήνυμα που αναφέρει "You have successfully registered". Η αλλαγή αυτή γίνεται στο αρχείο src/components/mainpage.js του service-provider, και για να την υποστηρίξουμε, εγκαθιστούμε τη βιβλιοθήκη SweetAlert2 με την εντολή npm install sweetalert2.
- Ενημέρωση του τίτλου και του υπότιτλου: Στο αρχείο FrontPage.js που βρίσκεται στη διαδρομή service-provider/src/components, αλλάζουμε τον τίτλο και τον υπότιτλο της σελίδας ώστε να αντικατοπτρίζουν το όνομα και τη λειτουργία της εφαρμογής μας: "Verification System". Επίσης, τροποποιούμε τη διαδρομή του κουμπιού "Login" ώστε να κατευθύνει τον χρήστη στη σωστή σελίδα σύνδεσης της νέας εφαρμογής.
- Προσαρμογές στα CSS και HTML αρχεία: Τα αρχεία Login.css και Login.html προσαρμόζονται για να ανταποκρίνονται στις νέες αλλαγές και να βελτιώσουν την εμπειρία χρήστη. Αυτό περιλαμβάνει ενημέρωση του στυλ και της δομής των HTML στοιχείων.

5.4.2 Παραμετροποίηση των χαρακτηριστικών των διαπιστευτηρίων

Η παραμετροποίηση των πεδίων διαπιστευτηρίων αποτελεί κρίσιμο βήμα για τη λειτουργία της εφαρμογής. Στο αρχείο **Storage.js**, πραγματοποιούνται αλλαγές για την εισαγωγή νέων χαρακτηριστικών που σχετίζονται με τα δεδομένα χρηστών, όπως το **First Name**, το **Last Name**, το **Clinician ID**, το **Email**, το **Contact Number** και το **Children Full Name**. Τα πεδία αυτά είναι αναγκαία για την ταυτοποίηση και την ασφαλή πρόσβαση των χρηστών στο σύστημα.

Για την υλοποίηση αυτών των τροποποιήσεων, έχουν ενσωματωθεί οι κατάλληλες μέθοδοι διαχείρισης δεδομένων, οι οποίες περιγράφονται αναλυτικά στο Παράρτημα **7.1**. Οι μέθοδοι αυτές περιλαμβάνουν την εισαγωγή νέων διαπιστευτηρίων μέσω της `insertUserDetails()`, καθώς και τη διαχείριση άλλων χαρακτηριστικών μέσω των `AddAttributes` και `DeleteAttributes`. Επιπλέον, στο αρχείο **LoginPage.js**, προστέθηκε λειτουργικότητα για τη σύνδεση χρηστών μέσω διαπιστευτηρίων, όπως αναφέρεται στο Παράρτημα.

5.4.3 Παραμετροποίηση της ιδιότητας που πιστοποιεί το διαπιστευτηριο

- Η δυνατότητα διαγραφής διαπιστευτηρίων παρέχει ευελιξία και ασφάλεια στη διαχείριση των δεδομένων χρηστών. Ένα νέο κουμπί, "Delete this credential", έχει προστεθεί στην εφαρμογή, επιτρέποντας τη διαγραφή συγκεκριμένων credentials. Όταν ο χρήστης επιλέγει αυτή τη λειτουργία, εκτελείται η μέθοδος `deleteCredential()` από το αρχείο **Storage.js**, η οποία αφαιρεί τα δεδομένα από τη βάση και εμφανίζει ένα μήνυμα επιβεβαίωσης.

- Οι λεπτομέρειες της υλοποίησης αυτής της λειτουργίας, καθώς και η διαχείριση των μηνυμάτων επιβεβαίωσης, περιγράφονται στο Παράρτημα **7.1**.

5.4.4 Αλλαγές στη σελίδα σύνδεσης (Login)

Η σελίδα σύνδεσης προσαρμόζεται ώστε να ενισχύσει τη διαδικασία ταυτοποίησης και επαλήθευσης:

- Προσθήκη κουμπιού "Log in with Credentials": Στο αρχείο LoginPage.js προσθέτουμε ένα κουμπί που επιτρέπει στους χρήστες να συνδεθούν χρησιμοποιώντας τα credentials τους. Όταν επιλέγεται το κουμπί, ο χρήστης ανακατευθύνεται στη σελίδα checkCredentials.
- Αναδυόμενο μήνυμα επιτυχούς σύνδεσης: Όταν ο χρήστης συνδέεται επιτυχώς, εμφανίζεται ένα αναδυόμενο μήνυμα "You are verified", το οποίο απαιτεί από τον χρήστη να επιλέξει "OK" για να συνεχίσει.

5.4.5 Διασφάλιση λειτουργικότητας μετά τις αλλαγές

Μετά από κάθε αλλαγή στον κώδικα του service-provider, εκτελούμε την εντολή npm run-script build για να μεταγλωττίσουμε τις αλλαγές μας. Στη συνέχεια, εκκινούμε την εφαρμογή στον τοπικό server με την εντολή npm start και ελέγχουμε αν οι αλλαγές έχουν εφαρμοστεί σωστά.

Με αυτές τις βελτιώσεις, η εφαρμογή όχι μόνο αποκτά αισθητική που ταιριάζει με τις ανάγκες του οργανισμού, αλλά βελτιώνεται και λειτουργικά, εξασφαλίζοντας ασφαλή και αποδοτική διαχείριση των δεδομένων των χρηστών.

5.4.6 Αναλυτική Περιγραφή Αλλαγών

1) Οι αλλαγές στο αρχείο **Login.css** επικεντρώνονται στη βελτίωση της αισθητικής και της ευχρηστίας της σελίδας σύνδεσης.

- **body:** Δημιουργούμε ένα γραμμικό gradient φόντο για τη σελίδα σύνδεσης με την παρακάτω συνάρτηση:

```
body {

  background: linear-gradient(304deg, rgba(255, 255, 255, 1) 0%, rgb(51, 167, 196) 35%, rgb(116, 131, 132) 100%);

}
```

Αυτή η αλλαγή προσφέρει ομαλές χρωματικές διαβαθμίσεις που βελτιώνουν την οπτική εμπειρία του χρήστη.

- **container**: Προσθέτουμε flexbox για να κεντράρουμε το περιεχόμενο και να ρυθμίσουμε τη διάταξή του οριζόντια:

```
.container {

  display: flex;

  flex-direction: row;

  align-items: center;

  justify-content: center;

}
```

- **form-control**: Εφαρμόζουμε σκιές και περιγράμματα στα πεδία εισαγωγής δεδομένων για να φαίνονται πιο καθαρά:

```
.form-control {

  box-shadow: 0px 0px 10px rgba(66, 66, 66, .75);

  border-radius: 5px;

}
```

- **button**: Δημιουργούμε ένα κουμπί με κυκλικές γωνίες και καθορισμένο χρώμα για hover και active εφέ:

```
button {
  background-color: #2dabf9;
  border-radius: 17px;
  transition: background-color 0.3s ease;
}
button:hover {
  background-color: #1a8cd8;
}
```

2) Οι αλλαγές στο **Login.html** έχουν σκοπό την προσαρμογή της φόρμας και της δομής της σελίδας στις νέες απαιτήσεις.

- Thymeleaf expressions: Εισάγουμε δυναμικό περιεχόμενο χρησιμοποιώντας Thymeleaf για να ελέγξουμε συνθήκες και ενέργειες:

```
<form th:action="@{/authenticate}" method="post"
th:object="${loginRequest}">
  <input type="text" id="username" name="username" class="form-control"
th:field="*{username}" placeholder="Enter Username"/>
</form>
```

- **HTML δομή:** Δημιουργούμε τη βασική δομή της σελίδας με containers για τα στοιχεία της φόρμας:

```
<div class="inner_container">
  <form th:action="@{/authenticate}" method="post"
th:object="${loginRequest}">
    <input type="text" id="username" name="username" class="form-control"
th:field="*{username}" placeholder="Enter Username"/>
    <input type="password" id="password" name="password" class="form-
control" th:field="*{password}" placeholder="Enter Password"/>
    <button type="submit">Login</button>
  </form>
```

** **

- CSS σύνδεση: Συνδέουμε το εξωτερικό αρχείο CSS για να εφαρμόσουμε τα στυλ:

<link th:href="@{/css/login1.css}" rel="stylesheet"/>

3) Storage.js

Οι αλλαγές στο Storage.js περιλαμβάνουν τη διαχείριση νέων πληροφοριών χρήστη, όπως το Specialty και το Clinician ID.

- Παράδειγμα αλλαγμένου κώδικα για την εμφάνιση των στοιχείων του χρήστη:

<h6 style={{color: "black"}}>Specialty: {user ? user.profile.given_name : "-"}</h6>

<h6 style={{color: "black"}}>Clinician ID: {user ? user.profile.nickname : "-"}</h6>

<h6 style={{color: "black"}}>Group ADHD: {user ? user.profile.middle_name : "-"}</h6>

4) FrontPage.js

Στο FrontPage.js, αλλάζουμε τον τίτλο και το κουμπί σύνδεσης για να αντικατοπτρίζει το νέο όνομα της εφαρμογής.

- **Τίτλος και υπότιτλος**:

<Typography variant="h2" align="center"> Verification System </Typography>

- **Διαδρομή στο κουμπί σύνδεσης:**

<Button

size="medium"

variant="outlined"

onClick={(event) => {

```
    event.preventDefault();

    this.props.dispatch(push("/login"));

  }}

>

  Login to Aurora Minds

</Button>
```

5) **Mainpage.js**

Στο mainpage.js, προσθέτουμε την εμφάνιση ενός αναδυόμενου μηνύματος (pop-up) μετά την επιτυχημένη εγγραφή:

```
var hasRegistered = localStorage.getItem('hasRegistered');

if (!hasRegistered) {

  swal.fire("You successfully registered");

  localStorage.setItem('hasRegistered', 'true');

}
```

6) **Watch-verifier.js**

Αλλάζουμε τη συνθήκη ελέγχου για τον Clinician ID και την ανακατεύθυνση του χρήστη με βάση την επαλήθευση των credentials.

```
const isClinicianID = details["Clinician Id"] === "8";
```

- **Συνθήκη για το κουμπί "Continue to application form"**:

```
const canUserContinue = () => {

  const atLeastOneFieldRevealed = Object.values(revealDetails).some((value) =>
value === true);

  const isClinicianID = details["Clinician Id"] === "8";

  return atLeastOneFieldRevealed && isClinicianID;

};
```

7) **LoginPage.js**

Προσθέτουμε κουμπί για σύνδεση με credentials και λειτουργία ανακατεύθυνσης.

- **Κουμπί "Log in with Credentials"**:

```
<Button
  size="large"
  variant="outlined"
  startIcon={<DirectionsIcon />}
  onClick={this.onLoginCredentialsClick}
  style={styles.button}
>
  Log in with Credentials
</Button>
```

Ανακατεύθυνση στη σελίδα checkCredentials:

```
onLoginCredentialsClick(event) {
  event.preventDefault();
  window.location.href = "http://localhost:8080/loginCredentials";
}
```

5.5 Παραμετροποίηση Credentials

Το σύστημα που υλοποιήθηκε βασίζεται στην τεχνολογία blockchain για να διασφαλίσει την ασφάλεια και την ιδιωτικότητα των χρηστών, ιδιαίτερα των μικρών παιδιών και των κλινικών επαγγελματιών. Μέσω της χρήσης διαπιστευτηρίων (credentials) και ταυτοποίησης μέσω identity proofs, επιτυγχάνεται η ασφαλής διαχείριση των προσωπικών δεδομένων.

1) Προσθήκη Νέων Credentials για Διάφορες Κατηγορίες Χρηστών

Για να προσαρμόσουμε την εφαρμογή στις ανάγκες διαφορετικών κατηγοριών χρηστών, προσθέτουμε νέα credentials για γονείς, παιδιά και κλινικούς επαγγελματίες:

- **Για τους γονείς**, τα credentials περιλαμβάνουν: οικογενειακό όνομα, όνομα, αριθμό ταυτότητας, όνομα παιδιού, συγκατάθεση συμμετοχής, ημερομηνία γέννησης, χώρα, πόλη και διεύθυνση.
- **Για τα παιδιά**, τα credentials περιλαμβάνουν: οικογενειακό όνομα, όνομα, αριθμό ταυτότητας, ημερομηνία γέννησης, διάγνωση, ομάδα χρήστη, συγκατάθεση γονέων, χώρα, πόλη και διεύθυνση.
- **Για τους κλινικούς επαγγελματίες**, τα credentials περιλαμβάνουν: οικογενειακό όνομα, όνομα, αριθμό ταυτότητας, ειδικότητα, ομάδα χρήστη, ημερομηνία γέννησης, χώρα, πόλη και διεύθυνση.

Με την εισαγωγή αυτών των credentials, η εφαρμογή μπορεί να διαχειριστεί καλύτερα τις πληροφορίες κάθε ομάδας χρηστών, προσφέροντας μια εξατομικευμένη εμπειρία.

2) Αλλαγές στα Αρχεία Storage.js και ClinicianUserRequest.java

Για να ανταποκριθούμε στα νέα δεδομένα, ενημερώνουμε τις μεταβλητές που περιγράφουν τα credentials του χρήστη στα αρχεία:

- **Storage.js**: Οι μεταβλητές ενημερώνονται ώστε να αντικατοπτρίζουν τα νέα πεδία, όπως το όνομα, η ημερομηνία γέννησης, η ειδικότητα, το αναγνωριστικό του κλινικού επαγγελματία και η ομάδα χρήστη.
- **ClinicianUserRequest.java**: Γίνονται αντίστοιχες αλλαγές για τη σωστή εισαγωγή και διαχείριση των δεδομένων στο back-end.

Με αυτές τις αλλαγές, διασφαλίζεται ότι η εφαρμογή χειρίζεται τα νέα credentials σωστά, διατηρώντας τη συνέπεια και την ακρίβεια στα δεδομένα των χρηστών.

3) **Δημιουργία Νέων Μοντέλων**

Η επόμενη φάση περιλαμβάνει τη δημιουργία τριών νέων μοντέλων που αντικαθιστούν το αρχικό UserClient μοντέλο:

- **CreateParent**: Περιλαμβάνει πεδία για την αποθήκευση των δεδομένων του γονέα και πληροφορίες που αφορούν τα παιδιά και τον κλινικό γιατρό με τον οποίο συνδέεται.
- **CreateChildren**: Επικεντρώνεται στα δεδομένα των παιδιών, συμπεριλαμβανομένων στοιχείων όπως η διάγνωση και η ομάδα χρήστη.
- **CreateClinician**: Περιέχει δεδομένα που αφορούν τους κλινικούς επαγγελματίες, όπως ειδικότητες και αναγνωριστικά.

Αυτά τα νέα μοντέλα επιτρέπουν τη διαχείριση των δεδομένων κάθε κατηγορίας χρηστών ξεχωριστά, βελτιώνοντας την ευελιξία και την αποτελεσματικότητα της εφαρμογής.

4) **Προσθήκη Identity Proof και Ενημέρωση Controllers**

Η τελευταία φάση περιλαμβάνει αλλαγές στον controller της εφαρμογής:

- Στην κλάση **oidcController** που βρίσκεται στο μονοπάτι front/src/main/java/oidc/controller/oidcController, προσθέτουμε ένα νέο attribute με την ονομασία identity proof. Αυτό το attribute παρέχει ένα επιπλέον επίπεδο ασφάλειας για τη διαχείριση και επαλήθευση της ταυτότητας των χρηστών.
- Η προσθήκη του identity proof επιτρέπει τη διασφάλιση της ταυτότητας των χρηστών κατά την αλληλεπίδρασή τους με την εφαρμογή, ενισχύοντας την ασφάλεια του συστήματος.

Με τις παραπάνω αλλαγές, η εφαρμογή γίνεται πιο ασφαλής και ευέλικτη, ικανοποιώντας τις ανάγκες διαφορετικών ομάδων χρηστών και βελτιώνοντας την εμπειρία τους.

Επιπλέον τα αρχεία που επεξεργάστηκαν και τα αντίστοιχα μονοπάτια τους:

1. **Storage.js**
    - **Μονοπάτι**: service-provider/src/components
    - Αυτό το αρχείο χειρίζεται τα credentials του χρήστη και ενημερώθηκε για να υποστηρίξει τα νέα πεδία για γονείς, παιδιά και κλινικούς επαγγελματίες.

2. **clinicianUserRequest.java**
    - **Μονοπάτι**: identity-master/front/src/main/java/oidc/model
    - Στο αρχείο αυτό, πραγματοποιήθηκαν αλλαγές για να προστεθούν τα νέα πεδία που αφορούν τους κλινικούς επαγγελματίες και να διασφαλιστεί η σωστή διαχείριση των δεδομένων τους.
3. **oidcController.java**
    - **Μονοπάτι**: front/src/main/java/oidc/controller
    - Σε αυτόν τον ελεγκτή προστέθηκε το identity proof, το οποίο ενισχύει την ασφάλεια της ταυτοποίησης και την επαλήθευση των χρηστών κατά την αλληλεπίδρασή τους με την εφαρμογή.

Αυτά τα αρχεία αντιπροσωπεύουν τις κύριες περιοχές όπου πραγματοποιήθηκαν τροποποιήσεις, ώστε να εξασφαλιστεί η ορθή διαχείριση των νέων credentials και η ασφάλεια των δεδομένων στο σύστημα.

5.6 Τροποποίηση input users

Για κάθε ένα από τα νέα πεδία που προσθέσαμε στα μοντέλα, υλοποιούμε τις κατάλληλες μεθόδους getter και setter. Αυτές οι μέθοδοι επιτρέπουν την ασφαλή πρόσβαση και τροποποίηση των δεδομένων που σχετίζονται με τα διάφορα credentials των χρηστών, όπως οι πληροφορίες για τους γονείς, τα παιδιά και τους κλινικούς επαγγελματίες. Με αυτόν τον τρόπο, διασφαλίζουμε την οργανωμένη και ασφαλή διαχείριση των δεδομένων εντός του συστήματος, κάτι που είναι κρίσιμο για τη σωστή λειτουργία της εφαρμογής και τη διασφάλιση της ιδιωτικότητας των χρηστών.

Ένα από τα νέα μοντέλα που δημιουργήσαμε είναι το CreateChildren. Αυτό το μοντέλο περιλαμβάνει τα απαραίτητα πεδία για την αποθήκευση των στοιχείων των παιδιών. Τα πεδία αυτά επιτρέπουν την καταγραφή σημαντικών πληροφοριών, όπως η ημερομηνία γέννησης, η διάγνωση και η συγκατάθεση των γονέων. Για τη διαχείριση αυτών των δεδομένων, έχουμε υλοποιήσει τις αντίστοιχες μεθόδους getter και setter, οι οποίες εξασφαλίζουν ότι η πρόσβαση και η τροποποίηση των δεδομένων των παιδιών πραγματοποιούνται με ασφαλή και οργανωμένο τρόπο.

Παράλληλα, δημιουργήσαμε το μοντέλο CreateClinician, το οποίο περιέχει τα απαραίτητα πεδία για την αποθήκευση των στοιχείων ενός κλινικού γιατρού. Αυτά τα πεδία περιλαμβάνουν βασικές πληροφορίες όπως το όνομα, την ειδικότητα και τον αριθμό ταυτότητας του κλινικού γιατρού. Όπως και στα προηγούμενα μοντέλα, υλοποιήσαμε τις μεθόδους getter και setter για κάθε πεδίο, ώστε να εξασφαλίσουμε τη σωστή διαχείριση των πληροφοριών των κλινικών επαγγελματιών και τη διατήρηση της ακεραιότητας των δεδομένων τους.

Μετά την προσθήκη των νέων μοντέλων, ήταν απαραίτητο να αναθεωρήσουμε τη μέθοδο δημιουργίας χρηστών ώστε να υποστηρίζει τα νέα αυτά μοντέλα. Η δημιουργία των χρηστών τώρα περιλαμβάνει την αποθήκευση και διαχείριση των δεδομένων για γονείς, παιδιά και κλινικούς, διασφαλίζοντας ότι η εφαρμογή μπορεί να εξυπηρετήσει διαφορετικές ομάδες χρηστών με βάση τις εξειδικευμένες ανάγκες τους. Για τη δημιουργία και αποθήκευση αυτών των πληροφοριών χρησιμοποιούμε τη μέθοδο POST, η οποία είναι κατάλληλη για την ασφαλή μεταφορά και αποθήκευση δεδομένων, ειδικά όταν πρόκειται για ευαίσθητα προσωπικά στοιχεία.

Με αυτές τις αλλαγές, διασφαλίζουμε ότι το σύστημα μπορεί να διαχειριστεί λεπτομερώς τις πληροφορίες των χρηστών, υποστηρίζοντας τη διαφοροποίηση μεταξύ γονέων, παιδιών και κλινικών επαγγελματιών, ενώ παράλληλα ενισχύεται η συνολική λειτουργικότητα της εφαρμογής.

Για κάθε ένα από τα νέα πεδία που προσθέσαμε στα μοντέλα, υλοποιούμε τις κατάλληλες μεθόδους getter και setter. Αυτές οι μέθοδοι επιτρέπουν την ασφαλή πρόσβαση και τροποποίηση των δεδομένων που σχετίζονται με τα διάφορα credentials των χρηστών, όπως οι πληροφορίες για τους γονείς, τα παιδιά και τους κλινικούς επαγγελματίες. Με αυτόν τον τρόπο, διασφαλίζουμε την οργανωμένη και ασφαλή διαχείριση των δεδομένων εντός του συστήματος, κάτι που είναι κρίσιμο για τη σωστή λειτουργία της εφαρμογής και τη διασφάλιση της ιδιωτικότητας των χρηστών.

Ένα από τα νέα μοντέλα που δημιουργήσαμε είναι το CreateChildren. Αυτό το μοντέλο περιλαμβάνει τα απαραίτητα πεδία για την αποθήκευση των στοιχείων των παιδιών. Τα πεδία αυτά επιτρέπουν την καταγραφή σημαντικών πληροφοριών, όπως η ημερομηνία γέννησης, η διάγνωση και η συγκατάθεση των γονέων. Για τη διαχείριση αυτών των δεδομένων, έχουμε υλοποιήσει τις αντίστοιχες μεθόδους getter και setter, οι οποίες εξασφαλίζουν ότι η πρόσβαση και η τροποποίηση των δεδομένων των παιδιών πραγματοποιούνται με ασφαλή και οργανωμένο τρόπο.

Παράλληλα, δημιουργήσαμε το μοντέλο CreateClinician, το οποίο περιέχει τα απαραίτητα πεδία για την αποθήκευση των στοιχείων ενός κλινικού γιατρού. Αυτά τα πεδία περιλαμβάνουν βασικές πληροφορίες όπως το όνομα, την ειδικότητα και τον αριθμό ταυτότητας του κλινικού γιατρού. Όπως και στα προηγούμενα μοντέλα, υλοποιήσαμε τις μεθόδους getter και setter για κάθε πεδίο, ώστε να εξασφαλίσουμε τη σωστή διαχείριση των πληροφοριών των κλινικών επαγγελματιών και τη διατήρηση της ακεραιότητας των δεδομένων τους.

Μετά την προσθήκη των νέων μοντέλων, ήταν απαραίτητο να αναθεωρήσουμε τη μέθοδο δημιουργίας χρηστών ώστε να υποστηρίζει τα νέα αυτά μοντέλα. Η δημιουργία των χρηστών τώρα περιλαμβάνει την αποθήκευση και διαχείριση των δεδομένων για γονείς, παιδιά και κλινικούς, διασφαλίζοντας ότι η εφαρμογή μπορεί να εξυπηρετήσει διαφορετικές ομάδες χρηστών με βάση τις εξειδικευμένες ανάγκες τους. Για τη δημιουργία και αποθήκευση αυτών των πληροφοριών χρησιμοποιούμε τη μέθοδο POST, η οποία είναι κατάλληλη για την ασφαλή μεταφορά και αποθήκευση δεδομένων, ειδικά όταν πρόκειται για ευαίσθητα προσωπικά στοιχεία.

Με αυτές τις αλλαγές, διασφαλίζουμε ότι το σύστημα μπορεί να διαχειριστεί λεπτομερώς τις πληροφορίες των χρηστών, υποστηρίζοντας τη διαφοροποίηση μεταξύ γονέων, παιδιών και κλινικών επαγγελματιών, ενώ παράλληλα ενισχύεται η συνολική λειτουργικότητα της εφαρμογής.

6. Συμπεράσματα και Μελλοντική Εργασία

6.1 Συμπεράσματα

Η παρούσα εργασία επικεντρώθηκε στην ανάπτυξη και υλοποίηση μιας διαδικτυακής εφαρμογής χρησιμοποιώντας σύγχρονες τεχνολογίες και πλαίσια. Στόχος της ήταν η βελτίωση της παραγωγικότητας στην ανάπτυξη λογισμικού, αλλά και η συμμόρφωση με τις σύγχρονες απαιτήσεις ασφαλείας, δεδομένων και απόδοσης. Ένα από τα σημαντικότερα συμπεράσματα είναι η σημασία της σωστής επιλογής τεχνολογιών, όπως το Spring Boot, που διευκολύνει τη δημιουργία ισχυρών και ευέλικτων εφαρμογών backend με υψηλή απόδοση και ασφάλεια. Επιπλέον, η χρήση του IntelliJ IDEA για την ανάπτυξη βελτίωσε την αποδοτικότητα των προγραμματιστών, παρέχοντας εργαλεία που απλοποιούν την ανάπτυξη κώδικα και τη διαχείριση εξαρτήσεων.

Το Maven προσέφερε την απαραίτητη αυτοματοποίηση για τη διαχείριση εξαρτήσεων και τη διαδικασία build, μειώνοντας τη χειροκίνητη παρέμβαση και το περιθώριο σφαλμάτων. Όσον αφορά τη διαχείριση προσωπικών δεδομένων, η ενσωμάτωση cookies για την εξατομίκευση της εμπειρίας χρήστη συνοδεύτηκε από πολιτικές συμμόρφωσης με κανονισμούς όπως ο GDPR, διασφαλίζοντας την ασφάλεια και την προστασία των δεδομένων των χρηστών.

Έτσι, η εργασία απέδειξε πως η ανάπτυξη ασφαλών και λειτουργικών διαδικτυακών εφαρμογών απαιτεί όχι μόνο την επιλογή των σωστών εργαλείων, αλλά και τη συνεπή συμμόρφωση με τις κανονιστικές απαιτήσεις. Πέρα από την τεχνολογική διάσταση, η εργασία πέτυχε τη δημιουργία ενός ολοκληρωμένου και αποδοτικού συστήματος που επιλύει συγκεκριμένες ανάγκες, όπως η ασφάλεια και η διαχείριση μεγάλων όγκων δεδομένων. Η εφαρμογή υποστηρίζει τόσο τους τεχνικούς στόχους όσο και τις ανάγκες των τελικών χρηστών, προσφέροντας μια ολιστική προσέγγιση στην ανάπτυξη λογισμικού.

6.2 Προτάσεις για Μελλοντική Βελτίωση

Παρόλο που η εργασία έθεσε ισχυρά θεμέλια για την ανάπτυξη ασφαλών και αποδοτικών διαδικτυακών εφαρμογών, υπάρχουν περιθώρια για περαιτέρω βελτιώσεις και επεκτάσεις. Πρώτον, θα μπορούσε να διερευνηθεί η ενσωμάτωση περισσότερων εργαλείων για τη **συνεχή ολοκλήρωση και παράδοση (CI/CD)**, προκειμένου να βελτιωθεί η διαδικασία ανάπτυξης και διανομής του λογισμικού. Η εφαρμογή αυτών των εργαλείων θα επέτρεπε την ταχύτερη διάθεση νέων λειτουργιών, τη μείωση του χρόνου ανάπτυξης και την εξασφάλιση της σταθερότητας του λογισμικού. Ένας άλλος τομέας για μελλοντική βελτίωση είναι η **κλιμακωσιμότητα** της εφαρμογής. Η χρήση τεχνολογιών όπως τα **microservices** μπορεί να προσφέρει μεγαλύτερη ευελιξία και αποδοτικότητα στην αντιμετώπιση αυξημένων απαιτήσεων απόδοσης. Επιπλέον, η περαιτέρω βελτίωση της **ασφάλειας** με τη χρήση τεχνολογιών κρυπτογράφησης αιχμής και σύγχρονων εργαλείων διαχείρισης ταυτοποίησης χρηστών μπορεί να ενισχύσει την προστασία των δεδομένων των χρηστών. Τέλος, προτείνεται η διερεύνηση της **χρήσης AI και machine learning** για την εξατομίκευση της εμπειρίας χρήστη και τη βελτιστοποίηση της διαχείρισης δεδομένων. Οι τεχνολογίες αυτές μπορούν να προσφέρουν βελτιωμένες υπηρεσίες στους χρήστες, επιτρέποντας την καλύτερη αξιοποίηση των δεδομένων που συλλέγονται και την αυτοματοποίηση λειτουργιών.

7. Παράρτημα

7.1 Κώδικας

Ενημέρωση Πεδίων στο Storage.js

Οι αλλαγές στο αρχείο **Storage.js** περιλαμβάνουν τη διαχείριση νέων πληροφοριών χρήστη. Τα πεδία που έχουν προστεθεί είναι τα εξής:

- **First Name**
- **Last Name**
- **Clinician ID**
- **Email**
- **Contact Number**
- **Children Full Name**

Παράδειγμα κώδικα που υλοποιεί την εμφάνιση των στοιχείων αυτών:

```
<h6 style={{color: "black"}}>First Name: {user ? user.profile.firstName : "-"}</h6>

<h6 style={{color: "black"}}>Last Name: {user ? user.profile.lastName : "-"}</h6>
```

```
<h6 style={{color: "black"}}>Clinician ID: {user ? user.profile.clinicianID : "-
"}</h6>

<h6 style={{color: "black"}}>Email: {user ? user.profile.email : "-"}</h6>

<h6 style={{color: "black"}}>Contact Number: {user ? user.profile.contactNumber :
"-"}</h6>

<h6 style={{color: "black"}}>Children Full Name: {user ?
user.profile.childrenFullName : "-"}</h6>
```

Διαγραφή Credentials

Η μέθοδος deleteCredential επιτρέπει στους χρήστες να διαγράφουν credentials. Όταν γίνεται διαγραφή, εμφανίζεται ένα μήνυμα επιβεβαίωσης.

Παράδειγμα κώδικα για τη λειτουργία διαγραφής:

```
// Διαγραφή credentials

function deleteCredential(credentialID) {

   database.delete(credentialID);

   console.log("Credentials deleted successfully.");

}
```

Συνθήκες Ελέγχου και Λειτουργικότητα Verifier

Στο αρχείο **watch-verifier.js**, έχει τροποποιηθεί η συνθήκη ελέγχου για τον **Clinician ID**:

```
const isClinicianID = details["Clinician Id"] === "8";


const canUserContinue = () => {

  const atLeastOneFieldRevealed = Object.values(revealDetails).some((value) =>
value === true);
```

```
  return atLeastOneFieldRevealed && isClinicianID;
};
```

Κουμπί Σύνδεσης με Credentials

Στο αρχείο **LoginPage.js**, προστέθηκε κουμπί για τη σύνδεση με credentials, καθώς και η ανακατεύθυνση στη σελίδα επαλήθευσης:

```
<Button
  size="large"
  variant="outlined"
  startIcon={<DirectionsIcon />}
  onClick={this.onLoginCredentialsClick}
  style={styles.button}
>
  Log in with Credentials
</Button>

onLoginCredentialsClick(event) {
  event.preventDefault();
  window.location.href = "http://localhost:8080/loginCredentials";
}
```

Διαχείριση και Αποθήκευση Credentials

Η μέθοδος AddAttributes επιτρέπει την προσθήκη νέων χαρακτηριστικών σε έναν χρήστη:

```
function AddAttributes(Username, Cookie, Nonce, Signature, idProof) {
  // Επεξεργασία χαρακτηριστικών
}
```

Η μέθοδος DeleteAttributes αφαιρεί χαρακτηριστικά:

```
function DeleteAttributes(Username, Cookie, Nonce, Signature, Attributes) {
  // Διαγραφή χαρακτηριστικών
}
```

8. Συμπεράσματα & Αποτελέσματα

Τα συμπεράσματα που προκύπτουν από την παρούσα εργασία αποδεικνύουν τη σημασία της σωστής επιλογής τεχνολογικών εργαλείων και πλαισίων για την ανάπτυξη ασφαλών και αποδοτικών διαδικτυακών εφαρμογών. Μέσω της αξιοποίησης του **Spring Boot** ως πλαίσιο για την ανάπτυξη εφαρμογών backend, η εργασία κατέδειξε την απλοποίηση της ανάπτυξης και της διαχείρισης εξαρτήσεων, επιτυγχάνοντας ευελιξία και αποτελεσματικότητα στη διαδικασία του λογισμικού. Τα εργαλεία όπως το **Maven** συνέβαλαν στην αυτοματοποίηση της διαχείρισης των εξαρτήσεων, μειώνοντας σημαντικά τα σφάλματα και επιταχύνοντας τη διαδικασία ανάπτυξης.

Η χρήση του **IntelliJ IDEA** διευκόλυνε τους προγραμματιστές παρέχοντας εξειδικευμένα εργαλεία για τον εντοπισμό λαθών και την αυτόματη συμπλήρωση του κώδικα, βελτιώνοντας την παραγωγικότητα της ομάδας ανάπτυξης. Τα cookies, μέσω της σωστής διαχείρισής τους, διασφάλισαν την εμπειρία χρήστη και την προστασία των προσωπικών δεδομένων, σύμφωνα με κανονισμούς όπως το **GDPR**.

Συνολικά, η εργασία αυτή απέδειξε τη σημασία της συνεργασίας πολλών πλαισίων και τεχνολογιών για την επιτυχή ανάπτυξη μιας διαδικτυακής εφαρμογής, διασφαλίζοντας ταυτόχρονα την ασφάλεια των δεδομένων και την ευχρηστία.

9. Βιβλιογραφία

1. Bernabe, J. B., García-Rodríguez, J., Krenn, S., Liagkou, V., Skarmeta, A. & Torres, R. (2021). 'Decentralized Identity Management Systems Using Blockchain'. *Journal of Secure Computing*, 12(3), pp. 234-245.
2. Moreno, R. T., Rodríguez, J. G., López, C. T., Bernabe, J. B. & Skarmeta, A. (2020). 'Privacy-Preserving Attribute-Based Credentials with Blockchain'. *IEEE Transactions on Information Forensics and Security*, 15(5), pp. 987-998.
3. Masood, A., Chaudhry, J., Ahmad, A., Ullah, F., Rehman, A., & Khan, S. (2024). 'Blockchain and Biometric Security: Enhancing Privacy in Identity Management'. *International Journal of Advanced Computer Science*, 19(2), pp. 102-117.
4. Song, W., Liu, Y., Wang, H., & Zhao, J. (2024). 'Zero-Knowledge Proofs in Blockchain-Based Digital Identity Systems'. *Security and Privacy Journal*, 8(4), pp. 225-236.
5. Rabah, K., Jaques, S., & Liang, C. (2024). 'Decentralized Anonymous Reputation Systems Using zkSNARKs'. *Journal of Cryptographic Engineering*, 14(2), pp. 89-102.
6. Jaques, S., Rabah, K., & Malik, R. (2024). 'Improving Anonymity and Performance in Decentralized Identity Systems with Accumulators'. *Blockchain and Privacy Journal*, 6(1), pp. 57-73.
7. Hossted Team (2024). *Streamlining Java Development with Maven: Enhancing Build and Dependency Management*. Hossted. [Online]. Available at: https://hossted.com/knowledge-base/osspedia/devops/application-development/streamlining-java-development-with-maven-enhancing-build-and-dependency-management/ [Accessed: 9 Oct. 2024].
8. Apache Maven (n.d.). *Introduction to the Dependency Mechanism*. Maven Central Repository. [Online]. Available at: https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html [Accessed: 9 Oct. 2024].
9. Dev Community (n.d.). *Getting Started with Maven: A Beginner's Guide to Java Build Automation*. [Online]. Available at: https://dev.to/maven/getting-started-with-maven-a-beginners-guide-to-java-build-automation-28dd [Accessed: 9 Oct. 2024].
10. L. Cavazos et al. (2021). *A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem*. ar5iv. [Online]. Available at: https://arxiv.org/abs/2001.07808 [Accessed: 9 Oct. 2024].

11. Di Bella, E., Sillitti, A., Succi, G. (2013). *A multivariate classification of open source developers*. Information Sciences, 221, pp. 72–83. SpringerLink. Available at: https://link.springer.com [Accessed: 10 Oct. 2024].
12. Böck, H. (2012). *IntelliJ IDEA and the NetBeans Platform*. In: *The Definitive Guide to NetBeans™ Platform 7*. Apress. DOI: https://doi.org/10.1007/978-1-4302-4102-7_40 [Accessed: 10 Oct. 2024].
13. Moser, R., Pedrycz, W., Succi, G. (2008). *Analysis of the reliability of a subset of change metrics for defect prediction*. In: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 309–311. ACM. Available at: https://link.springer.com [Accessed: 10 Oct. 2024].
14. Mishra, M., Srivastava, S. (2019). *Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks*. MDPI. Available at: https://www.mdpi.com [Accessed: 10 Oct. 2024].
15. Rajput, A.S., Singh, H.P., Bang, G., Joshi, S., Patidar, T. (2022). *Comparing Spring Boot and ReactJS with Other Web Development Frameworks: A Study*. *SpringerLink*. Available at: https://link.springer.com [Accessed: 10 Oct. 2024].
16. Technology Rivers (2022). *Top Web Application Development Frameworks*. Available at: https://technologyrivers.com [Accessed: 10 Oct. 2024].
17. Pantelic, O., Jovic, K., Krstovic, S. (2022). *Cookies Implementation Analysis and the Impact on User Privacy Regarding GDPR and CCPA Regulations*. *Sustainability*, 14(9), 5015. Available at: https://doi.org/10.3390/su14095015.
18. Van Lieshout, M., Kolkman, D. (2020). *A Secure Cookie Protocol and its Implementation in HTTPS*. *Journal of Web Engineering*, 19(4), 351-368.

Παράρτημα

Λεξιλόγιο Ορών

Identity management (IdM) = Σύστημα διαχείρισης ταυτότητας

Identity service providers(IdSP) = Πάροχος υπηρεσιών ταυτότητας

Token = Διαπιστευτήριο

One Time Password = Κωδικός πρόσβασης μίας χρήσης

Security Policies = Πολιτικές απορρήτου

One Time Password = Κωδικός πρόσβασης μίας χρήσης

Relying Party = Τρίτο Μέρος

Attributes = Χαρακτηριστικά

Service Provider (SP) = Πάροχος υπηρεσιών

Certificate Authority(CA) = Αρχή πιστοποίησης

Privacy Attribute-based Credentials (P-ABC) = Διαπιστευτήρια βάσει χαρακτηριστικών απορρήτου

User = Χρήστης

Issuer = Εκδότης

Identity provider (IdP) = Πάροχος ταυτότητας

Virtual Identity provider (vIdP) = Εικονικός Πάροχος Ταυτότητας

Distributed authentication = Κατανεμημένος έλεγχος ταυτότητας

Distributed credential module = Κατανεμημένη μονάδα διαπιστευτηρίων

Signature = Υπογραφή

Constructors = Κατασκευαστής

Empty constructor = Προεπιλεγμένος κατασκευαστής

Registration = Εγγραφή

Issuance = Έκδοση

Presentation = Παρουσίαση

JSON (JavaScript Object Notation) = Συμβολισμός αντικειμένου JavaScript