ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΒΙΒΛΙΟΘΗΚΗΣ ΓΙΑ ΑΥΘΕΝΤΙΚΟΠΟΙΗΣΗ ΧΡΗΣΤΩΝ ΤΟΥ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΜΑΡΓΑΡΙΤΑ ΚΟΥΜΠΟΥΡΗ

ΑΜ : 18390109

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΙΩΑΝΝΗΣ ΒΟΓΙΑΤΖΗΣ

ΚΑΘΗΓΗΤΗΣ

ΑΙΓΑΛΕΩ-ΑΘΗΝΑ, ΙΟΥΛΙΟΣ, 2023

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και Ανάπτυξη Βιβλιοθήκης για Αυθεντικοποίηση Χρηστών του Πανεπιστημίου Δυτικής Αττικής.

Μαργαρίτα Κουμπούρη

Α.Μ: 18390109

Εισηγητής:

ΙΩΑΝΝΗΣ ΒΟΓΙΑΤΖΗΣ, ΚΑΘΗΓΗΤΗΣ

Η διπλωματική εργασία εγκρίθηκε και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Ιωάννης Βογιατζής, Καθηγητής ΠΑΔΑ	Χρήστος Τρούσσας, Επίκουρος Καθηγητής ΠΑΔΑ	Γεώργιος Μελετίου, ΕΔΙΠ
Ioannis Voyiatzis Digitally signed by Ioannis Voyiatzis Date: 2024.03.07 11:37:00 +02'00' (Υπογραφή)	Christos Troussas Digitally signed by Christos Troussas Date: 2024.03.07 10:00:10 +02'00' (Υπογραφή)	Georgios Meletiou Digitally signed by Georgios Meletiou Date: 2024.03.06 22:18:41 +02'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη ΚΟΥΜΠΟΥΡΗ ΜΑΡΓΑΡΙΤΑ του ΧΡΥΣΟΣΤΟΜΟΥ με αριθμό μητρώου 18390109 φοιτήτρια του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Βεβαιώνω ότι είμαι συγγραφέας αυτή της διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από εμένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Η Δηλούσα

Κουμπούρη Μαργαρίτα

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία αποτελεί προϊόν της περάτωσης των σπουδών μου, καθώς και μια σειρά επίμονων προσπαθειών και εγχειρημάτων που πραγματοποιήθηκαν πάνω σε ένα ιδιαίτερα ενδιαφέρον γνωστικό αντικείμενο, το οποίο περικλείει μια σφαιρική γνώση τόσο συστημάτων, όσο και ασφάλειας. Η προσπάθεια μου αυτή στηρίχθηκε έμπρακτα από τον επιβλέποντα καθηγητή μου, κ. Γεώργιο Μελετίου, τον οποίο θα ήθελα να ευχαριστήσω. Τέλος θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Ιδρύματος για την καθοδήγηση που παρείχαν, καθώς και την οικογένειά μου για τη συμπαράσταση κατά τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία αφορά στο Σχεδιασμό και την Ανάπτυξη μιας Βιβλιοθήκης Αυθεντικοποίησης χρηστών του Πανεπιστημίου Δυτικής Αττικής. Συγκεκριμένα, στο Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών ή ακόμα και σε άλλα τμήματα του Ιδρύματος εντοπίζεται η ανάγκη δημιουργίας εφαρμογών που απαιτούν την αυθεντικοποίηση χρηστών, μέσω του ιδρυματικού τους λογαριασμού. Η πλειοψηφία των εφαρμογών έχει περιορισμένη χρονικά χρήση και χρησιμοποιείται ως πρακτικό/εργαστηριακό μέρος των διπλωματικών εργασιών. Η ανάγκη για ανάπτυξη περισσότερων ηλεκτρονικών ιδρυματικών υπηρεσιών και εφαρμογών, αυξάνει αναλογικά τόσο την ανάγκη για καλύτερη διαχείριση του φόρτου του συστήματος (υπολογιστές/ ανθρώπινο δυναμικό) όσο και την ανάγκη για έλεγχο πρόσβασης σε διακομιστές του ιδρύματος που αφορούν στην αυθεντικοποίηση χρηστών (όπως ο διακομιστής CAS).

Στην παρούσα διπλωματική εργασία διερευνάται η τεχνολογία SSO (Single Sign-On) και οι τρόποι εφαρμογής και ανάπτυξής της, με τη συμβολή των πρωτοκόλλων CAS (Central Authentication Service) και SAML (Security Assertion Markup Language). Συγκεκριμένα, αναλύεται η τεχνολογία SSO, καθώς και τα πρωτόκολλα CAS και SAML, όσον αφορά στην υλοποίησή τους στο κομμάτι της αυθεντικοποίησης που αποτελεί έναν από τους βασικούς πυλώνες τις ασφάλειας. Για τη δόμηση της βιβλιοθήκης χρησιμοποιήθηκε το μοντέλο client-server, το οποίο την καθιστά εύκολα εφαρμόσιμη στην ήδη υπάρχουσα δομή/αρχιτεκτονική του συστήματος του ιδρύματος.

Βασικός άξονας της παρούσας διπλωματικής εργασίας, αποτελεί η κατανόηση των τεχνολογιών που χρησιμοποιούνται για αυθεντικοποίηση, η μεθοδολογία που ακολουθήθηκε για την εφαρμογή τους, καθώς και ο τρόπος με τον οποίο δομήθηκε η βιβλιοθήκη. Τέλος, θα παρουσιαστεί η χρηστικότητα της βιβλιοθήκης, ο πηγαίος κώδικας και τα συμπεράσματα.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αυθεντικοποίηση, SSO, CAS, SAML, server, client, java, digital certificate

ABSTRACT

The present thesis concerns the design and development of a User Authentication Library of the University of West Attica. More specifically, in the Department of Informatics and Computer Engineering or even in other departments of the Foundation, the need to create applications that require user authentication through their institutional account is identified. The majority of the applications function using time sessions as a medium of security and are used as a practical tool of the diploma work for many students. The need to develop more electronic institutional services and applications, proportionally increases both the need for better system load management (computers/human resources) and the need for access control to the institution's servers related to user authentication (such as the CAS server). In this thesis, the SSO (Single Sign-On) technology and its application and development methods are investigated, with the contribution of the CAS (Central Authentication Service) and SAML (Security Assertion Markup Language) protocols. In particular, the SSO technology, as well as the CAS and SAML protocols, are analyzed in terms of their implementation in the authentication part, which is one of the main pillars of security. The client-server model was used to build the library, which makes it easily applicable to the already existing structure/architecture of the institution's system.

The main axis of this thesis is the understanding of the technologies used for authentication, the methodology followed concerning their application, as well as the way in which the library is constructed. Finally, the usability evaluation of the library, the source code and the final conclusion will also be presented.

KEYWORDS: Authentication, SSO, CAS, SAML, server, client, java, digital certificate

Πίνακας περιεχομένων

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

SSO	Single Sign-On
CAS	Central Authentication Service
SAML	Security Assertion Markup Language
SP	Service Provider
IdP	Identity Provider
OTP	One Time Password
MFA	Multifactor Authentication
MITM	Man-in-the-middle
API	Application Programming Interface
ADFS	Microsoft Active Directory Federation Services
RBAC	Role-based Access Control
ABAC	Attribute-based Access Control
RDP	Remote Desktop Protocol
OAuth	Open Authorization
ST	Service Ticket
EER	Enhanced Entity-Relationship
CA	Certified Authority
POM	Project Object Model
SOAP	Simple Object Access Protocol

Κεφάλαιο 1. Εισαγωγή

Στο παρόν κεφάλαιο περιγράφεται αναλυτικά το πλαίσιο και το αντικείμενο της διπλωματικής εργασίας, καθώς και η σύνταξη και δόμησή της σε κεφάλαια.

1.1 Πλαίσιο, περιγραφή και σκοπός διπλωματικής εργασίας

Η συνεχής και αδιάκοπη ανάπτυξη της τεχνολογίας έχει ως αποτέλεσα την ανάπτυξη πολλαπλών υπηρεσιών, εφαρμογών και ιστοσελίδων που ικανοποιούν σε αρκετά μεγάλο και επαρκή βαθμό τις ανάγκες της πλειοψηφίας των χρηστών. Ακόμα, στο μεγαλύτερο ποσοστό των υπηρεσιών αυτών απαιτείται η εγγραφή των χρηστών, με σκοπό την καλύτερη και προσωποποιημένη εξυπηρέτησή τους. Σε ένα σύστημα που παρέχει ένα μεγάλο αριθμό πόρων, είναι λογικό να υπάρχει η δυσχέρεια της συνεχούς δημιουργίας νέων διαπιστευτηρίων, με σκοπό την είσοδο στις υπηρεσίες αυτές και την ελαχιστοποίηση εισόδου μη εξουσιοδοτημένων χρηστών. Το γεγονός αυτό καθιστά δύσκολη την επίτευξη δημιουργίας νέων κωδικών, οι οποίοι αποδεικνύονται αρκετά ισχυροί και ανθεκτικοί, προς αποφυγή παντός τύπου επιθέσεων.

Το κύριο αντικείμενο, λοιπόν, της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η ανάπτυξη μιας βιβλιοθήκης, με την οποία οι χρήστες των ιδρυματικών λογαριασμών θα αποκτούν πρόσβαση σε ένα μεγάλο εύρος εφαρμογών και υπηρεσιών του πανεπιστημίου, με την αποκλειστική χρήση των ιδρυματικών τους διαπιστευτηρίων. Ως στόχος της παρούσας διπλωματικής εργασίας, λοιπόν, τίθεται η ευκολότερη εξυπηρέτηση των χρηστών, καθώς η ανάληψη του βάρους και η μείωση του χρόνου εύρεσης διαφορετικών και ανθεκτικών προς επιθέσεις κωδικών.

Συγκεκριμένα στον τομέα της κυβερνοασφάλειας, το Single Sign-On (SSO) προσφέρει μια ισχυρή άμυνα έναντι διαφόρων απειλών ασφαλείας ενοποιώντας τις διαδικασίες ελέγχου ταυτότητας. Αρχικά η τεχνολογία SSO μετριάζει τον κίνδυνο επιθέσεων που σχετίζονται με κωδικούς πρόσβασης, όπως το Phishing και το Γέμισμα Διαπιστευτηρίων (Credential Stuffing). Δεδομένου ότι οι χρήστες χρειάζεται να θυμούνται μόνο ένα σύνολο διαπιστευτηρίων, είναι λιγότερο πιθανό να πέσουν θύματα απάτης ηλεκτρονικού «ψαρέματος» (Phishing) που προσπαθούν να κλέψουν πολλούς κωδικούς πρόσβασης. Επιπλέον, το SSO σε ορισμένες περιπτώσεις πολύ υψηλής ασφάλειας ενσωματώνει έλεγχο ταυτότητας πολλαπλών παραγόντων (Multifactor Authentication – MFA), το οποίο προσθέτει ένα επιπλέον επίπεδο ασφαλείας και προστατεύει από μη

εξουσιοδοτημένη πρόσβαση, ακόμα και αν τα διαπιστευτήρια παραβιάζονται. Δεύτερον, το SSO βοηθά στην αποτροπή επιθέσεων man-in-the-middle (MITM) κρυπτογραφώντας τα δεδομένα ελέγχου ταυτότητας που ανταλλάσσονται μεταξύ της συσκευής του χρήστη και του παρόχου SSO, μειώνοντας την πιθανότητα υποκλοπής και χειραγώγησης ευαίσθητων διαπιστευτηρίων. Ενισχύοντας τους μηχανισμούς ελέγχου ταυτότητας και μειώνοντας την επιφάνεια επίθεσης, το SSO ενισχύει σημαντικά τη στάση της κυβερνοασφάλειας ενός οργανισμού, παρέχοντας παράλληλα μια απρόσκοπτη εμπειρία χρήστη.

Το πλαίσιο στο οποίο αναπτύχθηκε η πρωτοβουλία για την ανάληψη ενός τέτοιου εγχειρήματος, είναι η συνεχής ανάπτυξη των δομών του Πανεπιστημίου Δυτικής Αττικής, με την ανάπτυξη καινούργιων λογισμικών και υπηρεσιών που αποσκοπούν στη διερεύνηση των οριζόντων των φοιτητών του, αλλά και στη διευκόλυνση του διδακτικού προσωπικού του.

1.2 Δομή διπλωματικής εργασίας

Η παρούσα διπλωματική εργασία αποτελείται από έξι κεφάλαια, στα οποία εμπεριέχεται η εισαγωγή και στη συνέχεια το θεωρητικό υπόβαθρο (ή βιβλιογραφική ανασκόπηση) στην οποία περιγράφονται τα βασικά θεωρητικά στοιχεία που διέπουν το αντικείμενο της συγκεκριμένης εργασίας, η αρχιτεκτονική της βιβλιοθήκης, η μεθοδολογία που ακολουθήθηκε για τη δόμηση της βιβλιοθήκης, η αξιολόγησή της και τα τελικά συμπεράσματα.

Συγκεκριμένα, το δεύτερο κεφάλαιο περιέχει το Θεωρητικό Υπόβαθρο και Βιβλιογραφική Ανασκόπηση, δηλαδή θα οριστούν οι βασικές έννοιες των τεχνολογιών και των πρωτοκόλλων/προτύπων που θα χρησιμοποιηθούν για την υλοποίηση της βιβλιοθήκης, καθώς και το μοντέλο της αρχιτεκτονικής. Ακόμα, θα αναφερθεί η αναγκαιότητα εγκατάστασης πιστοποιητικού με σκοπό τη σωστή λειτουργία και παροχή της υπηρεσίας.

Στο τρίτο κεφάλαιο, αναφέρεται η Αρχιτεκτονική της βιβλιοθήκης Java που υλοποιήσαμε. Ειδικότερα, θα αναφερθεί η προεπεξεργασία και ο σχηματισμός της βιβλιοθήκης δομικά με σκοπό την εφαρμογή των καλύτερων πρακτικών δόμησης μιας βιβλιοθήκης, αλλά και το μοντέλο επικοινωνίας που χρησιμοποιήθηκε. Αναλύονται, δηλαδή, τα βασικά στοιχεία και οι κώδικες που χρησιμοποιήθηκαν.

Στο τέταρτο κεφάλαιο θα παρουσιαστεί το περιβάλλον στο οποίο υλοποιήθηκε η βιβλιοθήκη καθώς και τα πρωτόκολλα, αλλά και το μοντέλο επικοινωνίας.

Στο πέμπτο κεφάλαιο θα παρουσιαστούν τα αποτελέσματα της αξιολόγησης που διενεργήθηκε στη βιβλιοθήκη, αξιολογώντας αρκετές πλευρές της χρηστικότητας και αποτελεσματικότητάς της.

Τέλος, στο έκτο κεφάλαιο θα παρουσιαστούν τα τελικά συμπεράσματα τα οποία προκύπτουν από τη διπλωματική εργασία που εκπονήθηκε. Γενικά, θα παρατεθούν συμπεράσματα όσον αφορά στο σχεδιασμό και στην υλοποίηση της βιβλιοθήκης, καθώς και στην έρευνα που προηγήθηκε για να επιλεχθούν οι συγκεκριμένες τεχνολογίες και πρωτόκολλα.

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο και Βιβλιογραφική Ανασκόπηση

Στο παρόν κεφάλαιο εξετάζεται η δομή του Single Sign-On και δίνεται ιδιαίτερη έμφαση στη σημασία της, καθώς και τη συμβολή της στη συνολική εικόνα της ασφάλειας ενός συστήματος. Στη συνέχεια, εξετάζεται ο ρόλος και οι λόγοι επιλογής του πρωτοκόλλου CAS και του προτύπου SAML. Τέλος, αναφέρονται οι ήδη υπάρχουσες τεχνολογίες και βιβλιοθήκες Single Sign-On (SSO) και οι δυνατότητες εφαρμογής τους, καθώς και τα πλεονεκτήματα και τα μειονεκτήματά τους.

2.1 Προεπισκόπηση τεχνολογίας Single Sign-On

Γενικά το Single Sign-On (SSO) αποτελεί μια μέθοδο αυθεντικοποίησης, η οποία επιτρέπει στους χρήστες να αυθεντικοποιούνται ασφαλώς σε πολλαπλές και διαφορετικές εφαρμογές και ιστοσελίδες, χρησιμοποιώντας μόνο ένα σύνολο διαπιστευτηρίων. Τα διαπιστευτήρια (credentials) αυτά, αποτελούνται στην πλειοψηφία των περιπτώσεων από ένα ζευγάρι ενός *αναγνωριστικού/ονόματος χρήστη* (username) και ένα *μυστικό κωδικό* (password).

Ως γενική εικόνα, το SSO βασίζεται στη δημιουργία μιας σχέσης εμπιστοσύνης ανάμεσα σε μια εφαρμογή, η οποία είναι γνωστή ως *Πάροχος Υπηρεσιών* (Service Provider) και σε ένα *Πάροχος Υπηρεσιών* (Identity Provider). Η συγκεκριμένη σχέση εμπιστοσύνης βασίζεται συνήθως σε ένα πιστοποιητικό, το οποίο ανταλλάσσεται μεταξύ του Παρόχου Υπηρεσίας και του Παρόχου Ταυτότητας. Το πιστοποιητικό αυτό μπορεί να χρησιμοποιηθεί για την υπογραφή στοιχείων ταυτότητας που αποστέλλονται από τον Πάροχο Ταυτότητας στον Πάροχο Υπηρεσιών, έτσι ώστε ο Πάροχος Υπηρεσιών να γνωρίζει ότι προέρχονται από αξιόπιστη πηγή. Με την τεχνολογία SSO, τα δεδομένα ταυτότητας που προαναφέρθηκαν, έχουν τη μορφή συμβόλων (token), τα οποία περιέχουν αναγνωριστικά κομμάτια πληροφοριών σχετικά με τον χρήστη, όπως το όνομα χρήστη του.

2.1.1 Ροή Λειτουργίας SSO

Η ροή λειτουργίας της τεχνολογίας SSO, δομείται ως εξής:

Αρχικά ο χρήστης περιηγείται στην εφαρμογή ή την ιστοσελίδα (Service Provider), στην οποία επιθυμεί να αποκτήσει πρόσβαση. Ο Πάροχος Υπηρεσίας στέλνει ένα αναγνωριστικό (token) – το

οποίο περιέχει πληροφορίες που αφορούν το χρήστη, όπως για παράδειγμα το όνομα χρήστη που χρησιμοποιεί ή τη διεύθυνση ηλεκτρονικού του ταχυδρομείου) στο σύστημα SSO (Identity Provider), ως μέρος ενός αιτήματος αυθεντικοποίησης του χρήστη που επιθυμεί να χρησιμοποιεί την εν λόγω υπηρεσία.

Το σύστημα του SSO (Identity Provider) ελέγχει να διαπιστώσει αν ο χρήστης που ζητά πρόσβαση στην υπηρεσία έχει αυθεντικοποιηθεί ξανά, στην οποία περίπτωση θα του δώσει πρόσβαση στην εφαρμογή που λειτουργεί ως Πάροχος Υπηρεσίας (Service Provider).

Εάν συμβεί επιτυχώς το προαναφερθέν γεγονός, τότε μόλις ο Πάροχος Ταυτότητας (Identity Provider) επικυρώσει το χρήστη στέλνει πίσω στον Πάροχο Υπηρεσίας ένα αναγνωριστικό (token), με το οποίο επιβεβαιώνεται η επιτυχής αυθεντικοποίηση του χρήστη. Το συγκεκριμένο αναγνωριστικό μεταφέρεται στον Πάροχο Υπηρεσίας μέσω του προγράμματος περιήγησης του χρήστη και επικυρώνεται εκεί, σύμφωνα με τη σχέση εμπιστοσύνης που έχει ήδη εγκαθιδρυθεί ανάμεσα σε αυτόν και τον Πάροχο Ταυτότητας κατά την αρχική παραμετροποίηση. Τέλος, ο χρήστης καταφέρνει να αποκτήσει πρόσβαση στον πάροχο της υπηρεσίας που επιθυμούσε.

Στην περίπτωση που ο χρήστης δεν έχει αυθεντικοποιηθεί ήδη, μετά από τον έλεγχο του Παρόχου Ταυτότητας, προστίθεται ένα επιπλέον βήμα. Συγκεκριμένα, στην περίπτωση που ο χρήστης δεν έχει αυθεντικοποιηθεί ξανά, θα του ζητηθεί να πραγματοποιήσει την είσοδό του στην υπηρεσία, εισάγοντας τα προσωπικά του διαπιστευτήρια, τα οποία αποτελούνται από ένα ζευγάρι ονόματος χρήστη και μυστικού κωδικού, ή άλλης μορφής αυθεντικοποίηση, όπως ο κωδικός μιας χρήσης (One-Time Password – OTP).

Σημαντικό είναι να αναφέρουμε, ότι το SSO Token, το οποίο έχει ενεργό ρόλο στη ροή λειτουργικότητας του συστήματος SSO. Ειδικότερα, είναι μια συλλογή δεδομένων ή πληροφοριών, το οποίο μεταφέρεται από ένα σύστημα σε ένα άλλο, χρησιμοποιώντας τη μεθοδολογία/τεχνολογία SSO. Τα αναγνωριστικά (tokens) πρέπει κανονικά να είναι ψηφιακά υπογρεγραμμένα, έτσι ώστε ο παραλήπτης τους να δύναται να πιστοποιήσει ότι το συγκεκριμένο αναγνωριστικό που περιλαμβάνει, προέρχεται από μια έμπιστη πηγή. Το πιστοποιητικό, το οποίο χρησιμοποιείται για την ψηφιακή υπογραφή, ανταλλάσσεται κατά τη διάρκεια της αρχικής παραμετροποίησης.

2.1.2 Σύγκριση SSO Software με SSO Solution

Συχνά παρατηρείται η αναφορά και η διάκριση ανάμεσα σε SSO λογισμικό (SSO Software) και SSO Λύση (SSO Solution). Και οι δύο έννοιες σχετίζονται με το γενικό πλαίσιο του συστήματος του SSO.

Γενικά, ένα κομμάτι λογισμικού συνιστά κάτι που είναι εγκατεστημένο εξ' αρχής και είναι συνήθως σχεδιασμένο για να εκτελεί μόνο ένα συγκεκριμένο σύνολο εργασιών. Μια λύση υποδεικνύει τη δυνατότητα επέκτασης ή προσαρμογής των δυνατοτήτων του βασικού προϊόντος. Ένας τρόπος αναφοράς σε μια εταιρεία που παράγει ή φιλοξενεί τη λύση, είναι με την έννοια του παρόχου. Ένα χαρακτηριστικό παράδειγμα, αποτελεί το OneLogin, το οποίο είναι γνωστό ως πάροχος λύσεων SSO.

Πιο συγκεκριμένα, το SSO Software, αναφέρεται σε ένα συγκεκριμένο λογισμικό ή εργαλεία, τα οποία διευκολύνουν τη λειτουργικότητα της δομής του SSO εντός της υποδομής ενός οργανισμού. Όπως αναφέρθηκε παραπάνω, το λογισμικό SSO, περιλαμβάνει στοιχεία, όπως παρόχους ταυτότητας (IdP), παρόχους υπηρεσιών (SP), και πρωτόκολλα/πρότυπα για έλεγχο ταυτότητας και εξουσιοδότηση, όπως το SAML, OAuth.

Η τεχνολογία SSO παρέχει την απαραίτητη λειτουργικότητα για τον έλεγχο ταυτότητας των χρηστών, τη διαχείριση της ταυτότητας και της πρόσβασης χρήστη και την απρόσκοπτη πρόσβαση σε πολλαπλές εφαρμογές ή υπηρεσίες χρησιμοποιώντας ένα μοναδικό σύνολο διαπιστευτηρίων. Τα παραδείγματα λογισμικού SSO περιλαμβάνουν εμπορικές λύσεις SSO, πλαίσια SSO ανοιχτού κώδικα ή προσαρμοσμένες υλοποιήσεις SSO που χρησιμοποιούν συγκεκριμένες γλώσσες προγραμματισμού ή βιβλιοθήκες.

Παρομοίως, μια λύση SSO (SSO Solution), αναφέρεται σε ένα ολοκληρωμένο πακέτο, που περιλαμβάνει το λογισμικό SSO και σχετικές υπηρεσίες, υποστήριξη και ειδίκευση υλοποίησης. Οι λύσεις SSO σχεδιάζονται με σκοπό να ανταποκρίνονται στις ανάγκες που εκφράζονται από τους οργανισμούς που επιδιώκουν την εφαρμογή του συστήματος SSO στα περιβάλλοντα ανάπτυξής τους.

Τέλος, προς καλύτερη κατανόηση της διαφοράς μεταξύ ενός λογισμικού SSO και μιας λύσης SSO, αναφέρεται ότι μια λύση SSO (SSO Solution), είναι δυνατόν να περιέχει τα εξής:

- *Το λογισμικό SSO* : Το θεμελιώδες λογισμικό ή τα εργαλεία τα οποία επιτρέπουν τη λειτουργικότητα της μεθόδου SSO.
- *Υπηρεσίες Ανάπτυξης και Ενοποίησης* : Παραμετροποίηση, οδηγίες εγκατάστασης και ενσωμάτωση του λογισμικού SSO στην ήδη εγκατεστημένη υποδομή του οργανισμού.
- *Παραμετροποίηση και Επωνυμία* : Διαδικασία παραμετροποίησης και προσαρμογής της λύσης SSO, με σκοπό τη διαμόρφωσή της, σύμφωνα με τις ανάγκες, τις απαιτήσεις και την εμπειρία των χρηστών του οργανισμού.
- *Εκπαίδευση και Υποστήριξη* : Παροχή εκπαιδευτικών πόρων και συνεχούς τεχνικής υποστήριξης, με σκοπό την άμεση βοήθεια ως προς τη διαχείριση και αντιμετώπιση προβλημάτων ενός συστήματος SSO που έχει ενσωματώσει ένας οργανισμός.
- *Συμβουλευτικές Υπηρεσίες :* Παροχή συμβουλευτικών οδηγιών από εμπειρογνώμονες σχετικά με τις βέλτιστες πρακτικές SSO, ζητήματα ασφαλείας και αρχιτεκτονικό σχεδιασμό.

2.1.3 Υπάρχουσες λύσεις SSO και Τύποι SSO

Στην παρούσα ενότητα, θα εξεταστούν οι υπάρχουσες λύσεις, βιβλιοθήκες και πρωτόκολλα SSO που χρησιμοποιούνται συνήθως για το συνεχή έλεγχο ταυτότητας και την εξουσιοδότηση σε πολλές εφαρμογές και υπηρεσίες. Επιπλέον, θα συζητηθούν αρκετοί διαφορετικοί τύποι προσεγγίσεων SSO, με σκοπό την παροχή μιας ολοκληρωμένης κατανόησης της δομής του SSO.

2.1.3.1 Υπάρχουσες SSO Λύσεις και Πλατφόρμες

Παρατηρείται ότι στην αγορά υπάρχουν αρκετές καθιερωμένες λύσεις και πλατφόρμες SSO. Οι λύσεις αυτές, προσφέρουν ολοκληρωμένες δυνατότητες SSO μαζί με πρόσθετες δυνατότητες για διαχείριση ταυτότητας, ασφάλεια και έλεγχο πρόσβασης χρηστών.

Υπάρχουν τα εξής χαρακτηριστικά παραδείγματα:

- *Microsoft Azure Active Directory (Azure AD):* Το Azure AD παρέχει ισχυρές δυνατότητες SSO και ενσωματώνεται κατάλληλα στις υπηρεσίες cloud της Microsoft.
- *Okta*: Δημοφιλής πλατφόρμα διαχείρισης ταυτότητας και πρόσβασης που βασίζεται σε σύννεφο που προσφέρει ολοκληρωμένη λειτουργικότητα SSO.

- *OneLogin*: Πλατφόρμα διαχείρισης ταυτότητας που βασίζεται σε σύννεφο που παρέχει δυνατότητες SSO, ελέγχου ταυτότητας πολλαπλών παραγόντων και γνωρίσματα διάταξης χρηστών.
- *Google Cloud Identity Platform:* Πλατφόρμα που παρέχει λειτουργικότητα SSO με επιλογές ενοποίησης για το Google Workspace και άλλες υπηρεσίες σύννεφου.

2.1.3.2 SSO Βιβλιοθήκες και Πλαίσια (Frameworks)

Υπάρχει πληθώρα διαθέσιμων βιβλιοθηκών και πλαισίων SSO, οι οποίες μπορούν να αξιοποιηθούν για την εφαρμογή της λειτουργικότητας του SSO σε εφαρμογές αρκετών οργανισμών. Ο εν λόγω βιβλιοθήκες παρέχουν τα απαραίτητα εργαλεία και Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interface – API) για την αλληλεπίδραση με τα πρωτόκολλα και τα πρότυπα του SSO.

Κάποιες από τις ευρέως χρησιμοποιούμενες βιβλιοθήκες είναι οι εξής:

- *Spring Security (Java)*: Το Spring Security αποτελεί ένα αρκετά γνωστό πλαίσιο (framework) της Java, το οποίο προσφέρει λειτουργικότητα SSO και εκτεταμένες δυνατότητες ασφάλειας για εφαρμογές που απευθύνονται σε εταιρείες.
- *Passport.js (Node.js):* Αποτελεί ένα ενδιάμεσο λογισμικό ελέγχου ταυτότητας μικρής πολυπλοκότητας, για το Node.js, το οποίο επιστρατεύει πολλαπλές και διάφορες στρατηγικές ελέγχου ταυτότητας, συμπεριλαμβάνοντας τα πρωτόκολλα που διέπουν το SSO, όπως το OAuth.
- *SimpleSAMLphp (PHP):* Βιβλιοθήκη PHP, η οποία αναλαμβάνει την υλοποίηση των λειτουργιών του SSO, οι οποίες βασίζονται στο πρότυπο SAML, και απευθύνονται σε εφαρμογές γραμμένες σε PHP.
- *Satellizer (AngularJS):* Βιβλιοθήκη AngularJS, που υλοποιεί τις λειτουργίες του SSO και ελέγχου ταυτότητας σε εφαρμογές γραμμένες σε AngularJS.

2.1.3.3 SSO Πρωτόκολλα και Πρότυπα

Η λειτουργικότητα του SSO πραγματοποιείται με βάση κάποια καθιερωμένα πρωτόκολλα και πρότυπα. Η χρήση των πρωτοκόλλων αυτών είναι ιδιαίτερα σημαντική για την εφαρμογή του συστήματος του SSO, καθώς εξασφαλίζεται την ασφάλεια και τη διαλειτουργικότητά του.

Κάποια από τα πιο συχνά χρησιμοποιούμενα πρωτόκολλα και πρότυπα SSO, είναι τα παρακάτω:

- *OAuth 2.0:* Ο OAuth 2.0 είναι ένα πρωτόκολλο βιομηχανικού προτύπου, το οποίο χρησιμοποιείται κυρίως για σενάρια ανάθεσης πρόσβασης, όμως μπορεί να χρησιμοποιηθεί επίσης και για το σύστημα του SSO, αξιοποιώντας τις δυνατότητες εξουσιοδότησής του.
- *OpenID Connect:* Το OpenID Connect είναι ένα στρώμα ταυτότητας βασισμένο στη δόμηση του OAuth 2.0. Συγκεκριμένα, παρέχει έναν τυποποιημένο τρόπο απόκτησης πληροφοριών ταυτότητας χρήστη και επιτρέπει τη λειτουργικότητα SSO.
- *JSON Web Tokens (JWT)*: Πρότυπο το οποίο επιτρέπει τη διαμοίραση πληροφοριών μεταξύ δύο οντοτήτων, συνήθως μεταξύ ενός πελάτη (client) και ενός εξυπηρετητή (server). Περιέχει JSON Αντικείμενα, τα οποία περιέχουν την πληροφορία, η οποία διαμοιράζεται.

2.2 Προεπισκόπηση πρωτόκολλο CAS

Στην ενότητα αυτή, θα γίνει μια λεπτομερής επισκόπηση του πρωτοκόλλου Κεντρικής Υπηρεσίας Ελέγχου Ταυτότητας (Central Authentication Service), το οποίο θα χρησιμοποιηθεί για τη δόμηση της βιβλιοθήκης της παρούσας διπλωματικής εργασίας.

Το πρωτόκολλο Κεντρικής Υπηρεσίας Ελέγχου Ταυτότητας (Central Authentication Service – CAS), είναι ένα ευρέως γνωστό πρωτόκολλο Single Sign-On (SSO) ανοιχτού κώδικα που επιτρέπει στις διαδικτυακές εφαρμογές να αυθεντικοποιούν τους χρήστες και να τους παρέχουν πρόσβαση σε πολλές υπηρεσίες χωρίς να απαιτούν επαναλαμβανόμενα διαπιστευτήρια σύνδεσης.

Το CAS ακολουθεί ένα μοντέλο πελάτη-εξυπηρετητή, στο οποίο ο εξυπηρετητής CAS χειρίζεται τη διαδικασία ελέγχου ταυτότητας και οι εφαρμογές-πελάτες βασίζονται στο διακομιστή για την αυθεντικοποίηση του χρήστη. Βασικός σκοπός του πρωτοκόλλου CAS, είναι η αποσύνδεση της διαδικασίας αυθεντικοποίησης από μεμονωμένες εφαρμογές και η συγκέντρωσή της σε έναν αποκλειστικό διακομιστή CAS.

Το πρωτόκολλο CAS, μπορεί να χρησιμοποιηθεί για τον έλεγχο ταυτότητας μη αξιόπιστων διαδικτυακών εφαρμογών που απαιτούν ένα εισιτήριο υπηρεσίας (service ticket) για την πρόσβαση σε αυτές. Σημαντικό είναι να σημειωθεί ότι το συγκεκριμένο πρωτόκολλο, αποτελεί το εργαλείο ταυτοποίησης ενός χρήστη, έννοια η οποία διαφέρει από αυτή της εξουσιοδότησης ενός χρήστη. Η εξουσιοδότηση είναι συγκεκριμένη για την πραγματική εφαρμογή.

2.2.1 Βασικά δομικά στοιχεία/οντότητες του πρωτοκόλλου CAS

Το πρωτόκολλο CAS και συνεπώς και η ροή αυθεντικοποίησής του, περιλαμβάνει τα παρακάτω δομικά στοιχεία/οντότητες:

- *CAS Διακομιστής (CAS Server)*: Είναι υπεύθυνος για την αυθεντικοποίηση των χρηστών, τη διαχείριση των διαπιστευτηρίων των χρηστών και την έκδοση των εισιτηρίων υπηρεσίας (Service Tickets), μετά από μια επιτυχημένη αυθεντικοποίηση. Έχει, γενικά, το ρόλο της γενικής αρχής αυθεντικοποίησης στο περιβάλλον του πρωτοκόλλου CAS.
- *CAS πελάτης (CAS Client)*: Αποτελεί τη διαδικτυακή εφαρμογή ή υπηρεσία, η οποία χρησιμοποιεί τον CAS διακομιστή για την αυθεντικοποίηση του χρήστη. Ο CAS πελάτης,

επικοινωνεί με τον CAS διακομιστή, με σκοπό να επικυρώσει τα εισιτήρια υπηρεσίας και να αποκτήσει πληροφορίες σχετικά με το χρήστη.

- *Υπηρεσία (Service)*: Η υπηρεσία αντιπροσωπεύει τις ανεξάρτητες διαδικτυακές εφαρμογές ή υπηρεσίες, τι οποίες οι χρήστες επιθυμούν να αποκτήσουν πρόσβαση. Οι υπηρεσίες αυτές βασίζονται στον CAS διακομιστή για την αυθεντικοποίηση του χρήστη και ανακτούν τα απαραίτητα στοιχεία διαπιστευτηρίων του χρήστη, με σκοπό να παρέχουν πρόσβαση στους πόρους τους. Στα πλαίσια της έννοιας της υπηρεσίας, μπορεί να αναφερθεί το γεγονός ότι η διαδικτυακή εφαρμογή αποτελεί τη δομή, η οποία αναζητά τον έλεγχο ταυτότητας. Αντίστοιχα σε ένα Back-end Service, το πρωτόκολλο CAS μπορεί επίσης να περιλαμβάνει έναν Διακομιστή Βάσης Δεδομένων (Database Server), ο οποίος δε διαθέτει μια προσωπική HTTP διεπαφή, αλλά έχει τη δυνατότητα να επικοινωνεί με μια διαδικτυακή εφαρμογή.

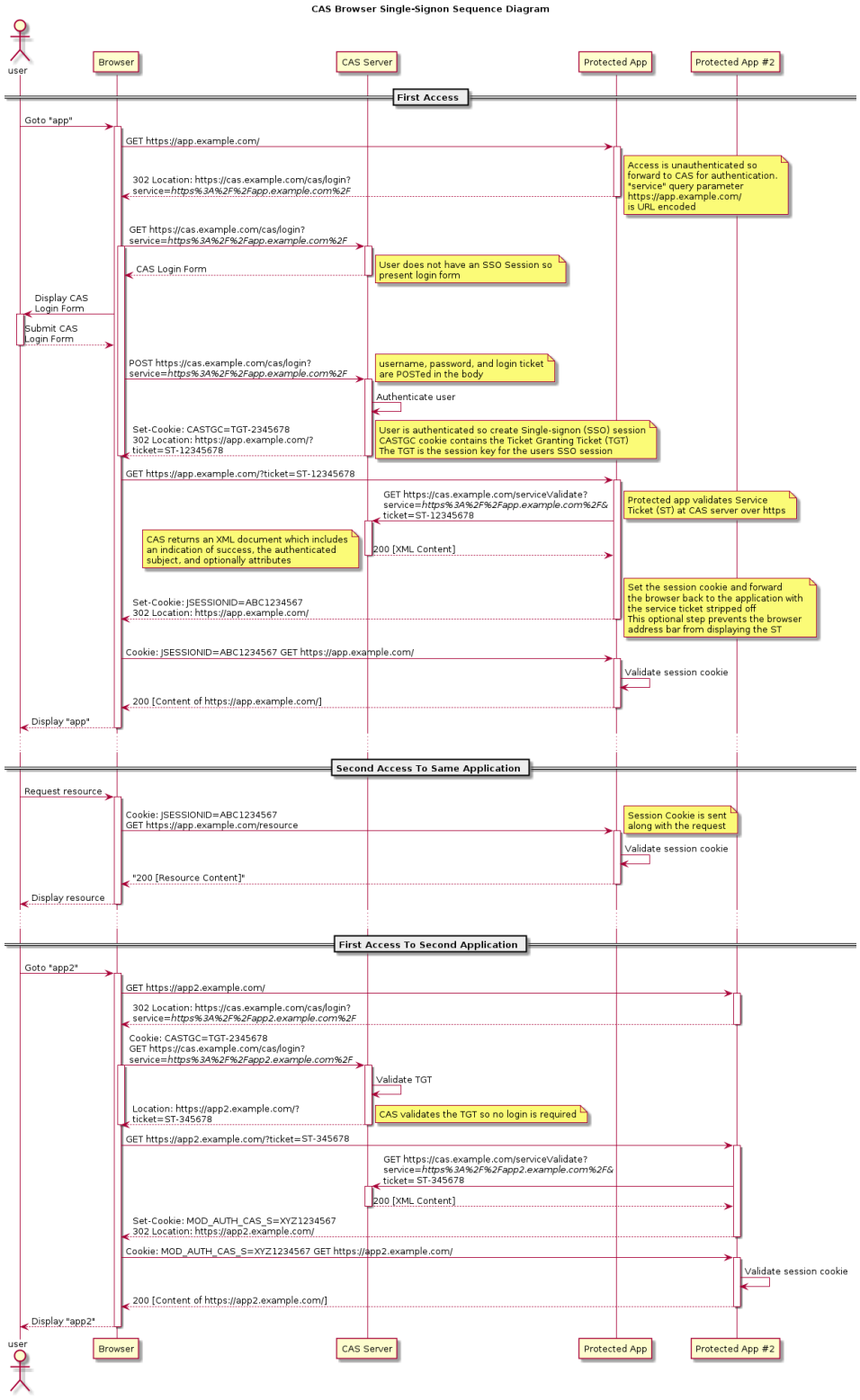
2.2.2 Ροή Εργασιών Πρωτοκόλλου CAS

Το πρωτόκολλο CAS ακολουθεί μια συγκεκριμένη ροή εργασιών με σκοπό να αυθεντικοποιήσει χρήστες και να εγκαθιδρύσει SSO συνεδρίες. Τα βήματα τα οποία ακολουθούνται στη ροή εργασιών του CAS, οργανώνονται με τον εξής τρόπο.

Αρχικά, υπάρχει η φάση της *Αρχικής Πρόσβασης*, όπου ο χρήστης επιχειρεί να αποκτήσει πρόσβαση σε έναν προστατευμένο πόρο ή υπηρεσία που παρέχεται από έναν πελάτη CAS. Δεδομένου ότι ο χρήστης δεν έχει αυθεντικοποιηθεί, ο πελάτης CAS ανακατευθύνει το χρήστη στο διακομιστή CAS, για έλεγχο ταυτότητας. Δεύτερη φάση είναι αυτή του *Αιτήματος Ελέγχου Ταυτότητας*. Συγκεκριμένα, ο διακομιστής CAS παρουσιάζει μια διεπαφή σύνδεσης στο χρήστη, επιτρέποντάς του να εισαγάγει τα διαπιστευτήριά του, όπως το όνομα χρήστη και τον κωδικό πρόσβασής του.

Στη συνέχεια ακολουθεί η *Επικύρωση Διαπιστευτηρίων*. Ο διακομιστής CAS επαληθεύει τα διαπιστευτήρια του χρήστη με τη βοήθεια ενός αποθηκευτικού χώρου χρήστη ή ενός εξωτερικού συστήματος ελέγχου ταυτότητας. Στην περίπτωση που τα διαπιστευτήρια κριθούν έγκυρα, ο διακομιστής CAS δημιουργεί ένα μοναδικό αναγνωριστικό συνεδρίας και το συνεχίζει με την αυθεντικοποιημένη συνεδρία του χρήστη. Μετά την επιτυχημένη αυθεντικοποίηση, ο διακομιστής CAS δημιουργεί ένα Εισιτήριο Υπηρεσίας (Service Ticket), το οποίο είναι μια μοναδική αναγνωριστική συμβολοσειρά, η οποία αντιπροσωπεύει την κατάσταση του ελέγχου ταυτότητας του χρήστη. Το Εισιτήριο Υπηρεσίας συσχετίζεται με την υπηρεσία που έχει ζητηθεί από το χρήστη και αποστέλλεται πίσω στον πελάτη CAS.

Στη συνέχεια ακολουθεί η *Επικύρωση Διαπιστευτηρίων*. Ο διακομιστής CAS επαληθεύει τα διαπιστευτήρια του χρήστη με τη βοήθεια ενός αποθηκευτικού χώρου χρήστη ή ενός εξωτερικού συστήματος ελέγχου ταυτότητας. Στην περίπτωση που τα διαπιστευτήρια κριθούν έγκυρα, ο διακομιστής CAS δημιουργεί ένα μοναδικό αναγνωριστικό συνεδρίας και το συνεχίζει με την αυθεντικοποιημένη συνεδρία του χρήστη. Μετά την επιτυχημένη αυθεντικοποίηση, ο διακομιστής CAS δημιουργεί ένα Εισιτήριο Υπηρεσίας (Service Ticket), το οποίο είναι μια μοναδική αναγνωριστική συμβολοσειρά, η οποία αντιπροσωπεύει την κατάσταση του ελέγχου ταυτότητας του χρήστη. Το Εισιτήριο Υπηρεσίας συσχετίζεται με την υπηρεσία που έχει ζητηθεί από το χρήστη και αποστέλλεται πίσω στον πελάτη CAS.



Εικόνα 2.1 – CAS Webflow Diagram

Ακολουθεί η *Επικύρωση του Εισιτηρίου Υπηρεσίας*. Ο πελάτης CAS λαμβάνει το εισιτήριο υπηρεσίας από το διακομιστή CAS και επικυρώνει την αυθεντικότητά του επικοινωνώντας με το διακομιστή CAS. Αυτή η επικύρωση εξασφαλίζει ότι το Εισιτήριο Υπηρεσίας έχει εκδοθεί από το διακομιστή CAS για τη συγκεκριμένη υπηρεσία

Τέλος, μόλις επικυρωθεί το Εισιτήριο Υπηρεσίας, ο πελάτης CAS θεωρεί ότι ο χρήστης έχει πιστοποιηθεί και παραχωρεί πρόσβαση στον αρχικά προστατευμένο πόρο ή υπηρεσία που είχε ζητηθεί. . Η εφαρμογή δημιουργεί μια περίοδο λειτουργίας για το χρήστη με βάση τα ληφθέντα χαρακτηριστικά και παραχωρεί, τελικώς, πρόσβαση στον πόρο που ζητήθηκε. Τα μεταγενέστερα αιτήματα που δύναται να υπάρξουν από τον πελάτη CAS προς άλλες υπηρεσίες, μπορούν να επαναχρησιμοποιήσουν την υπάρχουσα συνεδρία χωρίς να απαιτείται από το χρήστη να πραγματοποιήσει εκ νέου έλεγχο ταυτότητας.

Δηλαδή, εάν ο χρήστης επιχειρήσει να αποκτήσει πρόσβαση σε άλλο πόρο εντός της ίδιας εφαρμογής, το πρόγραμμα περιήγησης περιλαμβάνει το cookie περιόδου λειτουργίας που δημιουργήθηκε κατά την εκκίνηση της συνεδρίας (session) και η εφαρμογή εκχωρεί πρόσβαση, χωρίς να απαιτείται εκ νέου αυθεντικοποίηση στο CAS.

Εάν ο χρήστης προσπαθήσει να αποκτήσει πρόσβαση σε διαφορετική προστατευμένη εφαρμογή, η διαδικασία αρχίζει ξανά από την αρχή. Ωστόσο, επειδή ο χρήστης έχει ήδη ελέγξει την ταυτότητά του με το CAS, δε του ζητείται να συνδεθεί ξανά. Αντίθετα, ο διακομιστής CAS επικυρώνει το Δελτίο Υπηρεσίας (Service Ticket – ST), το οποίο σχετίζεται με τη δεύτερη εφαρμογή και επιστρέφει τα χαρακτηριστικά του χρήστη, επιτρέποντας στη δεύτερη εφαρμογή να εκχωρεί πρόσβαση χωρίς να απαιτείται ξεχωριστή αυθεντικοποίηση.

Γενικά, ένα εισιτήριο υπηρεσίας είναι μια αδιαφανής συμβολοσειρά που χρησιμοποιείται από τον πελάτη ως διαπιστευτήριο για να αποκτήσει πρόσβαση σε μια υπηρεσία. Το εισιτήριο υπηρεσίας λαμβάνεται από το CAS με την παρουσίαση των διαπιστευτηρίων και ενός αναγνωριστικού υπηρεσίας από τον πελάτη στο /login.

Τα εισιτήρια υπηρεσίας ισχύουν μόνο για το αναγνωριστικό υπηρεσίας που καθορίστηκε στο /login κατά τη δημιουργία τους. Το αναγνωριστικό υπηρεσίας δεν πρέπει να αποτελεί μέρος του δελτίου υπηρεσίας. Τα εισιτήρια υπηρεσίας πρέπει να ισχύουν μόνο για μία προσπάθεια επικύρωσης εισιτηρίου. Είτε η επικύρωση ήταν επιτυχής είτε όχι, το CAS πρέπει στη συνέχεια να

ακυρώσει το δελτίο, προκαλώντας αποτυχία όλων των μελλοντικών προσπαθειών επικύρωσης του ίδιου εισιτηρίου.

Τα μη επικυρωμένα εισιτήρια υπηρεσίας CAS πρέπει να λήγουν σε εύλογο χρονικό διάστημα μετά την έκδοσή τους. Εάν μια υπηρεσία παρουσιάσει ένα ληγμένο δελτίο υπηρεσίας για επικύρωση, το CAS πρέπει να απαντήσει με μια απάντηση αποτυχίας επικύρωσης. Συνιστάται η απάντηση επικύρωσης να περιλαμβάνει ένα περιγραφικό μήνυμα που να εξηγεί γιατί απέτυχε η επικύρωση.

Συνιστάται η διάρκεια ισχύος ενός εισιτηρίου υπηρεσίας πριν τη λήξη του να μην υπερβαίνει τα πέντε λεπτά. Η τοπική ασφάλεια και η χρήση CAS μπορεί να καθορίσουν τη βέλτιστη διάρκεια ζωής των μη επικυρωμένων εισιτηρίων υπηρεσίας. Τα εισιτήρια εξυπηρέτησης πρέπει να περιέχουν επαρκή ασφαλή τυχαία δεδομένα, έτσι ώστε ένα εισιτήριο να μην είναι προβλέψιμο.

Τέλος, τα εισιτήρια υπηρεσίας πρέπει να ξεκινούν με τους χαρακτήρες, ST-. Οι υπηρεσίες πρέπει να μπορούν να δέχονται εισιτήρια υπηρεσίας μήκους έως 32 χαρακτήρων. Συνιστάται οι υπηρεσίες να υποστηρίζουν εισιτήρια υπηρεσίας μήκους έως 256 χαρακτήρων.

2.2.3 Λειτουργίες Πρωτοκόλλου CAS

Το πρωτόκολλο CAS παρέχει πολλές και αξιοσημείωτες λειτουργίες, οι οποίες ενισχύουν την ασφάλεια, τη χρηστικότητα και την ευελιξία του. Κάποιες από τις λειτουργίες είναι οι εξής παρακάτω:

- *Single Sign-On (SSO):* Το CAS επιτρέπει στους χρήστες να αυθεντικοποιούνται μόνο μια φορά και να έχουν πρόσβαση σε πολλαπλές υπηρεσίες χωρίς να υπάρχει ανάγκη για επαναλαμβανόμενο έλεγχο ταυτότητας.
- *Αυθεντικοποίηση Διακομιστή Μεσολάβησης (Proxy Authentication):* Το CAS υποστηρίζει αυθεντικοποίηση διακομιστή μεσολάβησης επιτρέποντας σε έναν αυθεντικοποιημένο χρήστη να έχει πρόσβαση σε άλλη υπηρεσία εκ μέρους άλλου χρήστη.
- *Ανταλλαγή Χαρακτηριστικών (Attribute Exchange):* Το CAS επιτρέπει την ανταλλαγή χαρακτηριστικών (Attributes) χρήστη μεταξύ του διακομιστή CAS (CAS Server) και των εφαρμογών του πελάτη, παρέχοντας πρόσθετες πληροφορίες σχετικά με το χρήστη, μεταγενέστερα της αυθεντικοποίησής του.
- *Επεκτασιμότητα (Extensibility)*: Το CAS διαθέτει τη δυνατότητα επέκτασης, με σκοπό την υποστήριξη προσαρμοσμένων μηχανισμών ελέγχου ταυτότητας και ενσωμάτωσης με

διάφορα συστήματα αυθεντικοποίησης, όπως LDAP (Lightweight Directory Access Protocol) ή παρόχους εξωτερικών ταυτοτήτων.

2.2.4 Πλεονεκτήματα χρήστης πρωτοκόλλου CAS

Γενικά, το πρωτόκολλο CAS παρουσιάζει αρκετά πλεονεκτήματα κατά την εφαρμογή του. Αρχικά, διακρίνεται για την ευκολία κατά τη χρήση του, εφόσον οι χρήστες χρειάζεται να συνδεθούν στο διακομιστή CAS μόνο μια φορά, κατά τη διάρκεια μιας συνεδρίας για να αποκτήσουν πρόσβαση σε πολλές διαδικτυακές εφαρμογές, χωρίς την ανάγκη επιπρόσθετων συνδέσεων.

Ακολούθως, το πρωτόκολλο CAS παρουσιάζει το πλεονέκτημα της σταθερότητας, αφού η σελίδα εισόδου (login page) στο διακομιστή CAS (CAS Server), δεν αλλάζει για κάθε χρήστη, αλλά παραμένει η ίδια. Ομοίως, απαιτείται λιγότερη καταβολή προσπάθειας από την πλευρά των εφαρμογών, αφού οι ίδιες δε χρειάζεται να πραγματοποιήσουν επιπλέον ενέργειες με σκοπό τη μεμονωμένη υποδομή για τη διαδικασία της αυθεντικοποίησης.

Επιπλέον, θεωρείται επαρκώς ασφαλές πρωτόκολλο, αν ληφθεί υπόψιν το γεγονός ότι οι διαδικτυακές εφαρμογές δεν έχουν πρόσβαση στα διαπιστευτήρια που παρέχει ο χρήστης κατά τη σύνδεσή του.

Ενώ ο χρόνος που αφιερώνεται στην αρχική ρύθμιση του πρωτοκόλλου CAS, μπορεί να θεωρηθεί περισσότερος από τον αναμενόμενο, είναι γεγονός, ότι η εμπειρία τελικού χρήστη που προσφέρει αποτελείται από λιγότερες τριβές. Αυτό οφείλεται κυρίως στην ευκολία που παρουσιάζεται κατά τη διατήρησή του με την πάροδο του χρόνου, καθώς υπάρχει ένας κεντρικός διακομιστής που πρέπει να αντιμετωπίζεται για οποιαδήποτε αλλαγή και πρόβλημα σε ένα σύστημα, αντί διαχείρισης διαδικασιών αυθεντικοποίησης σε κάθε μεμονωμένη διαδικτυακή εφαρμογή.

2.2.5 Παράμετροι ασφαλείας πρωτοκόλλου CAS

Παρά το γεγονός ότι το πρωτόκολλο CAS θεωρείται επαρκώς ασφαλές, υπάρχουν αρκετές προτάσεις και παράμετροι ασφαλείας, οι οποίες είναι πάντα επίκαιρες και οφείλουν να λαμβάνονται υπόψιν, κατά την εφαρμογή του συγκεκριμένου πρωτοκόλλου.

Αρχικά, χρειάζεται η διαβεβαίωση ότι όλη η επικοινωνία μεταξύ του διακομιστή CAS (CAS Server), των εφαρμογών πελάτη (Client Applications) και των προγραμμάτων περιήγησης

χρηστών (User Browsers), είναι κρυπτογραφημένη με χρήση HTTPS για την προστασία από υποκλοπή και παραποίηση.

Στη συνέχεια, ζωτικής σημασίας θεωρείται η εφαρμογή ισχυρών τεχνικών διαχείρισης περιόδων σύνδεσης, όπως *Χρονικά Διαλείμματα Συνεδρίας* (Session Timeouts), *Ασφαλή Cookie Συνεδρίας* και *Προστασία από Παραβιάσεις Συνεδριών* (Session Hijacking) ή *Επιθέσεις Σταθεροποίησης* (Fixation Attacks).

Εξίσου σημαντική θεωρείται και η εφαρμογή *Μηχανισμών Καταγραφής και Παρακολούθησης* (Logging and Monitoring Mechanisms), με σκοπό την παρακολούθηση των δραστηριοτήτων των χρηστών, τον εντοπισμό διάφορων ανωμαλιών και την άμεση απόκριση σε ζητήματα που αφορούν την ασφάλεια.

Τέλος, καθοριστική θεωρείται η επιλογή βέλτιστων πρακτικών ασφαλείας για την παραμετροποίηση του διακομιστή και πελάτη CAS, συμπεριλαμβανομένης της ασφαλούς αποθήκευσης ευαίσθητων πληροφοριών, ισχυρών ελέγχων πρόσβασης και συχνών ενημερώσεων ασφαλείας.

2.2.6 Μηχανισμοί Ασφαλείας Πρωτοκόλλου CAS

Η Κεντρική Υπηρεσία Αυθεντικοποίησης (Central Authentication Service – CAS) χρησιμοποιεί ισχυρούς μηχανισμούς ασφάλειας για τη διασφάλιση της εμπιστευτικότητας, της ακεραιότητας και της διαθεσιμότητας των δεδομένων αυθεντικοποίησης, όπως ορίζεται από τους κεντρικούς πυλώνες της ασφάλειας. Οι συγκεκριμένοι μηχανισμοί είναι ζωτικής σημασίας κυρίως για την προστασία από κοινές απειλές και ευπάθειες στα κατανεμημένα συστήματα αυθεντικοποίησης.

Ένας από τους βασικούς μηχανισμούς ασφαλείας του πρωτοκόλλου CAS είναι η κρυπτογράφηση. Συγκεκριμένα, το CAS αξιοποιεί τεχνικές κρυπτογράφησης για την ασφάλεια των καναλιών επικοινωνίας μεταξύ του διακομιστή CAS, των εφαρμογών πελάτη και των προγραμμάτων περιήγησης χρηστών. Όλα τα δεδομένα που ανταλλάσσονται κατά τις συναλλαγές αυθεντικοποίησης, συμπεριλαμβανομένων των διαπιστευτηρίων χρήστη, των αναγνωριστικών αυθεντικοποίησης και των ισχυρισμών SAML, κρυπτογραφούνται χρησιμοποιώντας ισχυρούς κρυπτογραφικούς αλγόριθμους. Με την κρυπτογράφηση ευαίσθητων πληροφοριών, το CAS μετριάζει τον κίνδυνο υποκλοπής και μη εξουσιοδοτημένης πρόσβασης από κακόβουλες οντότητες.

Το CAS χρησιμοποιεί, επιπλέον, ψηφιακές υπογραφές για να διασφαλίσει τη την αυθεντικότητα και την ακεραιότητα των διακριτικών ελέγχου ταυτότητας και των ισχυρισμών SAML. Κάθε δήλωση που εκδίδεται από το διακομιστή CAS υπογράφεται ψηφιακά χρησιμοποιώντας ένα ιδιωτικό κλειδί και η υπογραφή επαληθεύεται από τον παραλήπτη χρησιμοποιώντας το αντίστοιχο δημόσιο κλειδί. Αυτή η κρυπτογραφική διαδικασία εγγυάται ότι οι ισχυρισμοί δεν έχουν παραβιαστεί ή τροποποιηθεί κατά τη μετάδοση, παρέχοντας διασφάλιση για την προέλευση και την ακεραιότητα του περιεχομένου τους.

Το πρωτόκολλο CAS, επίσης, χρησιμοποιεί ασφαλείς μηχανισμούς μετάδοσης αναγνωριστικών για να αποτρέψει την παραβίαση και τις επιθέσεις επανάληψης. Τα αναγνωριστικά αυθεντικοποίησης, όπως Εισιτήρια Υπηρεσίας (Service Tickets) και Εισιτήρια Μεσολάβησης (Proxy Tickets), δημιουργούνται με μοναδικά αναγνωριστικά και χρονικές σημάνσεις λήξης (expiration timestamps). Επιπλέον, τα διακριτικά μεταδίδονται μέσω ασφαλών καναλιών, όπως το HTTPS, για προστασία από υποκλοπή και μη εξουσιοδοτημένη χρήση. Επιβάλλοντας αυστηρούς κανόνες επικύρωσης αναγνωριστικών, το CAS εξασφαλίζει ότι μόνο έγκυρα, και αναγνωριστικά χωρίς λήξη γίνονται δεκτά για έλεγχο ταυτότητας και πρόσβαση.

Όσον αφορά τη διαχείριση συνεδρίας, το CAS εφαρμόζει ισχυρές τεχνικές διαχείρισης συνεδριών για τον έλεγχο των συνεδριών χρήστη και την αποτροπή μη εξουσιοδοτημένης πρόσβασης σε προστατευμένους πόρους. Τα αναγνωριστικά περιόδου λειτουργίας (Session Identifiers) αποθηκεύονται με ασφάλεια στην πλευρά του προγράμματος-πελάτη και μεταδίδονται μέσω κρυπτογραφημένων καναλιών, με σκοπό να αποτραπεί η παραβίαση περιόδων σύνδεσης και οι επιθέσεις σταθεροποίησης. Το CAS επιβάλλει χρονικά όρια συνεδρίας, λήξη περιόδου αδράνειας και ασφαλείς πολιτικές cookie περιόδου λειτουργίας για να μετριάσει τον κίνδυνο ευπαθειών που βασίζονται σε περιόδους σύνδεσης και μη εξουσιοδοτημένης πρόσβασης σε λογαριασμούς χρηστών.

Το CAS ακόμα, ενσωματώνει μηχανισμούς ελέγχου πρόσβασης για την επιβολή λεπτομερών αδειών και πολιτικών εξουσιοδότησης. Ο έλεγχος πρόσβασης βάσει ρόλων (Role-based Access Control – RBAC), ο έλεγχος πρόσβασης βάσει χαρακτηριστικών (Access-based Access Control – ABAC) και οι λεπτομερείς κανόνες εξουσιοδότησης εφαρμόζονται για τη ρύθμιση της πρόσβασης των χρηστών σε ευαίσθητους πόρους με βάση τους ρόλους, τα χαρακτηριστικά και τα προνόμιά τους. Επιβάλλοντας ελέγχους πρόσβασης τόσο σε επίπεδο εφαρμογής όσο και σε επίπεδο

υποδομής, το CAS ελαχιστοποιεί τον κίνδυνο μη εξουσιοδοτημένης πρόσβασης και παραβιάσεων δεδομένων.

Όπως γίνεται αντιληπτό, οι μηχανισμοί ασφαλείας του CAS διαδραματίζουν κρίσιμο ρόλο στη διαφύλαξη των δεδομένων αυθεντικοποίησης, στην προστασία του απορρήτου των χρηστών και στη διατήρηση της ακεραιότητας των συναλλαγών SSO. Μέσω κρυπτογράφησης, ψηφιακών υπογραφών, ασφαλούς μετάδοσης διακριτικών, διαχείρισης περιόδου λειτουργίας και ελέγχων πρόσβασης, το CAS παρέχει ένα ασφαλές και αξιόπιστο πλαίσιο ελέγχου ταυτότητας για κατανεμημένες εφαρμογές και υπηρεσίες.

2.2.7 Περιπτώσεις Χρήσης και Εφαρμογές Πρωτοκόλλου CAS

Το πρωτόκολλο CAS βρίσκει εφαρμογές ευρείας εμβέλειας σε διάφορους τομείς και κλάδους, χρησιμεύοντας ως θεμελιώδες στοιχείο για την εφαρμογή λύσεων διαχείρισης ενιαίας σύνδεσης (SSO) και ομοσπονδιακής ταυτότητας (federated identity).

Μια αρκετά γνωστή περίπτωση χρήσης και εφαρμογής είναι το Enterprise SSO. Σε εταιρικά περιβάλλοντα, το CAS χρησιμοποιείται για τον εξορθολογισμό του ελέγχου ταυτότητας χρήστη και του ελέγχου πρόσβασης σε μια πληθώρα εσωτερικών εφαρμογών και συστημάτων. Οι εργαζόμενοι μπορούν να συνδεθούν μία φορά μέσω CAS και να αποκτήσουν απρόσκοπτη πρόσβαση σε ηλεκτρονικό ταχυδρομείο, εργαλεία συνεργασίας, συστήματα προγραμματισμού πόρων επιχείρησης (Enterprise Resource Planning - ERP) και άλλες κρίσιμες για τις επιχειρήσεις εφαρμογές χωρίς την ανάγκη πολλαπλών διαπιστευτηρίων σύνδεσης. Με τον κεντρικό έλεγχο ταυτότητας, το CAS ενισχύει την παραγωγικότητα, απλοποιεί τη διαχείριση των χρηστών και βελτιώνει τη στάση ασφαλείας εντός του οργανισμού.

Το πρωτόκολλο CAS υιοθετείται ευρέως από εκπαιδευτικά ιδρύματα, πανεπιστήμια και ερευνητικούς οργανισμούς για την παροχή ενιαίας πρόσβασης σε ακαδημαϊκούς πόρους, συστήματα διαχείρισης μάθησης (LMS), φοιτητικές πύλες και διαδικτυακές βιβλιοθήκες. Οι φοιτητές, το διδακτικό προσωπικό και το γενικό προσωπικό επωφελούνται από μια απρόσκοπτη εμπειρία σύνδεσης σε διάφορες εκπαιδευτικές πλατφόρμες, επιτρέποντας την αποτελεσματική συνεργασία, την ανταλλαγή γνώσεων και την ακαδημαϊκή δέσμευση. Το CAS διευκολύνει, επίσης, την ασφαλή ενοποίηση με υπηρεσίες τρίτων, όπως διαδικτυακές πλατφόρμες μαθημάτων

και βάσεις δεδομένων έρευνας, ενισχύοντας τη μαθησιακή εμπειρία για όλους τους ενδιαφερόμενους.

Επιπλέον, οι κυβερνητικές υπηρεσίες και οι οργανισμοί του δημόσιου τομέα αξιοποιούν το CAS για να βελτιώσουν τις υπηρεσίες των πολιτών, να ασφαλίσουν τις ηλεκτρονικές συναλλαγές και να προστατεύσουν ευαίσθητα κρατικά δεδομένα. Οι πολίτες μπορούν να αυθεντικοποιηθούν μέσω CAS για να έχουν πρόσβαση σε κυβερνητικές πύλες, ηλεκτρονικές υπηρεσίες, συστήματα φορολογικής δήλωσης και βάσεις δεδομένων δημόσιων αρχείων με ασφάλεια. Το CAS διασφαλίζει τη συμμόρφωση με ρυθμιστικές απαιτήσεις, όπως η νομοθεσία περί απορρήτου δεδομένων και τα πρότυπα διασφάλισης ταυτότητας, ενώ επιτρέπει την εύκολη και φιλική προς το χρήστη πρόσβαση σε κρατικές υπηρεσίες από οπουδήποτε και οποτεδήποτε.

Τέλος, σε περιβάλλοντα υγειονομικής περίθαλψης, το CAS διαδραματίζει ζωτικό ρόλο στην εξασφάλιση δεδομένων ασθενών, ηλεκτρονικών αρχείων υγείας και συστημάτων κλινικών πληροφοριών. Οι πάροχοι υγειονομικής περίθαλψης, οι διαχειριστές και οι ασθενείς μπορούν να ελέγχουν την ταυτότητα μέσω CAS για πρόσβαση σε ιατρικά αρχεία, πλατφόρμες τηλεϊατρικής, πύλες ασθενών και ανταλλαγές πληροφοριών υγείας. Το CAS διασφαλίζει το απόρρητο των ασθενών, την εμπιστευτικότητα των δεδομένων και τη συμμόρφωση τους με κανονισμούς, ενώ διευκολύνει την αποτελεσματική παροχή υγειονομικής περίθαλψης, το συντονισμό της φροντίδας και τις πρωτοβουλίες δέσμευσης ασθενών.

2.3 Προεπισκόπηση Προτύπου SAML

Στην ενότητα αυτή, θα γίνει μια λεπτομερής επισκόπηση του πρωτοκόλλου SAML (Security Assertion Markup Language), το οποίο θα χρησιμοποιηθεί για τη δόμηση της βιβλιοθήκης της παρούσας διπλωματικής εργασίας.

Το *Security Assertion Markup Language (SAML)* είναι ένα πρότυπο που βασίζεται σε XML για την ανταλλαγή πληροφοριών αυθεντικοποίησης και εξουσιοδότησης μεταξύ παρόχων ταυτότητας (IdP) και παρόχων υπηρεσιών (SP). Επιτρέπει την ασφαλή ενιαία σύνδεση (SSO) και τη διαχείριση ομοσπονδιακής ταυτότητας σε κατανεμημένα συστήματα.

2.3.1 Εισαγωγή στο SAML

Η γλώσσα σήμανσης διαβεβαίωσης ασφαλείας (Security Assertion Markup Language – SAML) είναι ένα πρότυπο, το οποίο βασίζεται σε XML και έχει σχεδιαστεί για ασφαλή ανταλλαγή ταυτότητας και αναγνωριστικών (attributes) σε σενάρια SSO. Το SAML επιτρέπει στους χρήστες να πραγματοποιούν έλεγχο ταυτότητας μία φορά με έναν Πάροχο Ταυτότητας και να αποκτούν πρόσβαση σε πολλούς παρόχους υπηρεσιών χωρίς να χρειάζεται να παρέχουν διαπιστευτήρια επανειλημμένα.

2.3.2 Δομικά στοιχεία SAML

Το πρότυπο SAML περιλαμβάνει τα ακόλουθα βασικά δομικά στοιχεία:

- *Πάροχος Ταυτότητας – Identity Provider (IdP):* Το IdP είναι η οντότητα που είναι υπεύθυνη για τον έλεγχο ταυτότητας των χρηστών και τη δημιουργία ισχυρισμών (assertions) SAML που περιέχουν πληροφορίες ταυτότητας και χαρακτηριστικών. Είναι συνήθως μια αξιόπιστη οντότητα με την οποία οι χρήστες έχουν μια ήδη εγκαθιδρυμένη σύνδεση.
- *Πάροχος Υπηρεσίας – Service Provider (SP):* Το SP είναι η οντότητα που παρέχει υπηρεσίες ή πόρους στους χρήστες. Βασίζεται στο IdP για την αυθεντικοποίηση του χρήστη και λαμβάνει ισχυρισμούς SAML από το IdP για να καθορίσει την ταυτότητα του χρήστη και τα δικαιώματα πρόσβασης.
- *Πρότυπο SAML:* Το πρότυπο SAML ορίζει τη μορφή και τους κανόνες για την ανταλλαγή ισχυρισμών (assertion) SAML μεταξύ του IdP και του SP. Περιλαμβάνει ροές αιτημάτων-

απόκρισης (request-response), μορφές μηνυμάτων και μηχανισμούς ασφαλείας για ασφαλή επικοινωνία.

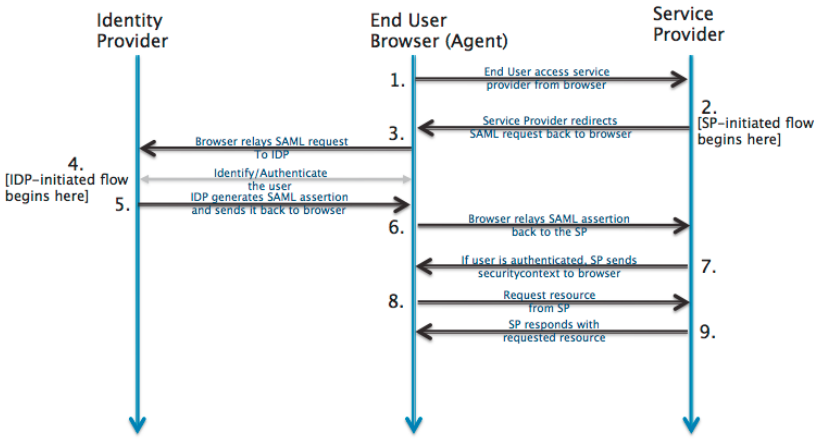
2.3.3 Ροή εργασιών προτύπου SAML

Το πρωτόκολλο SAML ακολουθεί μια συγκεκριμένη ροή εργασίας για τον έλεγχο ταυτότητας των χρηστών και τη δημιουργία εμπιστοσύνης μεταξύ του IdP και του SP. Τα τυπικά βήματα στη ροή εργασίας SAML περιγράφονται ως εξής:

Αρχικά, ο χρήστης επιχειρεί να αποκτήσει πρόσβαση σε έναν προστατευόμενο πόρο στο σύστημα του Service Provider. Ο τελευταίος δημιουργεί ένα αίτημα ελέγχου ταυτότητας SAML και ανακατευθύνει το χρήστη στο IdP μαζί με το αίτημα. Ο IdP παρουσιάζει μια διεπαφή σύνδεσης στο χρήστη, επιτρέποντάς του να εισάγει τα διαπιστευτήριά του.

Μετά τον επιτυχή έλεγχο ταυτότητας το IdP δημιουργεί έναν ισχυρισμό SAML που περιέχει πληροφορίες ταυτότητας χρήστη και χαρακτηριστικών. Ο ισχυρισμός υπογράφεται ψηφιακά από το IdP για να διασφαλιστεί η ακεραιότητα και η αυθεντικότητα. Στη συνέχεια, το IdP στέλνει τον ισχυρισμό SAML πίσω στο SP ως απόκριση στο αρχικό αίτημα ελέγχου ταυτότητας.

Ο SP επικυρώνει τον ισχυρισμό SAML επαληθεύοντας την υπογραφή και διασφαλίζοντας ότι εκδόθηκε από αξιόπιστο IdP. Τέλος, μόλις επικυρωθεί ο ισχυρισμός SAML, το SP θεωρεί ότι ο χρήστης έχει πιστοποιηθεί και παραχωρεί πρόσβαση στον πόρο ή την υπηρεσία που ζητήθηκε.



Εικόνα 2.2 – SAML Webflow Diagram

2.3.4 Προφίλ του SAML

Το SAML υποστηρίζει διαφορετικά προφίλ που ορίζουν συγκεκριμένες περιπτώσεις χρήσης και σενάρια ανάπτυξης. Μερικά προφίλ SAML που χρησιμοποιούνται συνήθως περιλαμβάνουν, αρχικά, ένα προφίλ SSO του προγράμματος περιήγησης Ιστού. Συγκεκριμένα, το προφίλ αυτό ενεργοποιεί το SSO για εφαρμογές ιστού, όπου το πρόγραμμα περιήγησης του χρήστη λειτουργεί ως παράγοντας χρήστη για την επικοινωνία μεταξύ του IdP και του SP.

Το προφίλ Single Logout επιτρέπει στους χρήστες να τερματίζουν τις περιόδους σύνδεσης SSO σε πολλά SP με ένα μόνο αίτημα αποσύνδεσης. Και τέλος, το Προφίλ Ανάκτησης Artifact. Το συγκεκριμένο προφίλ ορίζει έναν μηχανισμό για τη μεταφορά μεγάλων μηνυμάτων SAML χρησιμοποιώντας βραχύβια, αδιαφανή τεχνουργήματα που αναφέρονται στο πραγματικό μήνυμα που είναι αποθηκευμένο σε μια ασφαλή υπηρεσία ανάλυσης τεχνουργημάτων/ αναγνωριστικών.

2.3.5 Ζητήματα Ασφαλείας SAML

Η ασφάλεια είναι μια κρίσιμη πτυχή του πρωτοκόλλου SAML. Οι κύριες διαφορές του με το CAS είναι ότι το CAS χρησιμοποιεί έναν κεντρικό διακομιστή αυθεντικοποίησης, ενώ το SAML αξιοποιεί μια ομοσπονδία (Federation) ανεξάρτητων IdP και SP. Ακόμα, το SAML βασίζεται σε μηνύματα πρωτοκόλλου SAML για την επικοινωνία, ενώ το CAS χρησιμοποιεί συχνά ανακατευθύνσεις προγράμματος περιήγησης και φόρμες για ροές αυθεντικοποίησης.

Ακολούθως παρουσιάζονται τα ζητήματα ασφαλείας κατά την εφαρμογή SAML.

Αρχικά, όλες οι ανταλλαγές SAML πραγματοποιούνται μέσω ασφαλών καναλιών, όπως το HTTPS για προστασία από υποκλοπές και παραποίηση. Η εφαρμογή ψηφιακών υπογραφών σε ισχυρισμούς SAML και άλλα στοιχεία μηνυμάτων συμβάλλουν στη διασφάλιση της ακεραιότητας και της αυθεντικότητας.

Ακόμα, οι δηλώσεις SAML αποθηκεύονται με ασφάλεια στην πλευρά του SP για να αποτραπεί μη εξουσιοδοτημένη πρόσβαση και παραβίαση. Η δημιουργία μιας σχέσης εμπιστοσύνης με τον IdP επικυρώνοντας την ταυτότητά του και διασφαλίζοντας ότι πληροί τα απαιτούμενα πρότυπα ασφαλείας. Επίσης, η επαλήθευση της εγκυρότητας και της λήξης των ισχυρισμών SAML συμβάλλει στην αποτροπή επιθέσεων επανάληψης.

Στη συνέχεια, η πολιτική απελευθέρωσης χαρακτηριστικών και πιο συγκεκριμένα ο καθορισμός κατάλληλων πολιτικών για την απελευθέρωση χαρακτηριστικών, βοηθούν σημαντικά στον έλεγχο του όγκου των πληροφοριών χρήστη που μοιράζονται με τους SP. Τέλος, η ακεραιότητα και η εμπιστευτικότητα των μεταδεδομένων SAML χρειάζεται ιδιαίτερη προστασία, καθώς περιέχουν βασικές πληροφορίες σχετικά με τους IdP και τους SP.

2.3.6. Υλοποιήσεις SAML

Πολλές υλοποιήσεις SAML και βιβλιοθήκες είναι διαθέσιμες για να διευκολύνουν την ενσωμάτωση SAML σε εφαρμογές και συστήματα. Τα τέσσερα βασικότερα είναι τα εξής:

Αρχικά το OpenSAML είναι μια ευρέως χρησιμοποιούμενη βιβλιοθήκη Java ανοιχτού κώδικα για την υλοποίηση πρωτοκόλλου SAML. Το Shibboleth αποτελεί ένα πλαίσιο ομοσπονδίας ταυτότητας ανοιχτού κώδικα, το οποίο περιλαμβάνει μια υλοποίηση SAML. Αντίστοιχα, το OneLogin είναι ένας εμπορικός πάροχος SSO, ο οποίος προσφέρει μια ολοκληρωμένη εργαλειοθήκη SAML και λύσεις ενοποίησης του SAML με άλλα πρότυπα/πρωτόκολλα. Τέλος, το Microsoft Active Directory Federation Services (ADFS) αποτελεί μια δυνατότητα του Windows Server που περιλαμβάνει υποστήριξη SAML για εταιρικά σενάρια SSO.

Κατανοώντας το πρωτόκολλο SAML και τα στοιχεία του, οι οργανισμοί μπορούν να εφαρμόσουν ασφαλείς λύσεις διαχείρισης SSO και ομοσπονδιακής ταυτότητας, επιτρέποντας τον απρόσκοπτο έλεγχο ταυτότητας και την εξουσιοδότηση σε όλα τα κατανεμημένα συστήματα.

Κεφάλαιο 3: Μεθοδολογία

Στο παρόν κεφάλαιο περιγράφεται ο τρόπος, ο οποίος ακολουθήθηκε κατά την ανάπτυξη της βιβλιοθήκης, όπως τα εργαλεία ανάπτυξης και οι επιπρόσθετες τεχνολογίες που χρησιμοποιήθηκαν (εκτός του SSO που περιεγράφηκε παραπάνω κ.ο.κ), καθώς και ο σχεδιασμός του πρωτοκόλλου που χρησιμοποιήθηκε, με σκοπό να επιτευχθεί η βέλτιστη πρακτική.

3.1 Εργαλεία Ανάπτυξης

Παρακάτω παρουσιάζονται όλα τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της βιβίοθήκης, καθώς και τα περιβάλλοντα στα οποία σχεδιάστηκε και υλοποιήθηκε.

3.1.1 Apache Netbeans IDE

Για την ανάπτυξη του κώδικα της εφαρμογής χρησιμοποιήθηκε το Apache Netbeans IDE 21. Το Apache Netbeans IDE 21 είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που έχει σχεδιαστεί για να υποστηρίζει προγραμματιστές λογισμικού σε διάφορες γλώσσες προγραμματισμού και πλατφόρμες. Προσφέροντας, λοιπόν, μια φιλική προς το χρήστη διεπαφή και ένα πλούσιο σύνολο δυνατοτήτων, το NetBeans επιτρέπει την αποτελεσματική ανάπτυξη, εντοπισμό σφαλμάτων και ανάπτυξη εφαρμογών. Υποστηρίζει αρκετές γλώσσες προγραμματισμού, όπως Java καθώς και JavaScript, HTML, CSS, PHP, καλύπτοντας ένα ευρύ φάσμα αναγκών ανάπτυξης.

Ο ισχυρός επεξεργαστής κώδικα που παρέχει εργαλεία επισήμανσης σύνταξης, συμπλήρωσης κώδικα και αναδιαμόρφωσης, βελτιώνοντας την παραγωγικότητα των προγραμματιστών και την ποιότητα του κώδικα. Το NetBeans ενσωματώνεται, επίσης, με συστήματα ελέγχου εκδόσεων, όπως το Git, διευκολύνοντας τις συνεργατικές ροές εργασιών ανάπτυξης. Επιπλέον, η αρθρωτή αρχιτεκτονική του επιτρέπει την εύκολη προσαρμογή και επέκταση μέσω ενός τεράστιου οικοσυστήματος πρόσθετων επεκτάσεων.

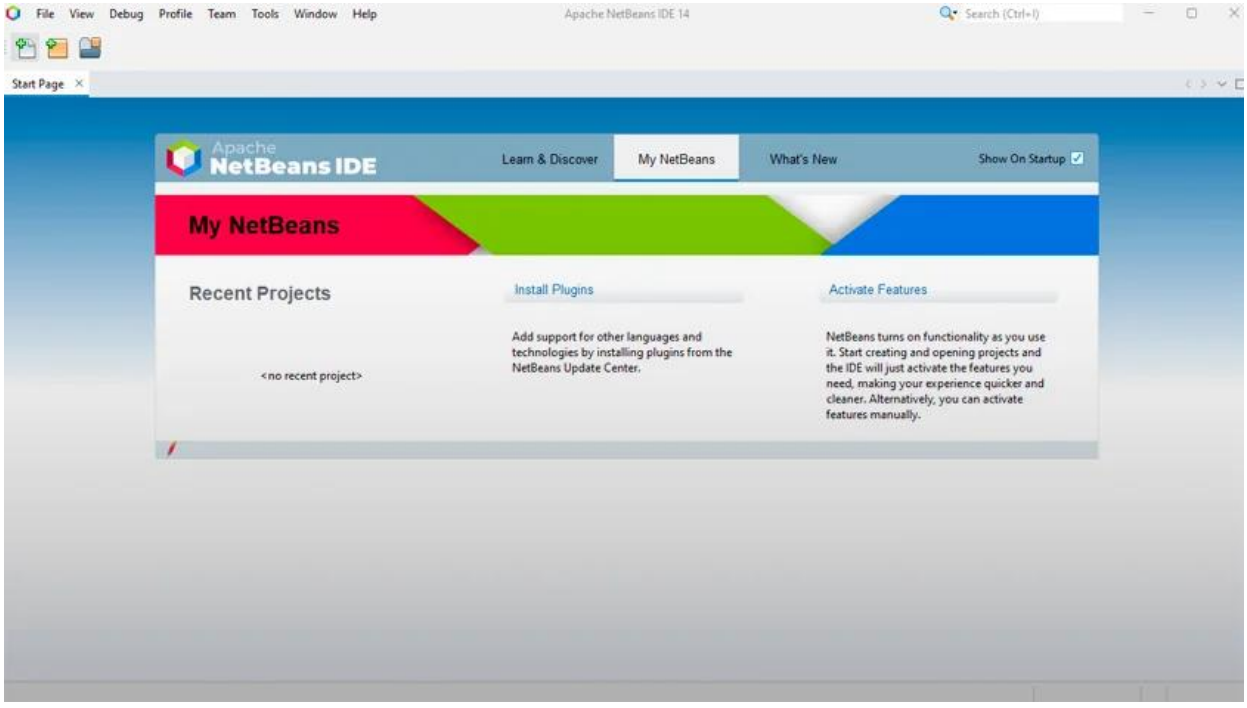
Μερικά από τα σημαντικότερα χαρακτηριστικά του Apache Netbeans IDE, τα οποία αξιοποιήθηκαν κατά τη σύνταξη και την ανάπτυξη της Java βιβλιοθήκης είναι τα εξής.

Αρχικά, προσφέρει Διαχείριση Εξάρτησης (Dependency Management). Συγκεκριμένα το NetBeans ενσωματώνεται απρόσκοπτα με δημοφιλή εργαλεία διαχείρισης εξαρτήσεων όπως το

Apache Maven, που χρησιμοποιήθηκε. Με το εργαλείο αυτό διευκολύνονται οι εξαρτήσεις βιβλιοθηκών, η επίλυση διενέξεων (conflicts) και η εισαγωγή εξωτερικών βιβλιοθηκών στα έργα. Με τον τρόπο αυτό απλοποιείται η διαδικασία ενσωμάτωσης στοιχείων τρίτων και εξασφαλίζεται ακεραιότητα στην ανάπτυξη της βιβλιοθήκης.

Ένα ακόμα αξιοσημείωτο χαρακτηριστικό είναι η υποστήριξη refactoring. Ειδικότερα το NetBeans παρέχει ισχυρές δυνατότητες αναδιαμόρφωσης που διευκολύνουν τη συντήρηση και την εξέλιξη κώδικα σε βιβλιοθήκες Java. Διευκολύνεται η αναδιαμόρφωση του κώδικα, μετονομάζοντας κλάσεις, μεθόδους και μεταβλητές με ασφάλεια, εξαγωγή διεπαφών και εκτέλεση δομικών αλλαγών. Το γεγονός αυτό εξασφαλίζει αναγνωσιμότητα κώδικα, δυνατότητα συντήρησης και τήρηση των προτύπων κωδικοποίησης καθ'όλη τη διάρκεια του κύκλου ζωής της βιβλιοθήκης.

Συνοπτικά, το Apache NetBeans IDE 21 δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν αποτελεσματικές λύσεις λογισμικού υψηλής ποιότητας, καθιστώντας το ένα πολύτιμο εργαλείο για τον τομέα ανάπτυξης λογισμικού.



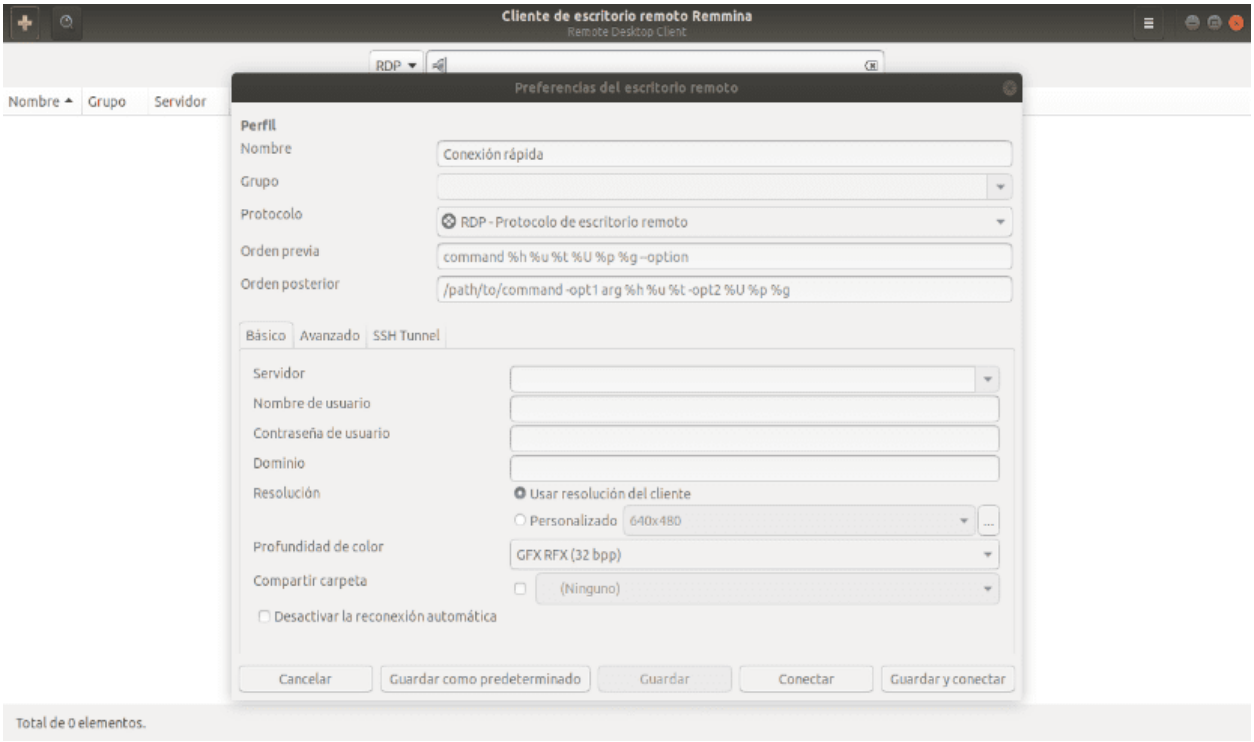
Εικόνα 3.1 – Περιβάλλον Apache Netbeans IDE

3.1.2 Remmina

Το Remmina αποτελεί μια εφαρμογή πελάτη απομακρυσμένης επιφάνειας εργασίας με πλούσια χαρακτηριστικά που παρέχει στους χρήστες ένα βολικό τρόπο πρόσβασης και διαχείρισης απομακρισμένων επιτραπέζιων υπολογιστών και εφαρμογών. Υπήρξε ιδιαίτερα χρήσιμο ως εργαλείο κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας, καθώς αποδείχθηκε ιδιαίτερα χρήσιμο στις συνδέσεις με πολλούς απομακρυσμένους διακομιστές.

Με υποστήριξη για διάφορα πρωτόκολλα απομακρυσμένης επιφάνειας εργασίας, όπως RDP, VNC, SSH και SPICE, το εργαλείο Remmina προσφέρει ευελιξία και διαλειτουργικότητα σε διαφορετικές πλατφόρμες και περιβάλλοντα. Η διαισθητική διεπαφή χρήστη, οι προσαρμοσμένες ρυθμίσεις και η διεπαφή πολλαπλών καρτελών διευκολύνουν τους χρήστες να συνδέονται σε πολλές απομακρυσμένες συνεδρίες ταυτόχρονα και να εναλλάσσονται μεταξύ τους χωρίς κόπο.

Επιπλέον το εργαλείο Remmina περιλαμβάνει προηγμένες λειτουργίες, όπως διοχέτευση σήραγγας SSH (SSH Tunneling), δυνατότητες μεταφοράς αρχείων και υποστήριξη για πολλαπλές διαμορφώσεις οθόνης, ενισχύοντας την ευελιξία και τη χρησιμότητά του για τις εργασίες απομακρυσμένης πρόσβασης.



Εικόνα 3.2 – Περιβάλλον Remmina

3.2 Τεχνολογίες που χρησιμοποιήθηκαν

Παρακάτω περιγράφονται όλες οι τεχνολογίες και τα περιβάλλοντα τα οποία χρησιμοποιήθηκαν, με σκοπό την τήρηση των βέλτιστων πρακτικών για το σχεδιασμό, καθώς και την ανάπτυξη της βιβλιοθήκης.

3.2.1 Oracle GlassFish Server

Το Oracle GlassFish Server, είναι μια πλατφόρμα διακομιστή εφαρμογών ανοιχτού κώδικα, το οποίο χρησιμοποιείται ως σκελετός ενός έργου (project), παρέχοντας ένα ισχυρό περιβάλλον χρόνου εκτέλεσης για την ανάπτυξη και την εκτέλεση εφαρμογών Java EE.

Αξιοποιώντας τις δυνατότητες και τα διάφορα χαρακτηριστικά του Glassfish γίνεται εφικτή η φιλοξενία και η διαχείριση των στοιχείων λογισμικού ενός έργου. Ως τεχνολογία, το Glassfish προσφέρει υποστήριξη για υπηρεσίες web, ανταλλαγή μηνυμάτων και επιμονή, δίνοντας τη δυνατότητα για ανάπτυξη εξελιγμένων εφαρμογών εταιρικού επιπέδου.

Η επεκτασιμότητα, η αξιοπιστία και η απόδοσή του εξασφαλίζουν την ομαλή λειτουργία ενός έργου, ενώ η τήρησή του στα πρότυπα Java EE διευκολύνει τη συμβατότητα και τη διαλειτουργικότητα με άλλες τεχνολογίες που βασίζονται σε Java.

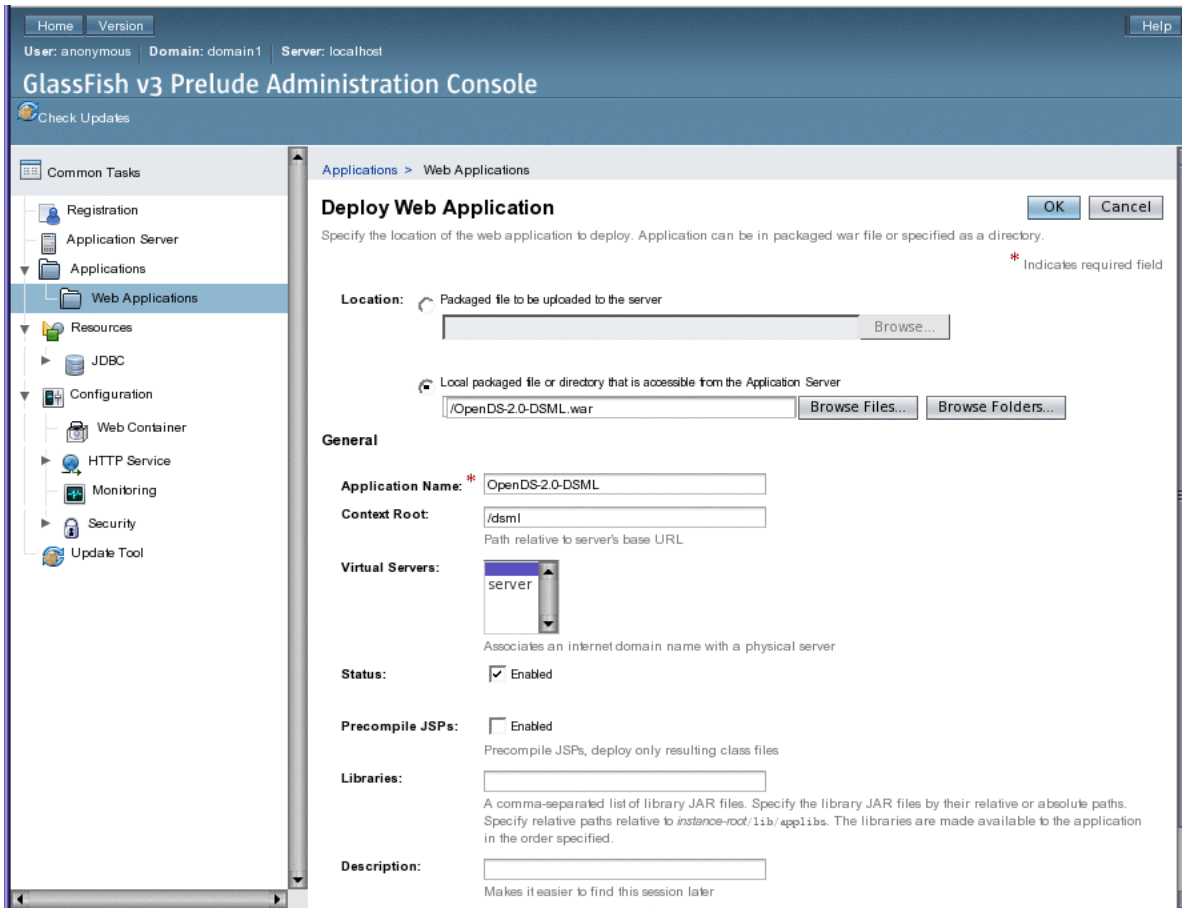
Συγκεκριμένα, προσφέρει επεκτασιμότητα, όσον αφορά τις υπηρεσίες διαδικτύου (web services). Προσφέρει, δηλαδή οριζόντια και κάθετη επεκτασιμότητα, επιτρέποντας στην υπηρεσία να χειρίζεται αποτελεσματικά την αυξανόμενη επισκεψιμότητα και τις απαιτήσεις φόρτου εργασίας. Μπορεί να κατανείμει δυναμικά πόρους και να κλιμακώνεται προς τα πάνω ή προς τα κάτω με βάση τη ζήτηση, εξασφαλίζοντας τη βέλτιστη απόδοση ακόμα και κατά τις περιόδους αιχμής χρήστης.

Το Glassfish ακόμα, υποστηρίζει μηχανισμούς ομαδοποίησης και ανακατεύθυνσης, διασφαλίζοντας υψηλή διαθεσιμότητα και ανοχή σφαλμάτων για τον εκάστοτε ιστότοπο. Διανέμοντας στοιχεία εφαρμογής σε πολλές εικόνες διακομιστών (server instances), το Glassfish ελαχιστοποιεί το χρόνο διακοπής λειτουργίας και διατηρεί την αδιάλειπτη πρόσβαση στις υπηρεσίες του εκάστοτε ιστότοπου.

Στη συνέχεια, το Glassfish παρέχει ισχυρές δυνατότητες ασφαλείας για την προστασία του εκάστοτε ιστοτόπου από απειλές και ευπάθειες. Υποστηρίζει κρυπτογράφηση SSL/TLS, έλεγχο

πρόσβασης βάσει ρόλων (Role-Based Access Control) και ενσωμάτωση με παρόχους ταυτότητας για ασφαλή έλεγχο ταυτότητας και εξουσιοδότηση. Επιπλέον, το Glassfish, προσφέρει εργαλεία για έλεγχο, παρακολούθηση και καταγραφή για τον αποτελεσματικό εντοπισμό και τον μετριασμό περιστατικών ασφαλείας.

Τέλος, ως υλοποίηση της πλατφόρμας Java EE, το Glassfish υποστηρίζει ένα ευρύ φάσμα προδιαγραφών Java EE και API. Αυτό διασφαλίζει τη συμβατότητα με εφαρμογές και διαδικτυακά πλαίσια (web applications and frameworks) που βασίζονται σε Java, επιτρέποντας την αξιοποίηση υπαρχουσών βιβλιοθηκών, στοιχείων και εργαλείων ανάπτυξης στη διαδικασία ανάπτυξης της εκάστοτε διαδικτυακής υπηρεσίας.



Εικόνα 3.3 – Περιβάλλον Oracle GlassFish Server

3.2.2 Remote Desktop Protocol (RDP)

Ως τεχνολογία που χρησιμοποιήθηκε για το έργο, το Remote Desktop Protocol (RDP) διευκολύνει την απομακρυσμένη πρόσβαση και τον έλεγχο των υπολογιστικών πόρων, επιτρέποντας την απρόσκοπτη συνεργασία, ανάπτυξη, καθώς και πληθώρα δραστηριοτήτων υποστήριξης.

Με τη χρήση RDP, δόθηκε η ικανότητα απομακρυσμένης πρόσβασης σε κοινόχρηστα περιβάλλοντα ανάπτυξης, εικονικές μηχανές και περιβάλλοντα δοκιμών, όπως στον διακομιστή του πανεπιστημίου, επιτρέποντας την εργασία από οποιοδήποτε μέρος με την ελάχιστη προϋπόθεση σύνδεσης στο διαδίκτυο.

Το RDP προωθεί την ευελιξία και την παραγωγικότητα εξαλείφοντας τα γεωγραφικά εμπόδια και επιτρέποντας την αμεσότητα σε πρόσβαση σε απομακρυσμένους πόρους. Συγκεκριμένα, το RDP παρέχει ασφαλή απομακρυσμένη πρόσβαση σε υπολογιστικούς πόρους μέσω κρυπτογραφημένων συνδέσεων, διασφαλίζοντας την εμπιστευτικότητα και την ακεραιότητα των δεδομένων κατά τις απομακρυσμένες συνεδρίες.

Επιπλέον, το RDP υποστηρίζει συμβατότητα πολλαπλών πλατφορμών, επιτρέποντας σε χρήστες να έχουν πρόσβαση σε απομακρυσμένους επιτραπέζιους υπολογιστές από διάφορες συσκευές, λειτουργικά συστήματα, συμπεριλαμβανομένων Windows, macOS και Linux.

Ακόμα, προσφέρει ισχυρές δυνατότητες διαχείρισης περιόδων σύνδεσης, επιτρέποντας στους χρήστες να αποσυνδέονται και να επανασυνδέονται σε απομακρυσμένες συνεδρίες απρόσκοπτα χωρίς να χαθεί η πρόοδος εργασίας τους.

Τέλος, το RDP υποστηρίζει ανακατεύθυνση πολυμέσων και αποτελεσματική χρήση εύρους ζώνης, επιτρέποντας στους χρήστες να μεταδίδουν περιεχόμενο ήχου, βίντεο και γραφικών μέσω απομακρυσμένων συνδέσεων με ελάχιστο λανθάνοντα χρόνο (minimal latency) και συμφόρηση δικτύου (network congestion).

Όλα αυτά τα χαρακτηριστικά καθιστούν συλλογικά το RDP μια ευέλικτη και αξιόπιστη τεχνολογία που επιτρέπει την εξ αποστάσεως πρόσβαση, ανάπτυξη και υποστήριξη στο περιβάλλον του έργου.

3.2 Σχεδιαστικά Πρότυπα

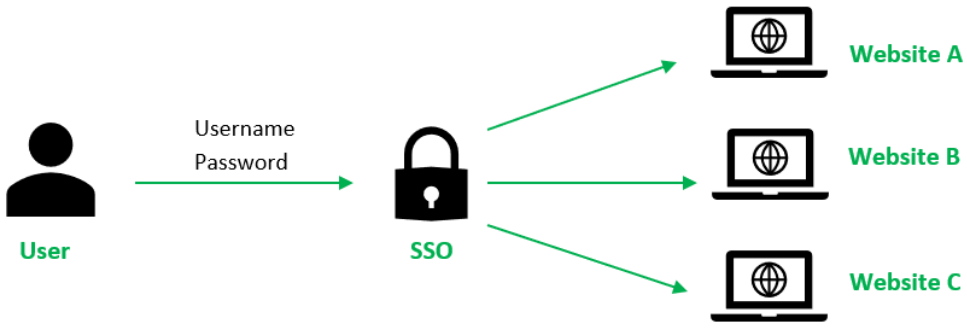
Παρακάτω παρατίθενται οι διαφορετικοί σχεδιασμοί που χρησιμοποιήθηκαν ως σημεία αναφοράς για το σχεδιασμό και κατά συνέπεια για την ανάπτυξη και υλοποίηση της βιβλιοθήκης, χρησιμοποιώντας επιμέρους ή ολόκληρα μέρη τους.

3.2.1 Η αρχιτεκτονική ενός SSO συστήματος

Όπως αναφέρθηκε στο κομμάτι της βιβλιογραφικής ανασκόπησης, το Single Sign-On (SSO) είναι ένα σύστημα ελέγχου ταυτότητας, όπου οι χρήστες μπορούν να αυθεντικοποιούνται με ασφάλεια και να αποκτούν πρόσβαση σε πολλές εφαρμογές και ιστοτόπους συνδέοντας μόνο με ένα μόνο όνομα χρήστη και κωδικό πρόσβασης. Χαρακτηριστικό παράδειγμα αποτελεί η σύνδεση στο λογαριασμό Google μια μοναδική φορά, το οποίο θα επιτρέψει την απόκτηση πρόσβασης σε διάφορες εφαρμογές Google, όπως τα Έγγραφα Google, το Gmail και το Google Drive.

Χωρίς τη λύση SSO (SSO Solution), ο ιστότοπος διατηρεί μια βάση δεδομένων με διαπιστευτήραι σύνδεσης – όνομα χρήστη και κωδικούς πρόσβασης. Κάθε φορά που ο χρήστης θα συνδεθεί στον ιστότοπο, ελέγχονται τα διαπιστευτήρια του χρήστη σε σχέση με τη βάση δεδομένων του και επαληθεύεται, με αυτό τον τρόπο, η ταυτότητα του χρήστη.

Με τη λύση SSO, ο ιστότοπος δεν αποθηκεύει διαπιστευτήρια σύνδεσης στη βάση δεδομένων του. Αντίθετα, το SSO χρησιμοποιεί ένα κοινό σύμπλεγμα διακομιστών αυθεντικοποίησης, όπου οι χρήστες απαιτείται να εισαγάγουν τα διαπιστευτήρια σύνδεσής τους μόνο μια φορά κατά την αυθεντικοποίηση. Με αυτό το χαρακτηριστικό μιας σύνδεσης και πολλαπλής πρόσβασης, είναι ζωτικής σημασίας να προστατεύονται τα διαπιστευτήρια σύνδεσης στα συστήματα SSO.



Εικόνα 3.4 – Γενική Αρχιτεκτονική Συστήματος SSO

Ως εκ τούτου, συνιστάται ιδιαίτερα η ενσωμάτωση SSO με άλλα ισχυρά μέσα αυθεντικοποίησης, όπως έξυπνα διακριτά ή κωδικούς πρόσβασης μιας χρήστης (OTP), με σκοπό να επιτευχθεί αυθεντικοποίηση πολλαπλών παραγόντων (Multi-factor Authentication).

Η αρχιτεκτονική ενός SSO συστήματος, διαχωρίζει τον πελάτη, τον Πάροχο Ταυτότητας (IdP) και τον Πάροχο Υπηρεσίας (SP) σε διαφορετικές και διακριτές οντότητες, οι οποίες έχουν τη δυνατότητα ξεχωριστής ανάπτυξης. Ο διακριτός αυτός διαχωρισμός διευκολύνει τη δομή και ανάπτυξη σε επίπεδο τμήματος, απλοποιώντας την αρχιτεκτονική, καθώς κάθε στοιχείο χαρακτηρίζεται από ένα καθαρά διακριτό σύστημα υποχρεώσεων.

Οι αρχιτεκτονικές SSO βασίζονται σε τυποποιημένα πρωτόκολλα και πλαίσια (protocols and frameworks) για επικοινωνία και διαλειτουργικότητα. Τα κοινά πρωτόκολλα περιλαμβάνουν SAML, OAuth και OpenID Connect. Τα πρωτόκολλα αυτά ορίζουν τις μορφές των μηνυμάτων, τις ροές για μηχανισμούς ασφαλείας για την ανταλλαγή πληροφοριών ελέγχου ταυτότητας και εξουσιοδότησης μεταξύ του IdP και του SP.

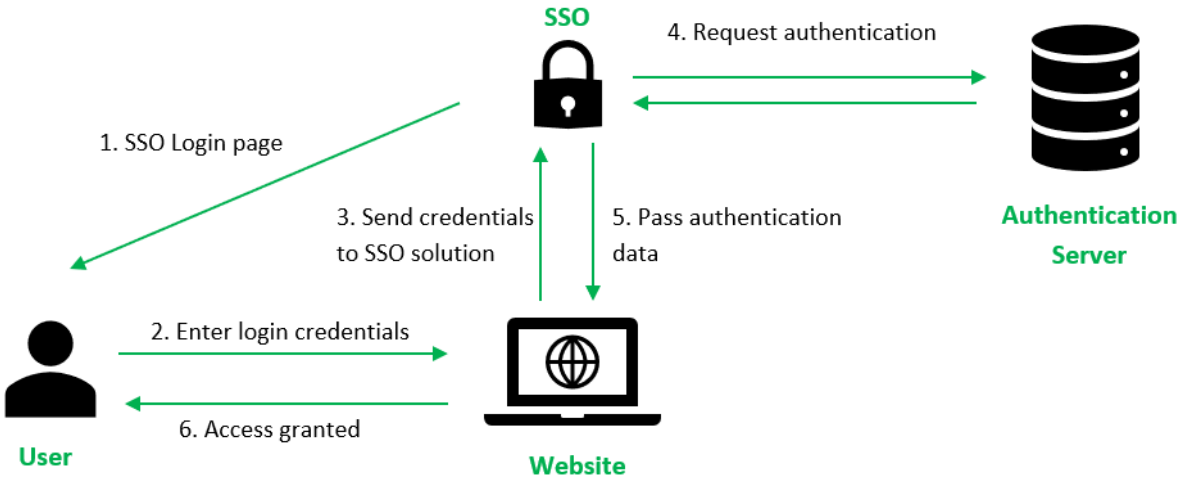
Επιπλέον, τα συστήματα SSO χρησιμοποιούν συχνά μηχανισμούς αυθεντικοποίησης που βασίζονται σε διακριτικά για να μεταφέρουν πληροφορίες ταυτότητας χρήστη και εξουσιοδότησης μεταξύ του IdP και του SP. Τα διακριτικά, όπως οι ισχυρισμοί SAML ή τα διακριτικά OAuth, ενσωματώνουν χαρακτηριστικά χρήστη, δικαιώματα και άλλες σχετικές πληροφορίες, επιτρέποντας στους SP να λαμβάνουν αποφάσεις ελέγχου πρόσβασης χωρίς να ρωτούν το IdP για κάθε αίτημα.

Ακόμα, σε πολύπλοκες αρχιτεκτονικές SSO, τα στοιχεία του ενδιάμεσου λογισμικού μπορούν να χρησιμοποιηθούν για τη διαχείριση ροής αυθεντικοποίησης, τη διαχείριση της επικύρωσης και της επεξεργασίας διακριτικών και την επιβολή πολιτικών ασφαλείας.

Αυτά τα στοιχεία ενδιάμεσου λογισμικού βοηθούν στην αφαίρεση της πολυπλοκότητας της ενοποίησης του SSO και παρέχουν μια ενοποιημένη διεπαφή για αλληλεπίδραση με πολλαπλές και ανεξάρτητες οντότητες Παρόχων Ταυτότητας (IdP) και Παρόχων Υπηρεσίας (SP).

Πιο συγκεκριμένα, η αρχιτεκτονική SSO, αποτελείται από τις σταθερές οντότητες του Παρόχου Ταυτότητας (IdP) και Παρόχου Υπηρεσίας (SP), αφήνοντας ως μεταβλητό μέρος τους χρήστες-πελάτες που συνδέονται καθώς και τις υπηρεσίες που βρίσκονται προσφέρονται μετά το πέρας της αυθεντικοποίησης.

Η γενική και απλούστερη αρχιτεκτονική ενός συστήματος SSO, αποτελείται από τα παρακάτω συγκεκριμένα διακριτά βήματα.



Εικόνα 3.5 – Απλή δομή αυθεντικοποίησης SSO

Αρχικά, η διαδικασία ξεκινάει με το χρήστη να εισάγει τα διαπιστευτήρια σύνδεσης στον ιστότοπο και κατά συνέπεια, ο ιστότοπος ελέγχει αν ο χρήστης έχει ήδη πιστοποιηθεί από τη λύση SSO (SSO Solution). Αν ναι, τότε ο χρήστης δύναται να αποκτήσει πρόσβαση στον ιστότοπο. Διαφορετικά, παρουσιάζεται στο χρήστη η λύση SSO, η οποία συνήθως αποτελεί μια φόρμα συμπλήρωσης των διαπιστευτηρίων του χρήστη, την οποία και καλείται για να συμπληρώσει για σύνδεση.

Ο χρήστης, λοιπόν, εισάγει τα διαπιστευτήριά του, τα οποία αποτελούνται από ένα όνομα χρήστη και έναν κωδικό πρόσβασης, τα οποία και αποστέλλονται στη λύση SSO. Η λύση SSO, με τη σειρά της αναζητά αυθεντικοποίηση από τον Πάροχο Ταυτότητας (IdP), όπως μια υπηρεσία καταλόγου Active Directory, για να επαληθεύσει την ταυτότητα του χρήστη. Μόλις επαληθευτεί η ταυτότητα του χρήστη, ο Πάροχος Ταυτότητας (IdP) στέλνει μια επαλήθευση στη λύση SSO.

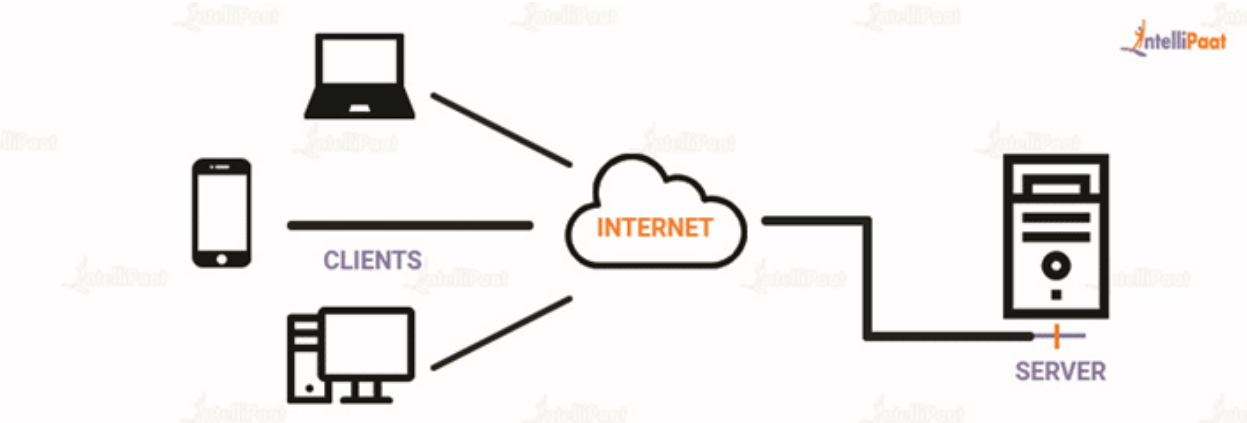
Οι πληροφορίες ελέγχου ταυτότητας μεταβιβάζονται από τη λύση SSO στον ιστότοπο όπου ο χρήστης θα έχει πρόσβαση. Μετά την επιτυχή σύνδεση με το SSO, ο ιστότοπος μεταβιβάζει δεδομένα ελέγχου ταυτότητας με τη μορφή διακριτικών ως μια μορφή επαλήθευσης, ότι ο χρήστης έχει πιστοποιηθεί και πλέον μπορεί να πλοηγείται σε διαφορετικές εφαρμογές ή ιστοσελίδα.

3.2.2 Αρχιτεκτονική Πελάτη-Διακομιστή (Client Server)

Μια από τις πιο διαδεδομένες δομές, η αρχιτεκτονική πελάτη-διακομιστή είναι ένα υπολογιστικό μοντέλο, στο οποίο ο διακομιστής φιλοξενεί, παραδίδει και διαχειρίζεται τους περισσότερους πόρους και υπηρεσίες που ζητούνται από τον πελάτη.

Είναι, επίσης, γνωστό ως μοντέλο υπολογιστών δικτύωσης ή δίκτυο πελάτη-διακομιστή, καθώς όλα τα αιτήματα και οι υπηρεσίες παραδίδονται μέσω ενός δικτύου. Η αρχιτεκτονική ή το μοντέλο πελάτη-διακομιστή έχει άλλα συστήματα συνδεδεμένα μέσω ενός δικτύου, όπου οι πόροι μοιράζονται μεταξύ των διαφορετικών υπολογιστών.

Συνήθως, η αρχιτεκτονική πελάτη-διακομιστή είναι διατεταγμένη με τέτοιο τρόπο, ώστε οι πελάτες να βρίσκονται συχνά σε σταθμούς εργασίας (workstations) ή σε προσωπικούς υπολογιστές, ενώ οι διακομιστές βρίσκονται αλλού στο δίκτυο, συνήθως σε πιο ισχυρά μηχανήματα. Ένα τέτοιο μοντέλο είναι ιδιαίτερα ωφέλιμο όταν οι πελάτες και ο διακομιστής εκτελούν εργασίες ρουτίνας. Ο διακομιστής παρέχει τις ζητούμενες υπηρεσίες, ενώ οι πελάτες είναι αυτοί που ζητούν τις εν λόγω υπηρεσίες.



Εικόνα 3.6 – Αρχιτεκτονική Μοντέλου Πελάτη – Διακομιστή

3.2.2.1 Αρχιτεκτονική Πελάτη-Διακομιστή για την τεχνολογία SSO

Το μοντέλο πελάτη-διακομιστή, λοιπόν, διαδραματίζει θεμελιώδη ρόλο στην εφαρμογή συστημάτων Single Sign-On (SSO) που χρησιμοποιούν την Κεντρική Υπηρεσία Αυθεντικοποίησης (Central Authentication Service -CAS) και το SAML. Στο μοντέλο αυτό, ο πελάτης αναφέρεται στον παράγοντα χρήστη, όπως ένα πρόγραμμα περιήγησης ιστού ή μια εφαρμογή για κινητά, ενώ ο διακομιστής αντιπροσωπεύει τον Πάροχο Ταυτότητας (IdP), ο οποίος ελέγχει την ταυτότητα των χρηστών και εκδίδει διακριτικά/αναγνωριστικά ή ισχυρισμούς ελέγχου ταυτότητας.

Το μοντέλο πελάτη-διακομιστή διευκολύνει την απρόσκοπτη ανταλλαγή πληροφοριών ελέγχου ταυτότητας και εξουσιοδότησης μεταξύ του παράγοντα χρήστη και του IdP, επιτρέποντας στους χρήστες να έχουν πρόσβαση σε πολλές εφαρμογές και υπηρεσίες με ένα μόνο σύνολο διαπιστευτηρίων.

Στο πλαίσιο του SSO που βασίζεται σε CAS, το μοντέλο πελάτη-διακομιστή περιλαμβάνει την αλληλεπίδραση μεταξύ του παράγοντα χρήστη, του διακομιστή CAS και των Παρόχων Υπηρεσιών (SP). Όταν ένας χρήστης επιχειρεί να αποκτήσει πρόσβαση σε ένα προστατευόμενο πόρο σε ένα SP, το SP ανακατευθύνει το χρήστη στο διακομιστή CAS για έλεγχο ταυτότητας. Η ανακατεύθυνση αυτή, ξεκινά τη ροή SSO, με τον παράγοντα χρήστη να ενεργεί ως πελάτης.

Ο διακομιστής CAS ζητά από το χρήστη να πραγματοποιήσει έλεγχο ταυτότητας χρησιμοποιώντας τα διαπιστευτήριά του, όπως όνομα χρήστη και κωδικό πρόσβασης. Μετά τον επιτυχή έλεγχο ταυτότητας, ο διακομιστής CAS εκδίδει ένα Δελτίο Υπηρεσίας (Service Ticket – ST) για την οντότητα του χρήστη, το οποίο στη συνέχεια παρουσιάζεται στο SP. Το SP επικυρώνει το ST με τον διακομιστή CAS για να καθορίσει την ταυτότητα του χρήστη και να παραχωρήσει πρόσβαση στο ζητούμενο πόρο.

Σε όλη αυτή τη διαδικασία, το μοντέλο πελάτη-διακομιστή εξασφαλίζει ασφαλή επικοινωνία και συντονισμό μεταξύ της οντότητας του χρήστη, του διακομιστή CAS και των SP, διευκολύνοντας την απρόσκοπτη λειτουργία της τεχνολογίας SSO σε πολλές εφαρμογές και υπηρεσίες.

Ομοίως, σε υλοποιήσεις SSO που βασίζονται σε SAML, το μοντέλο πελάτη-διακομιστή διέπει την αλληλεπίδραση μεταξύ του παράγοντα χρήστη, του Παρόχου Ταυτότητας (IdP) και των Παρόχων Υπηρεσιών (SP). Όταν ένας χρήστης επιχειρεί να αποκτήσει πρόσβαση σε έναν πόρο που

προστατεύεται από το SP, το SP ανακατευθύνει τον χρήστη στο IdP για αυθεντικοποίηση, με την οντότητα χρήστη ως πελάτη.

Το IdP ελέγχει την ταυτότητα του χρήστη και δημιουργεί έναν ισχυρισμό SAML, ο οποίος περιέχει πληροφορίες σχετικά με την ταυτότητα και τα χαρακτηριστικά του χρήστη. Αυτός ο ισχυρισμός, στη συνέχεια επιστρέφεται στην οντότητα του χρήστη, ο οποίος το παρουσιάζει στον Πάροχο Υπηρεσίας. Ο Πάροχος Υπηρεσίας επαληθεύει την αυθεντικότητα του ισχυρισμού SAML με τον Πάροχο Ταυτότητας, προσδιορίζοντας την ταυτότητα του χρήστη και παραχωρώντας πρόσβαση στο ζητούμενο πόρο.

Σε όλη αυτή τη διαδικασία, το μοντέλο πελάτη-διακομιστή διασφαλίζει για ακόμα μια φορά την ασφαλή επικοινωνία και ανταλλαγή δεδομένων μεταξύ του χρήστη, του IdP και του SP, επιτρέποντας την καλή λειτουργία της τεχνολογίας SSO σε κατανεμημένα συστήματα.

3.2.3 Έκδοση πιστοποιητικού

Η απόκτηση πιστοποιητικού για τον ενδιάμεσο διακομιστή είναι απαραίτητη για την ασφαλή λειτουργία της διαδικασίας μιας υποδομής Single Sign-On (SSO). Το πιστοποιητικό χρησιμεύει ως ψηφιακό διαπιστευτήριο, το οποίο επαληθεύει την ταυτότητα του ενδιάμεσου διακομιστή στο διακομιστή SSO.

Ο συγκεκριμένος μηχανισμός αυθεντικοποίησης δημιουργεί εμπιστοσύνη μεταξύ των δύο οντοτήτων, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι διακομιστές μπορούν να συμμετέχουν στο SSO οικοσύστημα. Με την παρουσίαση ενός έγκυρου πιστοποιητικού, ο ενδιάμεσος διακομιστής επαληθεύει τη νομιμότητα και την ακεραιότητά του, μετριάζοντας τον κίνδυνο πλαστοπροσωπίας ή μη εξουσιοδοτημένης πρόσβασης.

Επιπλέον, το πιστοποιητικό επιτρέπει τη δημιουργία ενός ασφαλούς και κρυπτογραφημένου καναλιού επικοινωνίας μεταξύ του ενδιάμεσου διακομιστή και του διακομιστή SSO. Αυτό διασφαλίζει ότι τα ευαίσθητα δεδομένα ελέγχου ταυτότητας που ανταλλάσσονται κατά τη διαδικασία SSO, όπως τα διαπιστευτήρια χρήστη και τα διακριτικά περιόδου λειτουργίας, παραμένουν εμπιστευτικά και προστατεύονται από υποκλοπή ή παραποίηση.

Επιπλέον, το πιστοποιητικό βοηθά στην προστασία από επιθέσεις man-in-the-middle, όπου ένας εισβολέας παρεμποδίζει και τροποποιεί την επικοινωνία μεταξύ των διακομιστών. Με την επικύρωση του πιστοποιητικού του ενδιάμεσου διακομιστή, ο διακομιστής SSO μπορεί να

εντοπίσει και να αποτρέψει τέτοιες επιθέσεις, διατηρώντας την ασφάλεια και την ακεραιότητα της υποδομής SSO.

3.2.3.l Διαδικασία απόκτησης πιστοποιητικού

Για να αποκτήσει το πιστοποιητικό από τον διακομιστή SSO, ο ενδιάμεσος διακομιστής ακολουθεί μια προκαθορισμένη διαδικασία που περιλαμβάνει την έναρξη μιας ασφαλούς σύνδεσης με τον διακομιστή SSO και την αίτηση του πιστοποιητικού.

Αρχικά, ο ενδιάμεσος διακομιστής εκκινεί ένα ασφαλές κανάλι επικοινωνίας με το διακομιστή SSO χρησιμοποιώντας το πρωτόκολλο HTTPS. Αυτό διασφαλίζει ότι τα δεδομένα που ανταλλάσσονται μεταξύ των δύο διακομιστών είναι κρυπτογραφημένα και προστατεύονται από υποκλοπές και παραποίηση. Ο ενδιάμεσος διακομιστής χρησιμοποιεί βιβλιοθήκες ή πλαίσια όπως το HttpsURLConnection της Java ή το Apache HttpClient για τη δημιουργία της ασφαλούς σύνδεσης.

Μόλις δημιουργηθεί η ασφαλής σύνδεση, ο ενδιάμεσος διακομιστής στέλνει ένα αίτημα πιστοποιητικού στο διακομιστή SSO. Αυτό το αίτημα περιλαμβάνει συνήθως πληροφορίες, όπως η ταυτότητα του διακομιστή και τυχόν πρόσθετες παραμέτρους που απαιτούνται από το διακομιστή SSO για την επεξεργασία του αιτήματος. Το αίτημα πιστοποιητικού αποστέλλεται μέσω του ασφαλούς καναλιού για να αποτραπεί η παρακολούθηση και η χειραγώγηση από μη εξουσιοδοτημένα μέρη.

Με τη λήψη του αιτήματος του πιστοποιητικού, ο διακομιστής SSO ελέγχει την ταυτότητα και επαληθεύει την ταυτότητα του ενδιάμεσου διακομιστή. Αυτή η διαδικασία περιλαμβάνει την επικύρωση των διαπιστευτηρίων του διακομιστή, όπως η ψηφιακή υπογραφή ή το πιστοποιητικό SSL/TLS, έναντι μιας Αξιόπιστης Αρχής (Certified Authority – CA) ή μιας αλυσίδας πιστοποιητικών. Ο διακομιστής SSO διασφαλίζει ότι ο ενδιάμεσος διακομιστής είναι εξουσιοδοτημένος να ζητήσει το πιστοποιητικό και να δημιουργήσει μια ασφαλή σύνδεση.

Μόλις ολοκληρωθεί επιτυχώς η διαδικασία αυθεντικοποίησης και επαλήθευσης, ο διακομιστής SSO εκδίδει το πιστοποιητικό που ζητήθηκε από τον ενδιάμεσο διακομιστή. Το πιστοποιητικό περιέχει το δημόσιο κλειδί του διακομιστή, την ψηφιακή υπογραφή και άλλα μεταδεδομένα που είναι απαραίτητα για τη δημιουργία ασφαλούς επικοινωνίας. Ο διακομιστής SSO υπογράφει το

πιστοποιητικό με το ιδιωτικό του κλειδί για να διασφαλίσει τη γνησιότητα και την ακεραιότητά του.

Με τη λήψη του πιστοποιητικού από το διακομιστή SSO, ο ενδιάμεσος διακομιστής εγκαθιστά και διαμορφώνει το πιστοποιητικό για χρήση με ασφαλή επικοινωνία. Το γεγονός αυτό περιλαμβάνει την αποθήκευση του πιστοποιητικού σε έναν ασφαλή χώρο αποθήκευσης κλειδιών ή αποθήκευσης αξιοπιστίας, τη διαμόρφωση των ρυθμίσεων SSL/TLS του διακομιστή, καθώς και τη δημιουργία σχέσεων εμπιστοσύνης με όλες τις οντότητες του οικοσυστήματος του SSO.

Ο ενδιάμεσος διακομιστής διασφαλίζει ότι το πιστοποιητικό έχει ρυθμιστεί σωστά ώστε να διατηρεί την εμπιστευτικότητα, την ακεραιότητα και την αυθεντικότητα της επικοινωνίας με το διακομιστή SSO.

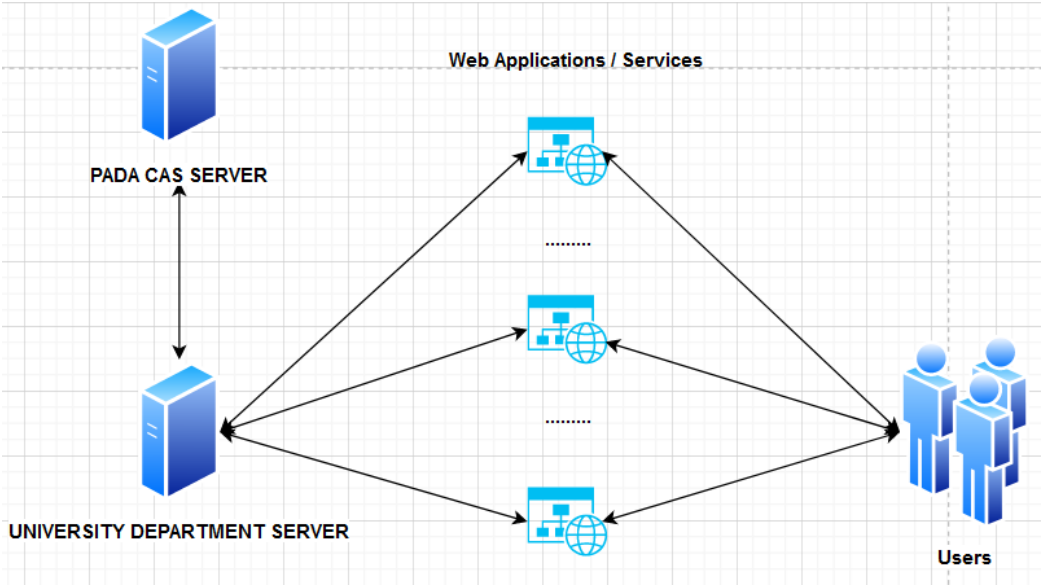
Συγκεκριμένα, χρειάζεται η εξαγωγή του πιστοποιητικού, από το διακομιστή CAS και στη συνέχεια η ένταξη και η παραμετροποίησή του σε ένα χώρο αποθήκευσης κλειδιών (keystore) ή αποθήκευσης αξιοπιστίας (truststore).

Κεφάλαιο 4: Σχεδιασμός και Ανάπτυξη

Σε αυτό το κεφάλαιο περιγράφεται κυρίως η αρχιτεκτονική της βιβλιοθήκης που σχεδιάστηκε και αναπτύχθηκε στο πλαίσιο της Διπλωματικής εργασίας. Στη συνέχεια αναλύονται κάποια επιμέρους κομμάτια σχεδιασμού και υλοποίησης, τα οποία συμβάλλουν στην ολοκλήρωση του εγχειρήματος.

4.1 Αρχιτεκτονική

Η αρχιτεκτονική του πληροφοριακού συστήματος για το οποίο αναπτύχθηκε η βιβλιοθήκη, αποτελεί υποδομή του Πανεπιστημίου Δυτικής Αττικής. Συγκεκριμένα πρόκειται για ένα πληροφοριακό σύστημα το οποίο υλοποιεί το μοντέλο αρχιτεκτονικής πελάτη – εξυπηρετητή (client – server model), χρησιμοποιώντας την τεχνολογία του Single Sign-On για την αυθεντικοποίηση χρηστών και κατά συνέπεια την εξασφάλιση πρόσβασής τους σε απομακρυσμένους, εμπιστευτικούς πόρους.



Εικόνα 4.1 – Αρχιτεκτονική του Πληροφοριακού Συστήματος του ΠΑΔΑ

Το συγκεκριμένο πληροφοριακό σύστημα, αποτελείται από τους Users, οι οποίοι μπορούν να συνδέονται σε διαδικτυακές εφαρμογές/ υπηρεσίες μέσω των διαπιστευτηρίων τους. Οι υπηρεσίες είναι ξεχωριστές η μία από την άλλη. Συνεπώς, για τη χρήση κάθε υπηρεσίας, χρειάζεται διαφορετική αυθεντικοποίηση με κάθε είσοδο, δηλαδή, ο χρήστης θα χρειαστεί να εισάγει τα διαπιστευτήριά του σε κάθε νέα υπηρεσία. Όπως φαίνεται και στην εικόνα παραπάνω, ο χρήστης

δεν είναι εφικτό να έχει άμεση επαφή με τον *Pada CAS Server*, καθώς δεν του επιτρέπεται η απευθείας σύνδεση, εφόσον ο χρήστης δεν έχει τα απαραίτητα δικαιώματα πρόσβασης.

Συγκεκριμένα, για την καλύτερη κατανόηση του πρωτοκόλλου λειτουργίας της υποδομής, παρατίθεται η εξής μελέτη περίπτωσης:

Ένας χρήστης επιχειρεί να αποκτήσει πρόσβαση στην εφαρμογή ή υπηρεσία web, *praktiki.uniwa.gr*, μέσω του προγράμματος περιήγησής του. Αρχικά συνδέεται με την IP του και όχι με κάποιο Log Name στην υπηρεσία. Με τη λήψη του αιτήματος του χρήστη, η υπηρεσία *praktiki.uniwa.gr* ανακατευθύνει το αίτημα στον *University Department Server*, ο οποίος λειτουργεί ως πύλη μεταξύ της υπηρεσίας web και του διακομιστή *PADA CAS Server* ή αλλιώς του *SSO Server*. Η ανακατεύθυνση περιλαμβάνει πληροφορίες για το όνομα της υπηρεσίας, τη διεύθυνση IP του χρήστη και δημιουργεί ένα μοναδικό αναγνωριστικό αιτήματος.

Ο ενδιάμεσος διακομιστής ελέγχει εάν ο χρήστης έχει ήδη πιστοποιηθεί. Δεδομένου ότι αυτή είναι η πρώτη προσπάθεια πρόσβασης του χρήστη, δε γίνεται έλεγχος ταυτότητας και ο ενδιάμεσος διακομιστής εκκινεί τη διαδικασία SSO. Ο ενδιάμεσος διακομιστής ανακατευθύνει το αίτημα του χρήστη, μαζί με τις σχετικές πληροφορίες (όνομα υπηρεσίας, διεύθυνση IP χρήστη, αναγνωριστικό αιτήματος), στο διακομιστή της Κεντρικής Υπηρεσίας Αυθεντικοποίησης (CAS).

Ο διακομιστής PADA CAS ζητά από το χρήστη να πραγματοποιήσει έλεγχο ταυτότητας εισάγοντας τα διαπιστευτήριά του, όπως όνομα χρήστη και κωδικό πρόσβασης. Ο χρήστης υποβάλλει τα διαπιστευτήριά του με ασφάλεια στο διακομιστή PADA CAS. Μετά την επιτυχή αυθεντικοποίησης, ο διακομιστής PADA CAS εκδίδει ένα διακριτικό ή μια δήλωση αυθεντικοποίησης που περιέχει την ταυτότητα και τα χαρακτηριστικά του χρήστη. Ο διακομιστής PADA CAS συσχετίζει αυτό το διακριτικό με την περίοδο λειτουργίας του χρήστη.

Ο διακομιστής CAS επιστρέφει το διακριτικό ή τον ισχυρισμό αυθεντικοποίησης (token) στον διακομιστή University Department, μαζί με το αναγνωριστικό αιτήματος του χρήστη. Ο University Department Server λαμβάνει το διακριτικό αυθεντικοποίησης από το διακομιστή PADA CAS. Επαληθεύει τη γνησιότητα και την εγκυρότητα του διακριτικού (token) για να διασφαλίσει ότι εκδόθηκε από αξιόπιστη πηγή και ότι δεν έχει παραβιαστεί.

Εάν το διακριτικό αυθεντικοποίησης είναι έγκυρο, ο University Department Server παραχωρεί πρόσβαση στο χρήστη για την υπηρεσία που ζητήθηκε. Συσχετίζει τη συνεδρία του χρήστη με το

αναγνωριστικό αιτήματος και αποθηκεύει σχετικές πληροφορίες αυθεντικοποίησης για μελλοντική αναφορά.

Με την επιτυχή αυθεντικοποίηση και εξουσιοδότηση του χρήστη, ο University Department Server προωθεί το αίτημα του χρήστη στην υπηρεσία, μαζί με το διακριτικό αυθεντικοποίησης και τυχόν απαραίτητες πληροφορίες χρήστη. Ο χρήστης έχει πρόσβαση στην υπηρεσία και μπορεί πλέον να επιδράσει με την εφαρμογή/υπηρεσία web, όπως έχει πιστοποιηθεί.

Τέλος, καθ'όλη τη διάρκεια της συνεδρίας του χρήστη, ο University Department Server διαχειρίζεται την κατάσταση αυθεντικοποίησης του χρήστη και τις πληροφορίες συνεδρίας, διασφαλίζοντας την ασφαλή πρόσβαση στην υπηρεσία *praktiki.uniwa.gr* και τυχόν μεταγενέστερα αιτήματα του χρήστη.

4.2 Διαχείριση Βάσης Δεδομένων Αιτημάτων και Εισιτηρίων Υπηρεσίας

Κατά την εφαρμογή της αυθεντικοποίησης Single Sign-On (SSO) για μια υπηρεσία ή εφαρμογή web, μια ειδική βάση δεδομένων λειτουργεί ως κρίσιμο στοιχείο για την αποθήκευση και τη διαχείριση πληροφοριών αιτημάτων και εισιτηρίων. Συγκεκριμένα, το παρόν κεφάλαιο περιγράφει κάποιες ενδεικτικές στρατηγικές δόμησης και διαχείρισης της βάσης δεδομένων, καθώς και τις σκέψεις για το χειρισμό δεδομένων αιτημάτων και Δελτίων Αυθεντικοποίησης στο πλαίσιο του SSO.

Το σχήμα βάσης δεδομένων έχει σχεδιαστεί για να φιλοξενεί την αποθήκευση των λεπτομερειών αιτήματος, των Εισιτηρίων Αυθεντικοποίησης και των σχετικών μεταδεδομένων. Αυτό περιλαμβάνει πίνακες για την αποθήκευση αναγνωριστικών αιτημάτων, διευθύνσεις IP χρήστη, ονόματα υπηρεσιών, διακριτικά αυθεντικοποίησης, χρόνους λήξης και άλλες σχετικές πληροφορίες. Το σχήμα βάσης δεδομένων είναι βελτιστοποιημένο για να διασφαλίζει αποτελεσματική ανάκτηση και χειρισμό δεδομένων για σκοπούς αυθεντικοποίησης.

Συγκεκριμένα, η βάση δεδομένων αποθηκεύει πληροφορίες σχετικά με τα εισερχόμενα αιτήματα χρηστών που ανακατευθύνονται από την υπηρεσία web στον ενδιάμεσο διακομιστή (University Department Server). Σε κάθε αίτημα εκχωρείται ένα μοναδικό αναγνωριστικό αιτήματος, το οποίο χρησιμεύει ως αναγνωριστικό κλειδιού για την παρακολούθηση της διαδικασίας αυθεντικοποίησης. Η βάση δεδομένων καταγράφει διευθύνσεις IP χρηστών, ονόματα υπηρεσιών,

χρονικές σημάνσεις και άλλα μεταδεδομένα αιτημάτων, με σκοπό να διευκολύνει το χειρισμό και τον έλεγχο των αιτημάτων.

Τα εισιτήρια αυθεντικοποίησης που εκδίδονται από το διακομιστή της Κεντρικής Υπηρεσίας Αυθεντικοποίησης (CAS) αποθηκεύονται με ασφάλεια στη βάση δεδομένων. Τα εισιτήρια συσχετίζονται με το αντίστοιχο αναγνωριστικό αιτήματος και περιέχουν πληροφορίες ελέγχου ταυτότητας χρήστη, όπως ονόματα χρήστη, ρόλους και χαρακτηριστικά περιόδου σύνδεσης. Οι χρόνοι λήξης των εισιτηρίων καταγράφονται για την επιβολή των πολιτικών ασφαλείας και της διάρκειας ζωής των συνεδριών.

Η βάση δεδομένων είναι ακόμα υπεύθυνη για την επικύρωση και την επαλήθευση των διακριτικών αυθεντικοποίησης που ανακτώνται από το διακομιστή CAS κατά τη διαδικασία αυθεντικοποίησης. Η επικύρωση του διακριτικού περιλαμβάνει τον έλεγχο της ακεραιότητας, της γνησιότητας και της κατάστασης λήξης του διακριτικού για να διασφαλιστεί ότι εκδόθηκε από αξιόπιστη αρχή και ότι δεν έχει παραβιαστεί. Τα ερωτήματα (queries) εκτελούνται σε σχέση με τη βάση δεδομένων για την επικύρωση του διακριτικού έναντι του αντίστοιχου αναγνωριστικού αιτήματος και της περιόδου λειτουργίας χρήστη.

Η βάση δεδομένων διατηρεί πληροφορίες συνεδρίας για να παρακολουθεί τις καταστάσεις ελέγχου ταυτότητας χρήστη και τη διάρκεια ζωής της περιόδου λειτουργίας. Τα δεδομένα περιόδου σύνδεσης, συμπεριλαμβανομένων των ταυτοτήτων χρήστη, των καταστάσεων ελέγχου ταυτότητας και των χαρακτηριστικών περιόδου λειτουργίας, αποθηκεύονται με ασφάλεια στη βάση δεδομένων. Οι λειτουργίες διαχείρισης περιόδου σύνδεσης διασφαλίζουν ότι οι πιστοποιημένοι χρήστες διατηρούν πρόσβαση στην υπηρεσία Ιστού, ενώ τηρούν τις πολιτικές ασφαλείας και τα χρονικά όρια περιόδου λειτουργίας.

Ένα ακόμα βασικό στοιχείο στο σύστημα διαχείρισης της βάσης δεδομένων για την αυθεντικοποίηση με χρήση της τεχνολογίας SSO, είναι η διαχείριση των SAML Assertions. Συγκεκριμένα, κρίνεται αναγκαίο να γίνεται καταγραφή και αποθήκευση SAML βεβαιώσεων (SAML Assertions), οι οποίες διαδραματίζουν κρίσιμο ρόλο στη διαδικασία ελέγχου ταυτότητας.

Κάθε SAML Assertion αντιπροσωπεύει μια δήλωση που εκδίδεται από τον Πάροχο Ταυτότητας (IdP) που βεβαιώνει την αυθεντικοποίηση ενός χρήστη. Μέσα στον πίνακα, κάθε ισχυρισμός προσδιορίζεται μοναδικά από ένα AssertionID, επιτρέποντας την αποτελεσματική ανάκτηση και

διαχείριση. Επιπλέον, διατηρείται μια αναφορά ξένου κλειδιού στο αντίστοιχο Δελτίο Αυθεντικοποίησης, δημιουργώντας μια σύνδεση μεταξύ του ισχυρισμού SAML και της επαληθευμένης περιόδου σύνδεσης του χρήστη.

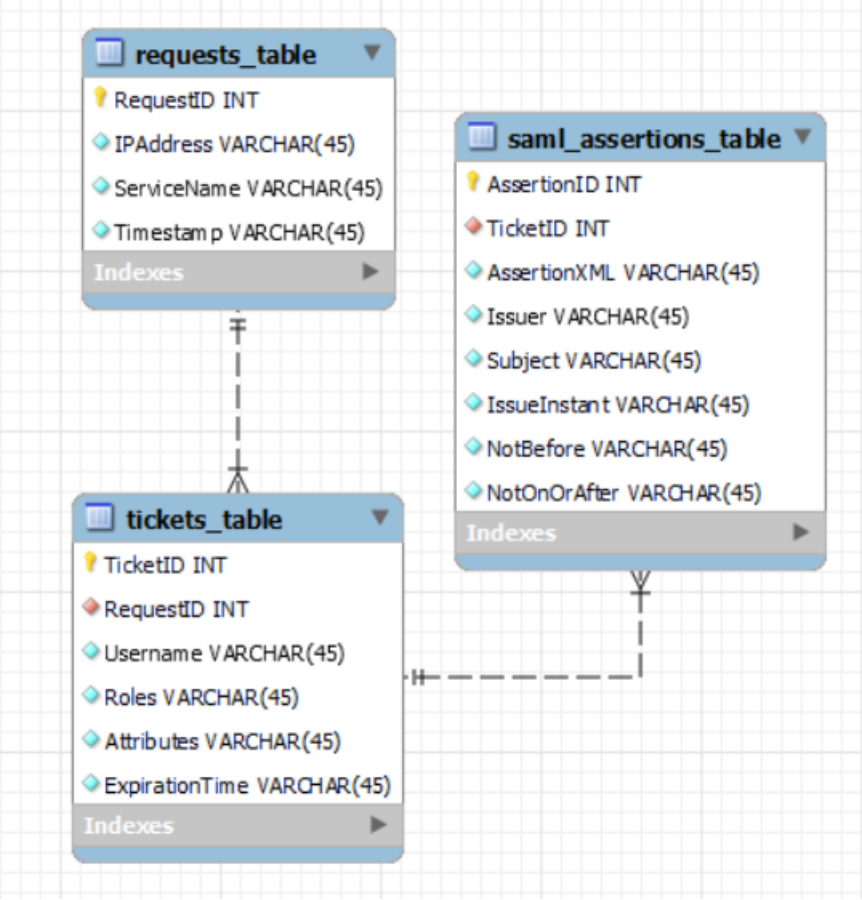
Τα βασικά χαρακτηριστικά που είναι αποθηκευμένα στον πίνακα SAML Assertions περιλαμβάνουν το AssertionXML, το οποίο περιέχει την αναπαράσταση XML του ισχυρισμού SAML. Αυτή η δομή δεδομένων ενσωματώνει ζωτικής σημασίας πληροφορίες όπως ο εκδότης, το θέμα, η στιγμή έκδοσης και η περίοδος ισχύος του ισχυρισμού. Το χαρακτηριστικό Issuer καθορίζει το EntityID του IdP που είναι υπεύθυνο για την έκδοση του ισχυρισμού, παρέχοντας πλαίσιο σχετικά με την προέλευση και την αξιοπιστία του.

Το πεδίο Θέμα υποδηλώνει την κύρια οντότητα που ελέγχεται, η οποία τυπικά αντιπροσωπεύεται από ένα όνομα χρήστη ή ένα αναγνωριστικό χρήστη. Επιπλέον, χρονικά χαρακτηριστικά όπως IssueInstant, NotBefore και NotOnOrAfter δηλώνουν τη χρονική εγκυρότητα του ισχυρισμού, οριοθετώντας την περίοδο κατά την οποία ο ισχυρισμός θεωρείται έγκυρος για σκοπούς επαλήθευσης ταυτότητας.

Διατηρώντας ένα ολοκληρωμένο αρχείο των ισχυρισμών SAML στη βάση δεδομένων, ο πίνακας SAML Assertions διευκολύνει τον ισχυρό έλεγχο ταυτότητας και τη διαχείριση περιόδων σύνδεσης εντός του οικοσυστήματος SSO. Επιτρέπει την αποτελεσματική αποθήκευση, ανάκτηση και επικύρωση δεδομένων ελέγχου ταυτότητας, δίνοντας τη δυνατότητα στο σύστημα να επιβάλλει πολιτικές ασφαλείας και να διασφαλίζει την ακεραιότητα και το απόρρητο των περιόδων σύνδεσης χρήστη. Επιπλέον, η δομημένη οργάνωση των δεδομένων βεβαιώσεων εντός του πίνακα ενισχύει τη δυνατότητα ελέγχου και τη συμμόρφωση, επιτρέποντας στους διαχειριστές να παρακολουθούν και να αναλύουν συμβάντα ελέγχου ταυτότητας για λόγους ασφάλειας και ρυθμιστικών αρχών.

Το παρακάτω Διάγραμμα Ενισχυμένης Σχέσης Οντοτήτων (Enhanced Entity-Relationship Diagram – EER) είναι μια πιο εκτεκταμένη έκδοση των διαγραμμάτων ER. Γενικά, τα μοντέλα EER είναι χρήσιμα εργαλεία για το σχεδιασμό βάσεων δεδομένων με μοντέλα υψηλού επιπέδου. Με τις βελτιωμένες λειτουργίες τους, είναι εφικτός ο σχεδιασμός βάσεων δεδομένων πιο διεξοδικά, εξετάζοντας τις ιδιότητες και τους περιορισμούς με μεγαλύτερη ακρίβεια.

Σημειώνεται ότι οι πίνακες συσχετίζονται μεταξύ τους, μέσω του ξένου κλειδιού (foreign key). Συγκεκριμένα, το RequestID αποτελεί το ξένο κλειδί για τον πίνακα 'requests_table' και το TicketID αποτελεί το ξένο κλειδί για τον πίνακα 'ticket_table'.



Εικόνα 4.2 – EER Diagram

Η δομή των πινάκων και η σημασία των στηλών του κάθε πίνακα, φαίνονται στους πίνακες παρακάτω:

requests_table	
RequestID (Primary key)	Μοναδικό αναγνωριστικό κάθε αιτήματος
IPAddress	IP διεύθυνση του αιτούντος
ServiceName	Web εφαρμογή ή υπηρεσία στην οποία ζητείται πρόσβαση

Timestamp	Χρονική σήμανση που υποδεικνύει πότε έγινε το αίτημα

Πίνακας 4.1 – Στήλες πίνακα requests_table

tickets_table	
TicketID (Primary Key)	Μοναδικό αναγνωριστικό για κάθε Δελτίο Αυθεντικοποίησης
RequestID (Foreign Key)	Αναφέρεται στο αίτημα που σχετίζεται με το εισιτήριο
Username	Όνομα χρήστη του πιστοποιημένου χρήστη
Roles	Ρόλοι που ανατίθονται στο χρήστη σχετικά με τα δικαιώματά του
Attributes	Πρόσθετα χαρακτηριστικά που σχετίζονται με την περίοδο σύνδεσης του χρήστη
ExpirationTime	Χρονική σήμανση που υποδεικνύει πότε λήγει το εισιτήριο

Πίνακας 4.2 – Στήλες πίνακα tickets_table

saml_assertions_table	
AssertionID (Primary Key)	Μοναδικό αναγνωριστικό για κάθε ισχυρισμό SAML
TicketID (Foreign Key)	Αναφέρεται στο Δελτίο Αυθεντικοποίησης που σχετίζεται με τον ισχυρισμό
AssertionXML	Αναπαράσταση του XML του ισχυρισμού SAML
Issuer	EntityID του IdP που εκδίδει τον ισχυρισμό
Subject	Αντικείμενο του ισχυρισμού (π.χ. όνομα χρήστη)
IssueInstant	Χρονική σήμανση που υποδεικνύει πότε εκδόθηκε ο ισχυρισμός
NotBefore	Χρονική σήμανση που υποδεικνύει πότε ο ισχυρισμός καθίσταται έγκυρος
NotOnOrAfter	Χρονική σήμανση που υποδεικνύει πότε λήγει ο ισχυρισμός

Πίνακας 4.3 – Στήλες πίνακα saml_assertions_table

Τέλος, εφαρμόζονται μέτρα για τη διασφάλιση της εμπιστευτικότητας, της ακεραιότητας και της ασφάλειας των δεδομένων που είναι αποθηκευμένα στη βάση δεδομένων. Αυτό περιλαμβάνει κρυπτογράφηση ευαίσθητων πληροφοριών, πολιτικές ελέγχου πρόσβασης και μηχανισμούς ελέγχου για την παρακολούθηση της δραστηριότητας της βάσης δεδομένων και τον εντοπισμό μη εξουσιοδοτημένης πρόσβασης ή τροποποιήσεων.

4.3 Βελτιστοποίηση βιβλιοθήκης

Στο προηγούμενο κεφάλαιο, έγινε υλοποίηση μιας βάσης δεδομένων SQL, στην οποία θα αποθηκεύονται όλες οι πληροφορίες που σχετίζονται με τη διαδικασία της αυθεντικοποίησης ενός χρήστη με την τεχνολογία Single Sign-On (SSO).

Όμως, γίνεται εύκολα αντιληπτό, ότι για κάθε έργο (project) χρειάζεται να δεσμευτούν διαφορετικοί πόροι, έτσι ώστε να καλύπτει τις ανάγκες των χρηστών, αλλά ταυτόχρονα, να είναι εύκολο στην υλοποίηση, καθώς και αποτελεσματικό, όσον αφορά την απόδοση.

Για το λόγο αυτό, επιλέχθηκε να γίνει η υλοποίηση της βιβλιοθήκης, με μια διαφορετικού τύπου προσέγγιση. Η προσέγγιση αυτή, περιλαμβάνει μια ανάπτυξη με τις οντότητες χρήστη, Ticket και τη δήλωση μιας προσέγγισης βάσης, το Singleton.

Γενικά, το Singleton και οι βάσεις δεδομένων, αποτελούν δύο διαφορετικά στοιχεία, εξυπηρετώντας διαφορετικούς σκοπούς και δεν είναι άμεσα συγκρίσιμες. Ωστόσο, υπάρχουν σενάρια, όπως αυτό της βιβλιοθήκης που υλοποιείται, στα οποία μπορούμε να επιλέξουμε να χρησιμοποιήσουμε ένα μοτίβο Singleton για τη διαχείριση συγκεκριμένων δεδομένων, αντί αυτά να αποθηκευτούν σε μια βάση δεδομένων SQL.

Αρχικά, η πρόσβαση σε δεδομένα που είναι αποθηκευμένα στη μνήμη (που διαχειρίζεται ένα Singleton) είναι συνήθως ταχύτερη από την ανάκτηση δεδομένων από μια βάση δεδομένων SQL, ειδικά για δεδομένα με συχνή πρόσβαση ή μεταβατικά δεδομένα. Εάν τα δεδομένα είναι μικρά, έχουν συχνά πρόσβαση και δε χρειάζεται να διατηρηθούν κατά την επαναχρησιμοποίηση της βιβλιοθήκης. Συμπεραίνεται εύκολα, λοιπόν, ότι η αποθήκευσή τους στη μνήμη με τη χρήστη ενός Singleton μπορεί να βελτιώσει την απόδοση.

Η αποθήκευση δεδομένων στη μνήμη με χρήστη Singleton μπορεί να απλοποιήσει την αρχιτεκτονική της εφαρμογής/βιβλιοθήκης και να μειώσει τα έξοδα, τα οποία σχετίζονται με συνδέσεις βάσεων δεδομένων, ερωτήματα (queries) και συναλλαγές. Για μικρά σύνολα δεδομένων ή δεδομένα που δεν απαιτούν περίπλοκες ερωτήσεις ή σχεσιακή δομή, η διαχείρισή τους με ένα Singleton μπορεί να είναι πιο αποτελεσματική και απλή.

Τα μοτίβα Singleton χρησιμοποιούνται συχνά για την προσωρινή αποθήκευση δεδομένων με συχνή πρόσβαση για τη βελτίωση της απόδοσης της εφαρμογής. Αποθηκεύοντας δεδομένα στη μνήμη, μπορεί να εφαρμοστούν μηχανισμοί προσωρινής αποθήκευσης, με σκοπό να μειωθεί η ανάγκη για επαναλαμβανόμενα ερωτήματα (queries) βάσης δεδομένων. Αυτό μπορεί να είναι ιδιαίτερα ωφέλιμο για εφαρμογές που απαιτούν ανάγνωση ή όταν υπάρχει ενασχόληση με δεδομένα που δεν αλλάζουν συχνά.

Αν τα δεδομένα είναι προσωρινά και χρειάζονται μόνο για τη διάρκεια της περιόδου λειτουργίας εφαρμογής ή του αιτήματος, η αποθήκευσή τους στη μνήμη με ένα Singleton μπορεί να είναι πιο κατάλληλη από την παραμονή τους σε μια βάση δεδομένων. Το γεγονός αυτό, μπορεί να μειώσει τα γενικά έξοδα διαχείρισης των συναλλαγών βάσης δεδομένων και των λειτουργιών καθαρισμού (cleanup operations).

Τα μοτίβα Singleton είναι χρήσιμα για τη διαχείριση κοινόχρηστων πόρων, όπως ρυθμίσεις διαμόρφωσης εφαρμογών, περιπτώσεις καταγραφής ή ομάδες νημάτων. Αυτοί οι πόροι είναι συνήθως προσπελάσιμοι συχνά σε διαφορετικά μέρη της εφαρμογής και επωφελούνται από την ύπαρξη μιας ενιαίας, παγκοσμίως προσβάσιμης παρουσίας.

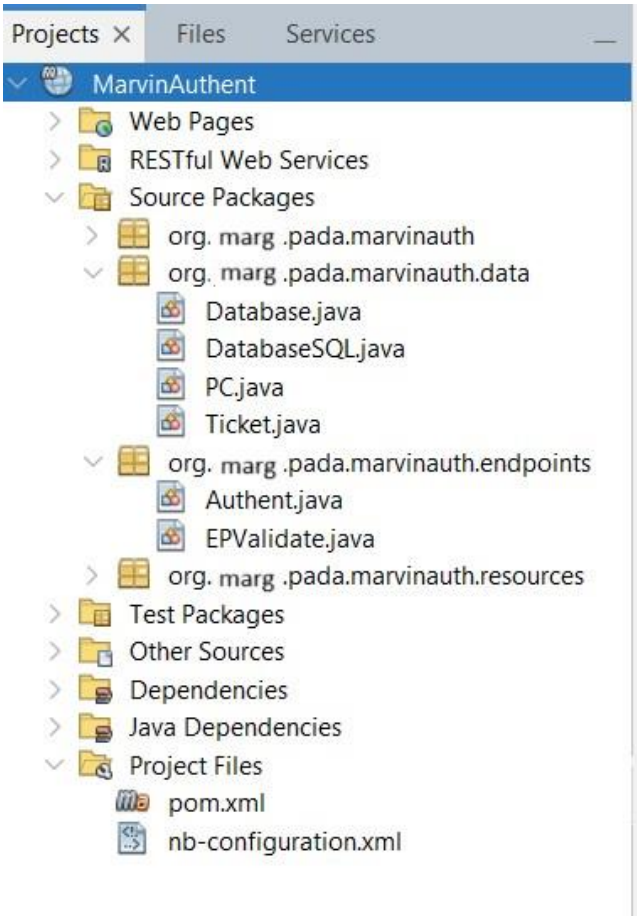
Ενώ υπάρχουν σενάρια όπου η χρήση ενός μοτίβου Singleton για τη διαχείριση δεδομένων στη μνήμη είναι κατάλληλη, είναι σημαντικό να ληφθούν υπόψη οι συμβιβασμούς και οι περιορισμοί. Τα δεδομένα που είναι αποθηκευμένα στη μνήμη είναι εύκολο να χαθούν, αν η εφαρμογή επανεκκινηθεί, ενώ τα δεδομένα που παραμένουν σε μια βάση δεδομένων SQL είναι ανθεκτικά και μπορούν να επιβιώσουν από την επανεκκίνηση της εφαρμογής.

Γενικά ως γενικό υποσύνολο, το έργο μας ασχολείται με ένα σχετικά μικρό αριθμό χρηστών σε σχέση με άλλα έργα παραγωγής τα οποία αφορούν ένα μεγαλύτερο σύνολο ατόμων/χρηστών. Συνεπώς, η χρήση των Singleton για τη διαχείριση μεγάλων ή κρίσιμων συνόλων δεδομένων μπορεί να οδηγήσει σε διόγκωση μνήμης και πιθανά προβλήματα απόδοσης. Επομένως, είναι

απαραίτητο να αξιολογηθούν προσεκτικά οι απαιτήσεις του έργου και να επιλεχθεί η κατάλληλη στρατηγική διαχείρισης δεδομένων, με βάση παράγοντες, όπως η απόδοση, η ανθεκτικότητα, η επεκτασιμότητα και η πολυπλοκότητα.

4.3.1 Υλοποίηση προσέγγισης

Η δομή του έργου που υλοποιήθηκε και περιεγράφηκε παραπάνω, είναι η παρακάτω:



Εικόνα 4.3 - Δομή Έργου Υλοποίησης

Συγκεκριμένα, η κλάση Database.java περιέχει όλη την πληροφορία για τα tickets, τους συνεργαζόμενους διακομιστές, και λειτουργεί ως οντότητα αποθήκευσης και διαχείρισης των δεδομένων που διέπουν τη διαδικασία του SSO. Αντίστοιχα, η κλάση PC.java περιγράφει τα μηχανήματα, τα οποία συνδέονται και λαμβάνουν μέρος στη διαδικασία SSO, διαχειριζόμενη στοιχεία, όπως το όνομα την ταυτότητα του μηχανήματος, την IP διεύθυνση καθώς και την υπηρεσία στην οποία θέλησε πρόσβαση. Η κλάση Ticket, αναπαριστά την οντότητα του Service

Ticket, και περιγράφει τις πληροφορίες που το συνοδεύουν, όπως την ταυτότητά του και τους χρόνους πρόσβασης, λήξης, με σκοπό να είναι εύκολος ο εντοπισμός εγκυρότητάς του. Ακολουθεί κώδικας για την κάθε κλάση στο παράρτημα.

Στην κλάση Authent.java πραγματοποιείται η διαδικασία αυθεντικοποίησης με SSO, χρησιμοποιώντας το πρωτόκολλο CAS και SAML, με όλες τις υπηρεσίες και endpoints που θα περιγράψουμε παρακάτω. Η κλάση EPValidate.java αντιστοίχως, χρησιμοποιείται ως διαχειριστής επιβεβαίωσης εγκυρότητας και η λειτουργία του γίνεται αντιληπτή κυρίως στα log files του Glassfish Server.

Τέλος, το πρόγραμμα εκτελείται ως Maven Project και το pom.xml αρχείο περιέχει όλες τις εξαρτήσεις και εξωτερικές βιβλιοθήκες που χρειάστηκαν. Συγκεκριμένα, η εκτέλεση του έργου ως έργο Maven, προσφέρει πολλά πλεονεκτήματα όσον αφορά στη διαχείριση έργου, τη διαχείριση εξαρτήσεων και την αυτοματοποίηση κατασκευής. Το Maven, είναι ένα ευρέως χρησιμοποιούμενο εργαλείο αυτοματισμού κατασκευής κυρίως για έργα Java, απλοποιεί τον κύκλο ζωής ανάπτυξης του έργου παρέχοντας μια δομημένη και τυποποιημένη προσέγγιση για τη διαχείριση εξαρτήσεων (dependencies), τη μεταγλώττιση πηγαίου κώδικα, εκτέλεση δοκιμών και τεχνουργήματα συσκευασίας (packaging artifacts).

Ο πυρήνας ενός έργου Maven βρίσκεται στο αρχείο Project Object Model (POM), το οποίο συνήθως ονομάζεται pom.xml και χρησιμεύει ως διαμόρφωση του έργου και καθορίζει τη δομή, τις εξαρτήσεις, τις προσθήκες και τις οδηγίες κατασκευής. Το αρχείο pom.xml λειτουργεί ως κεντρική πηγή μεταδεδομένων έργου, προσδιορίζοντας βασικές λεπτομέρειες όπως το αναγνωριστικό ομάδας (project’s group ID), το αναγνωριστικό τεχνουργήματος (project artifact ID), την έκδοση και τον τύπο συσκευασίας του έργο (packaging type).

Επιπλέον, το αρχείο pom.xml διευκολύνει τη διαχείριση εξαρτήσεων επιτρέποντάς τη δήλωση εξαρτήσεων έργου και την αυτόματη επίλυσή τους από το Maven Central Repository ή άλλα αποθετήρια τα οποία καθορίζονται στη διαμόρφωση (configuring build process). Η προσέγγιση της σύμβασης-υπέρ-διαμόρφωσης (convention-over-configuration) του Maven απλοποιεί τη ρύθμιση και τη διαχείριση έργου, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στη σύνταξη κώδικα αντί στη διαμόρφωση των διαδικασιών κατασκευής.

Τέλος, η αρχιτεκτονική των προσθηκών Maven παρέχει επεκτασιμότητα, επιτρέποντας την απρόσκοπτη ενσωμάτωση πρόσθετων λειτουργιών, όπως η ανάλυση κώδικα, δημιουργία τεκμηρίωσης και αυτοματοποίηση ανάπτυξης, στη διαδικασία κατασκευής του έργου.

4.3.1.1 Εξαρτήσεις και βιβλιοθήκες

Οι βασικές εξαρτήσεις που βρίσκονται στο αρχείο pom.xml, είναι οι *jakarta.platform* και *javax.servlet.*

Στο έργο Maven που παρουσιάζεται στη παρούσα διπλωματική εργασία, η χρήση της πλατφόρμας Jakarta και των βιβλιοθηκών javax.servlet προσφέρει σημαντική λειτουργικότητα και ευελιξία στην εφαρμογή μας Single Sign-On (SSO). Συγκεκριμένα, η πλατφόρμα Jakarta, παλαιότερα γνωστή ως Java EE (Enterprise Edition), παρέχει ένα ισχυρό σύνολο API και προδιαγραφών για τη δημιουργία εφαρμογών εταιρικής ποιότητας.

Η αξιοποίηση των βιβλιοθηκών Jakarta Platform στο παρόν έργο SSO, το εξοπλίζει με βασικά στοιχεία, όπως container servlet, persistence frameworks, API ασφαλείας και πολλά άλλα, διευκολύνοντας την ανάπτυξη επεκτάσιμων και ασφαλών λύσεων SSO. Επιπλέον, το javax.servlet, το οποίο αποτελεί μέρος της πλατφόρμας Jakarta, διαδραματίζει κεντρικό ρόλο στην ανάπτυξη εφαρμογών ιστού, προσφέροντας λειτουργίες servlet και HTTP.

Οι Servlets, ως θεμελιώδεις δομικοί λίθοι των εφαρμογών ιστού Java, επιτρέπουν το χειρισμό αιτημάτων και απαντήσεων HTTP, διαχείρισης συνεδριών και άλλων εργασιών που σχετίζονται με τον ιστό. Με την ενσωμάτωση βιβλιοθηκών java.servlet στο συγκεκριμένο έργο Maven, αποκτάται πρόσβαση σε τυποποιημένα API και κλάσεις servlet, απλοποιώντας την ανάπτυξη της λύσης SSO. Αυτές οι εξαρτήσεις ενδυναμώνουν το έργο με ισχυρά χαρακτηριστικά, τήρηση των βιομηχανικών προτύπων και διαλειτουργικότητα με άλλα πλαίσια και εφαρμογές που είναι συμβατά με Java EE, καθιστώντας τα απαραίτητα στοιχεία της υλοποίησης SSO.

4.3.1.2 Εκδόσεις CAS και SAML

Με βάση τις δυνατότητες του SSO Διακομιστή του τμήματος, γίνεται γνωστό, πως υποστηρίζονται οι εκδόσεις CASv1 και SAML1.1 για την επικύρωση του Δελτίου Υπηρεσίας καθώς και την απελευθέρωση των στοιχείων του χρήστη.

Συγκεκριμένα, το CASv1 (έκδοση 1 της Κεντρικής Υπηρεσίας Αυθεντικοποίησης) και το SAML1.1 (Security Assertion Markup Language έκδοση 1.1) είναι και τα δύο πρωτόκολλα αυθεντικοποίησης, τα οποία διευκολύνουν τη λειτουργία Single Sign-On σε εφαρμογές web, αν και με διαφορετικές προσεγγίσεις και δυνατότητες.

Ειδικότερα, το CASv1, το οποίο αναπτύχθηκε από το Jasig (πλέον Apareo Foundation), είναι ένα δημοφιλές πρωτόκολλο SSO ανοιχτού κώδικα που υιοθετείται ευρέως σε περιβάλλοντα τριτοβάθμιας εκπαίδευσης και επιχειρήσεων. Λειτουργεί σε ένα μοντέλο αυθεντικοποίησης, που βασίζεται σε εισιτήρια, όπου ένας κεντρικός διακομιστής CAS επαληθεύει τους χρήστεςς και εκδίδει διαπιστευτήρια εισιτηρίων που μπορούν να επικυρωθούν σε πολλές υπηρεσίες μέσα στο ίδιο οικοσύστημα CAS.

Αντίθετα, το SAML1.1 είναι ένα πρωτόκολλο βασισμένο σε πρότυπα που διέπεται από το OASIS (Οργανισμός για την Προώθηση των Δομημένων Πληροφοριακών Προτύπων), που έχει σχεδιαστεί για να επιτρέπει την ασφαλή ανταλλαγή δεδομένων αυθεντικοποίησης και εξουσιοδότησης μεταξύ Παρόχων Ταυτότητας (IdPs) και Παρόχων Υπηρεσιών (SP) σε ένα ομοσπονδιακό περιβάλλον SSO. Το SAML1.1 βασίζεται σε ισχυρισμούς που βασίζονται σε XML, με σκοπό να μεταφέρει την ταυτότητα και τα χαρακτηριστικά των χρηστών, προσφέροντας ευελιξία και διαλειτουργικότητα σε διάφορα συστήματα ελέγχου ταυτότητας.

Ενώ τόσο το CASv1 όσο και το SAML 1.1 στοχεύουν στον εξορθολογισμό του ελέγχου ταυτότητας χρήστη και της διαχείρισης πρόσβασης, το CASv1 εστιάζει στην απλότητα, την ευκολία υλοποίησης και τον κεντρικό έλεγχο ταυτότητας σε ένα οικοσύστημα CAS, ενώ το SAML 1.1 δίνει έμφαση στην τυποποίηση, τη συνένωση και τη διαλειτουργικότητα σε ετερογενή συστήματα διαχείρισης ταυτότητας. Κάθε πρωτόκολλο έχει τις δυνατότητές του και τους περιορισμούς του και η επιλογή μεταξύ CASv1 και SAML 1.1 εξαρτάται από παράγοντες όπως οι απαιτήσεις συστήματος, τα σενάρια ανάπτυξης και η συμβατότητα με την υπάρχουσα υποδομή.

Όσον αφορά τα πλεονεκτήματα, το CASv1 παρέχει μια απλή και ελαφριά λύση για την εφαρμογή SSO στο οικοσύστημα ενός οργανισμού. Το μοντέλο ελέγχου ταυτότητας που βασίζεται σε εισιτήρια απλοποιεί τη διαδικασία ελέγχου ταυτότητας, επιτρέποντας στους χρήστες να ελέγχουν ταυτότητα μία φορά με έναν κεντρικό διακομιστή CAS και να έχουν απρόσκοπτη πρόσβαση σε πολλές υπηρεσίες χωρίς την ανάγκη επαναλαμβανόμενων συνδέσεων. Το CASv1 είναι σχετικά εύκολο στην ανάπτυξη και τη διαμόρφωση, καθιστώντας το κατάλληλο για περιβάλλοντα μικρού

έως μεσαίου μεγέθους με περιορισμένους πόρους. Επιπλέον, το CASv1 προσφέρει εξαιρετική υποστήριξη για ενσωμάτωση με διάφορες πηγές ελέγχου ταυτότητας, συμπεριλαμβανομένων των παρόχων LDAP, Active Directory και προσαρμοσμένου ελέγχου ταυτότητας, επιτρέποντας στους οργανισμούς να αξιοποιήσουν τα υπάρχοντα συστήματα διαχείρισης ταυτότητας.

Παρά την απλότητα και την ευκολία ανάπτυξής του, το CASv1 έχει ορισμένους περιορισμούς σε σύγκριση με πιο σύγχρονα πρωτόκολλα SSO. Ένα μειονέκτημα του CASv1 είναι η έλλειψη εγγενούς υποστήριξης για ομοσπονδιακή διαχείριση ταυτότητας και διαλειτουργικότητα με εξωτερικά συστήματα SSO. Η προσέγγιση που βασίζεται σε εισιτήρια του CASv1 μπορεί επίσης να εισάγει προκλήσεις επεκτασιμότητας σε εγκαταστάσεις μεγάλης κλίμακας με μεγάλο όγκο συγχρονισμού και κίνησης. Επιπλέον, η εξάρτηση του CASv1 σε ένα ιδιόκτητο πρωτόκολλο και σε προσαρμοσμένους μηχανισμούς επικύρωσης εισιτηρίων ενδέχεται να εμποδίσει τη διαλειτουργικότητα και την ενοποίηση με πλαίσια και πρότυπα ελέγχου ταυτότητας τρίτων.

Αντίστοιχα, όσον αφορά τα θετικά στοιχεία του SAML1.1, προσφέρει ένα ισχυρό και τυποποιημένο πλαίσιο για την εφαρμογή ομοσπονδιακού SSO σε ετερογενή περιβάλλοντα. Η μορφή διαβεβαίωσης που βασίζεται σε XML παρέχει έναν ευέλικτο και επεκτάσιμο μηχανισμό για την ανταλλαγή δεδομένων ελέγχου ταυτότητας και εξουσιοδότησης μεταξύ παρόχων ταυτότητας (IdP) και Παρόχων Ππηρεσιών (SP). Το SAML 1.1 υποστηρίζει ένα ευρύ φάσμα σεναρίων ελέγχου ταυτότητας, συμπεριλαμβανομένων των SSO μεταξύ τομέων, μεμονωμένης αποσύνδεσης και ελέγχου πρόσβασης βάσει χαρακτηριστικών, καθιστώντας το κατάλληλο για περίπλοκες περιπτώσεις εταιρικής χρήσης. Επιπλέον, η συμμόρφωση του SAML 1.1 στα πρότυπα και τις προδιαγραφές του κλάδου διασφαλίζει τη διαλειτουργικότητα με ένα ποικίλο οικοσύστημα λύσεων διαχείρισης ταυτότητας συμβατών με SAML και πλατφορμών SSO.

4.3.1.3 CASv1 Endpoints

Τα URI τα οποία χρησιμοποιούνται για την πρώτη έκδοση του CAS (CASv1), είναι τα εξής:

URI	Περιγραφή
/login	Αιτών/Αποδέκτης Διαπιστευτηρίων
/logout	Καταστροφή της συνεδρίας CAS (αποσύνδεση)
/validate	Επικύρωση Δελτίου Υπηρεσίας (Service Ticket)

Πίνακας 4.4 – Πίνακας CASv1 URI

Κάποια παραδείγματα /login φαίνονται παρακάτω:

Απλό παράδειγμα /login:

https://cas.example.org/cas/login?service=http%3A%2F%2Fwww.example.org%2Fservice

Χωρίς προτροπή εισαγωγής διαπιστευτηρίων:

https://cas.example.org/cas/login?service=http%3A%2F%2Fwww.example.org%2Fservice&gateway=true

Συνεχής προτροπή για εισαγωγή διαπιστευτηρίων:

https://cas.example.org/cas/login?service=http%3A%2F%2Fwww.example.org%2Fservice&renew=true

Χρήση απαντήσεων POST στη θέση των ανακατευθύνσεων:

https://cas.example.org/cas/login?method=POST&service=http%3A%2F%2Fwww.example.org%2Fservice

Η παράμετρος service δεν είναι υποχρεωτική, αλλά αποτελεί γενικά το URL της εφαρμογής στην οποία ο πελάτης επιθυμεί να αποκτήσει πρόσβαση. Το CAS υποχρεούται να ανακατευθύνει τον πελάτη στο συγκεκριμένο URL, μετά από μια πετυχημένη αυθεντικοποίηση.

Μία από τις ακόλουθες απαντήσεις πρέπει να παρέχεται από το /login όταν λειτουργεί ως αποδέκτης διαπιστευτηρίων. Με επιτυχημένη σύνδεση ο πελάτης ανακατευθύνεται στη διεύθυνση URL που καθορίζεται από την παράμετρο υπηρεσίας με τρόπο που δεν θα προκαλέσει την

προώθηση των διαπιστευτηρίων του πελάτη στην υπηρεσία. Αυτή η ανακατεύθυνση πρέπει να έχει ως αποτέλεσμα ο πελάτης να εκδώσει αίτημα GET στην υπηρεσία. Το αίτημα πρέπει να περιλαμβάνει ένα έγκυρο δελτίο υπηρεσίας, που έχει περάσει ως παράμετρος αιτήματος HTTP, «εισιτήριο».

Εάν η υπηρεσία δεν είχε καθοριστεί, το CAS πρέπει να εμφανίσει ένα μήνυμα που ειδοποιεί τον πελάτη ότι έχει ξεκινήσει με επιτυχία μια μόνο περίοδο σύνδεσης.

Κατά την αποτυχία σύνδεσης γίνεται επιστροφή στο /login ως αιτών διαπιστευτηρίων. Συνιστάται σε αυτήν την περίπτωση ο διακομιστής CAS να εμφανίζει ένα μήνυμα σφάλματος στον χρήστη που περιγράφει γιατί απέτυχε η σύνδεση (π.χ. κακός κωδικός πρόσβασης, κλειδωμένος λογαριασμός κ.λπ.) και, εάν χρειάζεται, να παρέχει στον χρήστη την ευκαιρία να προσπαθήσει να συνδεθεί ξανά.

Γενικά, το /login URI λειτουργεί με δύο συμπεριφορές: ως αιτών διαπιστευτηρίων και ως αποδέκτης διαπιστευτηρίων. Απαντά στα διαπιστευτήρια ενεργώντας ως αποδέκτης διαπιστευτηρίων και διαφορετικά ενεργεί ως αιτών διαπιστευτηρίων.

Εάν ο πελάτης έχει ήδη δημιουργήσει μία μόνο περίοδο σύνδεσης με το CAS, το πρόγραμμα περιήγησης ιστού παρουσιάζει στο CAS ένα ασφαλές cookie που περιέχει μια συμβολοσειρά που προσδιορίζει ένα εισιτήριο έκδοσης εισιτηρίων. Αυτό το cookie ονομάζεται cookie έκδοσης εισιτηρίων. Εάν τα κλειδιά cookie έκδοσης εισιτηρίων σε ένα έγκυρο εισιτήριο έκδοσης εισιτηρίων, το CAS μπορεί να εκδώσει ένα εισιτήριο υπηρεσίας, υπό την προϋπόθεση ότι πληρούνται όλες οι άλλες προϋποθέσεις σε αυτήν την προδιαγραφή.

Ένα cookie έκδοσης εισιτηρίων είναι ένα cookie HTTP[5] που ορίζεται από το CAS κατά τη δημιουργία μιας ενιαίας περιόδου σύνδεσης. Αυτό το cookie διατηρεί την κατάσταση σύνδεσης για τον πελάτη και ενώ είναι έγκυρο, ο πελάτης μπορεί να το παρουσιάσει στο CAS αντί για τα κύρια διαπιστευτήρια.

Γενικά, ένα cookie έκδοσης εισιτηρίων πρέπει να οριστεί να λήγει στο τέλος της περιόδου λειτουργίας προγράμματος περιήγησης του πελάτη, εάν η μακροπρόθεσμη υποστήριξη δεν είναι ενεργή για το αντίστοιχο TGT. Το CAS πρέπει να ορίσει τη διαδρομή cookie ώστε να είναι όσο το δυνατόν πιο περιοριστική. Για παράδειγμα, εάν ο διακομιστής CAS έχει ρυθμιστεί κάτω από τη διαδρομή /cas, η διαδρομή cookie θα ρυθμιστεί σε /cas.

Η αξία των cookie έκδοσης εισιτηρίων πρέπει να περιέχει επαρκή ασφαλή τυχαία δεδομένα, έτσι ώστε ένα cookie έκδοσης εισιτηρίων να μην είναι προβλέψιμο σε εύλογο χρονικό διάστημα. Το όνομα των cookies που χορηγούν εισιτήρια πρέπει να ξεκινά με τους χαρακτήρες TGC-. Η αξία των cookies έκδοσης εισιτηρίων πρέπει να ακολουθεί τους ίδιους κανόνες με το εισιτήριο έκδοσης εισιτηρίων. Συνήθως, η αξία των cookie χορήγησης εισιτηρίων μπορεί να περιέχει το ίδιο το εισιτήριο χορήγησης εισιτηρίων ως αναπαράσταση της επαληθευμένης περιόδου σύνδεσης μεμονωμένης σύνδεσης.

Αντίστοιχα, το /logout καταστρέφει την απλή σύνδεση CAS ενός πελάτη. Το cookie παραχώρησης εισιτηρίων καταστρέφεται και τα επόμενα αιτήματα για /login δεν θα λάβουν εισιτήρια υπηρεσίας έως ότου ο χρήστης παρουσιάσει ξανά τα κύρια διαπιστευτήρια (και ως εκ τούτου δημιουργήσει μια νέα μεμονωμένη περίοδο σύνδεσης).

Εάν έχει καθοριστεί μια παράμετρος υπηρεσίας, το πρόγραμμα περιήγησης ενδέχεται να ανακατευθυνθεί αυτόματα στη διεύθυνση URL που καθορίζεται από την υπηρεσία μετά την εκτέλεση της αποσύνδεσης από το διακομιστή CAS. Εάν η ανακατεύθυνση από το διακομιστή CAS πραγματοποιηθεί όντως, εξαρτάται από τη διαμόρφωση του διακομιστή.

Ως απάντηση, το /logout πρέπει να εμφανίζει μια σελίδα που δηλώνει ότι ο χρήστης έχει αποσυνδεθεί. Εάν υλοποιηθεί η παράμετρος αιτήματος "url", το /logout πρέπει επίσης να παρέχει έναν σύνδεσμο προς την παρεχόμενη διεύθυνση URL

Κάποια παραδείγματα /login φαίνονται παρακάτω:

Απλό παράδειγμα /validate:

https://cas.example.org/cas/validate?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7

Βεβαίωση ότι το εισιτήριο υπηρεσίας εκδόθηκε με την προσκόμιση των βασικών διαπιστευτηρίων:

https://cas.example.org/cas/validate?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&renew=true

Τέλος, το /validate ελέγχει την εγκυρότητα ενός δελτίου υπηρεσίας. Το /validate είναι μέρος του πρωτοκόλλου CAS 1.0 και επομένως δεν χειρίζεται τον έλεγχο ταυτότητας διακομιστή μεσολάβησης. Το CAS πρέπει να απαντήσει με μια απόκριση αποτυχίας επικύρωσης εισιτηρίου όταν ένα δελτίο μεσολάβησης μεταβιβάζεται στο /validate.

Η παράμετρος της υπηρεσίας (service) είναι υποχρεωτική, καθώς αντιπροσωπεύει το αναγνωριστικό της υπηρεσίας για την οποία εκδόθηκε το εισιτήριο. Το εισιτήριο υπηρεσίας (ticket) που εκδίδεται από το /login είναι εξίσου υποχρεωτικό. Τέλος, η ανανέωση (renew) αν έχει οριστεί ως παράμετρος, η επικύρωση του εισιτηρίου θα είναι επιτυχής μόνο εάν το εισιτήριο υπηρεσίας εκδόθηκε από την παρουσίαση των κύριων διαπιστευτηρίων του χρήστη. Θα αποτύχει αν το εισιτήριο εκδόθηκε από μία μόνο περίοδο σύνδεσης.

Το /validate θα επιστρέψει μία από τις δύο παρακάτω απαντήσεις. Συγκεκριμένα, εάν η επικύρωση του εισιτηρίου ήταν επιτυχής, θα επιστρέψει το εξής :

yes<LF>username<LF>

Αντίθετα, στην περίπτωση αποτυχίας:

no<LF>

4.1.3.4 SAML1.1 Endpoints

Στην παρούσα διπλωματική μελετάται η σύμπραξη των πρωτοκόλλων CAS και SAML με σκοπό την εφαρμογή της τεχνολογίας SSO. Στην υλοποίηση που πραγματοποιήθηκε, υπάρχει και μελετάται ξεχωριστά ως εξάρτηση/βιβλιοθήκη η μονάδα cas-server-support-saml.

Η λειτουργική μονάδα cas-server-support-saml είναι ένα στοιχείο του λογισμικού διακομιστή CAS (Central Authentication Service) που παρέχει υποστήριξη για το πρωτόκολλο SAML (Security Assertion Markup Language). Επιτρέπει στους διακομιστές CAS να λειτουργούν τόσο ως πάροχοι ταυτότητας SAML (IdP) όσο και ως πάροχοι υπηρεσιών (SP), διευκολύνοντας τη διαλειτουργικότητα και τη συνένωση με άλλα συστήματα SSO που βασίζονται σε SAML.

Με την ενσωμάτωση της μονάδας cas-server-support-saml σε μια ανάπτυξη διακομιστή CAS, οι οργανισμοί αποκτούν τη δυνατότητα να ενσωματώνουν την υποδομή SSO που βασίζεται σε CAS με εξωτερικούς παρόχους ταυτότητας και παρόχους υπηρεσιών συμβατούς με SAML. Αυτό επιτρέπει την αυθεντικοποίηση και τη διαχείριση πρόσβασης σε ετερογενή περιβάλλοντα,

επιτρέποντας στους χρήστες να έχουν πρόσβαση σε ένα ευρύ φάσμα εφαρμογών και υπηρεσιών χρησιμοποιώντας τα διαπιστευτήριά τους CAS.

Η μονάδα cas-server-support-saml προσφέρει πολλά πλεονεκτήματα, όπως, αρχικά τη διαλειτουργικότητα. Επιτρέπει, δηλαδή, στους διακομιστές CAS να συμμετέχουν σε ομόσπονδα σενάρια SSO που βασίζονται σε SAML, επιτρέποντας την ενοποίηση με ένα ποικίλο οικοσύστημα παρόχων ταυτότητας συμβατών με SAML και παρόχων υπηρεσιών.

Στη συνέχεια παρέχει συμμόρφωση με τα πρότυπα. Η ενότητα συμμορφώνεται με τις προδιαγραφές SAML που ορίζονται από οργανισμούς όπως ο OASIS (Οργανισμός για την Προώθηση των Προτύπων Δομημένης Πληροφορίας), διασφαλίζοντας συμβατότητα και διαλειτουργικότητα με άλλα συστήματα που βασίζονται σε SAML.

Επιπλέον, αξιοποιώντας τους ισχυρούς μηχανισμούς αυθεντικοποίησης και εξουσιοδότησης που παρέχονται από το SAML, οι οργανισμοί μπορούν να ενισχύσουν την ασφάλεια της υποδομής SSO τους και να προστατεύσουν ευαίσθητα δεδομένα χρήστη κατά την αυθεντικοποίηση και τη διαχείριση περιόδων σύνδεσης.

Τέλος, συμβάλλει στην επεκτασιμότητα, εφόσον η ενότητα υποστηρίζει κλιμακούμενες διαμορφώσεις ανάπτυξης, επιτρέποντας στους οργανισμούς να προσαρμόσουν τους αυξανόμενους πληθυσμούς χρηστών και να αυξάνουν τη ζήτηση για υπηρεσίες SSO χωρίς να θυσιάζεται η απόδοση ή η αξιοπιστία.

Παρακάτω βρίσκεται το endpoint το οποίο δίνεται από το CAS:

URI	Περιγραφή
/cas/actuator/samlValidate	Χειρισμός αιτήματος επικύρωσης και δημιουργία ωφέλιμου φορτίου SAML1

Πίνακας 4.5 – SAML1.1 URI Endpoint

Το endpoint *samlValidate* που περιγράφεται στην τεκμηρίωση CAS (CAS Documentation), χρησιμεύει ως endpoint για την επικύρωση των ισχυρισμών SAML (SAML Assertions) στο διακομιστή CAS.

Γενικά , το endpoint *samlValidate* είναι υπεύθυνο για την επικύρωση των ισχυρισμών SAML που παρουσιάζονται από τους πελάτες κατά τη διαδικασία αυθεντικοποίησης. Όταν ένας πελάτης παρουσιάζει έναν ισχυρισμό SAML στο διακομιστή CAS για αυθεντικοποίηση, το endpoint *samlValidate* λαμβάνει και επεξεργάζεται τον ισχυρισμό αυτό, με σκοπό να επαληθεύσει τη γνησιότητα και την ακεραιότητά του.

Το endpoint samlValidate εκτελεί διάφορους ελέγχους επικύρωσης στον ισχυρισμό SAML, συμπεριλαμβανομένης της επαλήθευσης της ψηφιακής υπογραφής, του ελέγχου της περιόδου ισχύος του ισχυρισμού και της επικύρωσης της ταυτότητας του εκδότη. Μόλις ολοκληρωθούν επιτυχώς αυτοί οι έλεγχοι, ο διακομιστής CAS μπορεί να προχωρήσει στην αυθεντικοποίηση του χρήστη με βάση τις πληροδορίες που παρέχονται στον ισχυρισμό.

Το endpoint *samlValidate* χρησιμοποιείται συνήθως κατά τη διαδικασία ελέγχου ταυτότητας SAML, όταν ένας πελάτης παρουσιάζει έναν ισχυρισμό SAML ως μέρος του αιτήματος αυθεντικοποίησης. Αυτό το endpoint καλείται από τον διακομιστή CAS για να επικυρώσει τον ισχυρισμό SAML και να καθορίσει την ταυτότητα και την κατάσταση αυθεντικοποίησης του χρήστη.

Οι πελάτες, όπως οι πάροχοι υπηρεσιών SAML (SPs), βασίζονται στο endpoint samlValidate για να διασφαλίσουν την ακεραιότητα και την αυθεντικότητα των ισχυρισμών SAML που λαμβάνουν από παρόχους ταυτότητας (IdP) κατά τη ροή αυθεντικοποίησης. Με την επικύρωση των ισχυρισμών, ο διακομιστής CAS επιβεβαιώνει ότι η ταυτότητα του χρήστη έχει επιβεβαιωθεί από ένα αξιόπιστο IdP και ότι ο ισχυρισμός δεν έχει παραποιηθεί ή πλαστογραφηθεί.

Συγκεκριμένα, χρησιμοποιείται και στέλνεται ένα SAML Αίτημα, με τη μέθοδο HTTP POST. Η ροή του αιτήματος, ξεκινάει όταν ένας πελάτης ξεκινά τη διαδικασία αυθεντικοποίησης με τον διακομιστή CAS, μπορεί να παρουσιάσει έναν ισχυρισμό SAML που λαμβάνεται από ένα εξωτερικό IdP. Ο πελάτης περιλαμβάνει αυτόν τον ισχυρισμό στο αίτημα αυθεντικοποίησης που αποστέλλεται στον διακομιστή CAS. Μόλις λάβει το αίτημα αυθεντικοποίησης με τη δήλωση SAML, ο διακομιστής CAS δρομολογεί το αίτημα στο endpoint samlValidate για επικύρωση. Το endpoint εξάγει τον ισχυρισμό SAML από το ωφέλιμο φορτίο αιτήματος (request payload) και ξεκινά τη διαδικασία επικύρωσης.

Το endpoint samlValidate εκτελεί διάφορους ελέγχους επικύρωσης στον ισχυρισμό, όπως περιεγράφηκε προηγουμένως. Εάν ο ισχυρισμός περάσει όλους τους ελέγχους επικύρωσης, ο διακομιστής CAS προχωρά στην αυθεντικοποίηση του χρήστη και στην έκδοση ενός δελτίου υπηρεσίας ή στην παραχώρηση πρόσβασης στον ζητούμενο πόρο. Εάν ο ισχυρισμός αποτύχει σε κάποιον από τους ελέγχους επικύρωσης, ο διακομιστής CAS απορρίπτει το αίτημα ελέγχου ταυτότητας και επιστρέφει μια απάντηση σφάλματος στον πελάτη, υποδεικνύοντας ότι ο ισχυρισμός δεν είναι έγκυρος ή δεν μπορεί να είναι αξιόπιστος.

Ένα ενδεικτικό αίτημα φαίνεται παρακάτω:

```
POST /cas/samlValidate?ticket=
Host: cas.example.com
Content-Length: 491
Content-Type: text/xml

<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <samlp:Request xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
MajorVersion="1"
      MinorVersion="1" RequestID="_192.168.16.51.1024506224022"
      IssueInstant="2002-06-19T17:03:44.022Z">
      <samlp:AssertionArtifact>
        ST-1-u4hrm3td92cLxpCvrjylcas.example.com
      </samlp:AssertionArtifact>
    </samlp:Request>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Κατά συνέπεια επιστρέφεται μια απόκριση SAML1.1 που ενσωματώνεται σε ένα φάκελο SOAP (SOAP envelope – Simple Object Access Protocol). Η απάντηση δημιουργείται από έναν πάροχο ταυτότητας (IdP) ως απόκριση σε ένα αίτημα ελέγχου ταυτότητας από έναν πάροχο υπηρεσιών (SP).

Το στοιχείο <SOAP-ENV:Envelope> χρησιμεύει ως το βασικό στοιχείο του μηνύματος SOAP, ορίζοντας τον χώρο ονομάτων XML για το SOAP. Εντός του φακέλου, το στοιχείο <SOAP-ENV:Body> περιέχει το πραγματικό ωφέλιμο φορτίο απόκρισης SAML (actual SAML response payload). Το στοιχείο <saml1p:Response> αντιπροσωπεύει την ίδια την απάντηση SAML, με

χαρακτηριστικά όπως InResponseTo, IssueInstant, MajorVersion, MinorVersion και ResponseID που παρέχουν μεταδεδομένα σχετικά με την απόκριση.

Μέσα στην απόκριση, το στοιχείο <saml1p:Status> υποδεικνύει τη συνολική κατάσταση της απόκρισης, με ένα θυγατρικό στοιχείο <saml1p:StatusCode> που καθορίζει την τιμή κατάστασης ως saml1p:Success. Το στοιχείο <saml1:Assertion> ενσωματώνει τον ισχυρισμό SAML που εκδόθηκε από το IdP. Περιέχει μεταδεδομένα όπως AssertionID, IssueInstant, Issuer, MajorVersion και MinorVersion. Εντός του ισχυρισμού, το στοιχείο <saml1:Conditions> ορίζει τις συνθήκες υπό τις οποίες είναι έγκυρος ο ισχυρισμός, συμπεριλαμβανομένων των χρονικών σφραγίδων NotBefore και NotOnOrAfter και ενός <saml1:AudienceRestrictionCondition> που προσδιορίζει το κοινό για το οποίο απευθύνεται ο ισχυρισμός.

Το στοιχείο <saml1:AuthenticationStatement> παρέχει πληροφορίες σχετικά με το συμβάν αυθεντικοποίησης, συμπεριλαμβανομένης της χρονικής σφραγίδας AuthenticationInstant και του AuthenticationMethod που χρησιμοποιείται. Το στοιχείο <saml1:AttributeStatement> περιέχει ένα ή περισσότερα στοιχεία <saml1:Attribute>, καθένα από τα οποία αντιπροσωπεύει ένα χαρακτηριστικό που σχετίζεται με τον πιστοποιημένο χρήστη. Τα χαρακτηριστικά μπορεί να περιλαμβάνουν AttributeName, AttributeNamespace και ένα ή περισσότερα στοιχεία <saml1:AttributeValue> που περιέχουν τις τιμές των χαρακτηριστικών.

Συνοπτικά, ο παρεχόμενος κώδικας XML αντιπροσωπεύει μια απόκριση SAML1.1 που περιέχει πληροφορίες αυθεντικοποίησης και χαρακτηριστικών που εκδίδονται από ένα IdP ως απόκριση σε αίτημα αυθεντικοποίησης, σύμφωνα με τις προδιαγραφές του πρωτοκόλλου SAML1.1.

Κεφάλαιο 5: Γενικά Συμπεράσματα

Στην παρούσα Διπλωματική εργασία μελετήθηκε η τεχνολογία SSO με τη σύμπραξη των πρωτοκόλλων CAS και SAML, με σκοπό την ανάπτυξη μιας βιβλιοθήκης αυθεντικοποίησης χρηστών για το Πανεπιστήμιο Δυτικής Αττικής. Κατά την εκπόνησή της παρατηρήθηκαν, εντοπίστηκαν και ξεπεράστηκαν αρκετές δυσκολίες και προβλήματα τόσο σε ζητήματα υποδομής, όσο και σε ζητήματα υλοποίησης.

Μέσω της λύσης SSO που αναπτύχθηκε, λύνεται κατά ένα μεγάλο ποσοστό το κύριο πρόβλημα του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, το οποίο είναι η αύξηση δημιουργίας διαφορετικών ιδρυματικών εφαρμογών και υπηρεσιών και κατά συνέπεια η αύξηση της ανάγκης για καλύτερη διαχείριση του φόρτου, αλλά και της διασφάλισης της καλής εμπειρίας χρήστη, δίνοντας έμφαση σε ζητήματα ασφάλειας, κυρίως στην εμπιστευτικότητα.

Στην αρχή, οι στόχοι οριοθετήθηκαν με σαφήνεια, με στόχο την αντιμετώπιση της πολυπλοκότητας της αυθεντικοποίησης χρήστη, αξιοποιώντας παράλληλα τα οφέλη αποτελεσματικότητας και ασφάλειας των μηχανισμών Single Sign-On (SSO). Σε όλη τη διαδικασία ανάπτυξης, χρησιμοποιήθηκαν μεθοδολογίες που συνδυάζουν την αυστηρή έρευνα με την πρακτική εφαρμογή, που στηρίζονται στη συνετή επιλογή τεχνολογιών και εργαλείων.

Παρά τις προκλήσεις που συναντήθηκαν στην πορεία, κάθε εμπόδιο χρησίμευσε ως ευκαιρία για ανάπτυξη και τελειοποίηση, οδηγώντας τελικά στη δημιουργία μιας ολοκληρωμένης βιβλιοθήκης Java. Η αρχιτεκτονική τη συγκεκριμένης βιβλιοθήκης, υποστηρίζει την κατανόηση των πρωτοκόλλων SSO, όπως το SAML και το CAS, διασφαλίζοντας απρόσκοπτη ενοποίηση και συμβατότητα με διαφορετικά περιβάλλοντα αυθεντικοποίησης.

Κατά τον έλεγχο αποδοτικότητας της βιβλιοθήκης, πραγματοποιήθηκαν διαδικασίες επικύρωσης, ενδυναμώνοντας τους βασικούς πυλώνες της ασφάλειας. Τα εμπειρικά δεδομένα που συλλέχθηκαν και τα σχόλια των χρηστών που συγκεντρώθηκαν κατά τη διάρκεια της αξιολόγησης της βιβλιοθήκης, βοήθησαν στην περαιτέρω βελτίωσή της.

Επιπλέον, οι συνεισφορές του έργου εκτείνονται πέρα από τα όρια της βάσης κωδικών, προσφέροντας νέες ιδέες και μεθοδολογίες που προάγουν τη συζήτηση γύρω από την αυθεντικοποίηση χρηστών και την εφαρμογή της τεχνολογίας SSO.

Αναγνωρίζοντας τους περιορισμούς και τις δυνατότητες μελλοντικής εξερεύνησης, η παρούσα διατριβή ολοκληρώνεται, έχοντας ως στόχο τη διαρκή βελτιστοποίηση τόσο των υποδομών στον οποίο υλοποιείται, όσο και των τεχνολογιών και εργαλείων που χρησιμοποιούνται, δίνοντας ευκαιρίες για συνεχή καινοτομία και βελτίωση στον τομέα της εμπιστευτικότητας και βελτίωσης της εμπειρίας χρήστη.

5.1 Μελλοντικές προσθήκες και Βελτιστοποίηση

Η βιβλιοθήκη που αναπτύχθηκε στα πλαίσια της παρούσας Διπλωματικής εργασίας, θα αποτελέσει τη βάση για τη διασύνδεση του διακομιστή του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλαδή η δημιουργία ενός metaserver, ο οποίος θα είναι πιστοποιημένος να χρησιμοποιεί τον CAS διακομιστή του Ιδρύματος και η χρήση του από τις προσωρινές εφαρμογές που χρειάζονται αυθεντικοποίηση με τους Ιδρυματικούς Λογαριασμούς.

Μια πολύ σημαντική βελτιστοποίηση, θα ήταν η αναβάθμιση του CAS διακομιστή του Ιδρύματος, έτσι ώστε να υποστηρίζει νεότερες εκδόσεις των πρωτοκόλλων CAS και SAML. Η πιο πρόσφατη έκδοση του πρωτοκόλλου CAS είναι το CAS 6.x, ενώ η τελευταία έκδοση του πρωτοκόλλου SAML είναι το SAML 2.0. Τόσο το CAS 6.x όσο και το SAML2.0 εισάγουν σημαντικές προόδους και βελτιώσεις σε σχέση με τις αντίστοιχες πρώτες εκδόσεις που υποστηρίζονται από τον CAS διακομιστή του ιδρύματος.

Το CAS 6.x, συγκεκριμένα, ενσωματώνει βελτιώσεις στην επεκτασιμότητα και την απόδοση, προσφέροντας βελτιωμένη υποστήριξη για σύγχρονους μηχανισμούς αυθεντικοποίησης, αυθεντικοποίησης πολλαπλών παραγόντων (MFA) και προσαρμοστικές πολιτικές αυθεντικοποίησης. Επιπλέον, το CAS 6.x εισάγει λειτουργίες όπως υποστήριξη OAuth/OpenID Connect, λεπτομερή έλεγχο πρόσβασης και βελτιωμένες δυνατότητες ενσωμάτωσης με συστήματα διαχείρισης ταυτότητας και πρόσβασης (IAM).

Από την άλλη πλευρά, το SAML 2.0 προσφέρει βελτιωμένα χαρακτηριστικά ασφαλείας, όπως υποστήριξη για ψηφιακές υπογραφές, κρυπτογράφηση και ισχυρισμούς με πιο αναλυτικό έλεγχο πρόσβασης σε επίπεδο χαρακτηριστικών. Επιπλέον, το SAML2.0 παρέχει βελτιωμένη διαλειτουργικότητα, επεκτασιμότητα και συμβατότητα με σύγχρονα πρότυπα και πρωτόκολλα ιστού, καθιστώντας ευκολότερη την ενσωμάτωση με ένα ευρύ φάσμα Παρόχων Ταυτότητας (IdP) και Παρόχων Υπηρεσιών (SP).

Μια ακόμα χρήσιμη προσθήκη θα ήταν η αποδοτικότερη διαχείριση συνεδριών (session management). Συγκεκριμένα, η διαχείριση συνεδρίας διαδραματίζει κρίσιμο ρόλο στα συστήματα SSO διατηρώντας την κατάσταση αυθεντικοποίησης χρήστη σε πολλαπλές εφαρμογές και υπηρεσίες. Με τη διαχείριση περιόδων σύνδεσης, οι χρήστες μπορούν να έχουν απρόσκοπτη πρόσβαση σε διαφορετικούς πόρους χωρίς να χρειάζεται να ελέγχουν εκ νέου την ταυτότητα κάθε φορά που κάνουν εναλλαγή μεταξύ εφαρμογών. Η εφαρμογή της διαχείρισης περιόδων σύνδεσης στη βιβλιοθήκη, θα συνεπαγόταν την αποτελεσματικότερη και πέρα από τους περιορισμούς του πρωτοκόλλου CAS διαχείριση διακριτικών συνεδρίας ή cookie που εκδίδονται κατά τη διαδικασία αυθεντικοποίησης και τη διασφάλιση της εγκυρότητας και της ασφάλειάς τους καθ' όλη τη διάρκεια της περιόδου λειτουργίας του χρήστη.

Τέλος, η ενσωμάτωση της λειτουργίας μεμονωμένης αποσύνδεσης στη βιβλιοθήκη (Single Logout) θα επιτρέψει στους χρήστες να αποσυνδεθούν από όλες τις εφαρμογές και τις υπηρεσίες που είναι συνδεδεμένες στο σύστημα SSO με μια μόνο ενέργεια. Η απλή αποσύνδεση συμβάλλει στη βελτίωση της ασφάλειας και της ευκολίας των χρηστών, διασφαλίζοντας ότι οι ρήστες αποσυνδέονται πλήρως από όλες τις εφαρμογές και τις υπηρεσίες όταν επιλέγουν να τερματίσουν τη συνεδρία τους.

Παράρτημα

Κώδικας Java – Classes

Database.java

```
package org.marg.pada.marvinauth.data;

import java.net.Inet4Address;
import java.net.UnknownHostException;
import java.util.ArrayList;

/**
 * Singleton που περιέχει όλα τα δεδομένα που διαχειρίζεται το
σύστημα.
 * @author marg
 */
public class Database
{
    private static Database Instance = null;

    ArrayList <PC> ProjectServers;
    ArrayList<Ticket> Tickets;

    public static Database GetInstance ()
    {
        if (Instance == null)
        {
            Instance = new Database ();
        }
        return Instance;
    }

    private Database ()
    {
        ProjectServers = new ArrayList ();
        LoadProjectServers ();
        Tickets = new ArrayList ();

    }


    /**
     * Δημιουργεί, καρφωτά, την λίστα με τους συνεργαζόμενους Servers.
Αν προστεθεί νέο συναργαζόμενο project θα πρέπει να ενημερωθεί ο
     * κώδικας και το project να γίνει redeploy.
```

```
     */
    private void LoadProjectServers ()
    {
        try
        {
            PC tmp;
            tmp = new PC ("Marvin", "Marvin", (Inet4Address)
Inet4Address.getByName ("195.130.106.91"));
            ProjectServers.add (tmp);
            tmp = new PC ("Enterprise", "Test 1", (Inet4Address)
Inet4Address.getByName ("Enterprise.marg.gr"));
            ProjectServers.add (tmp);
        }
        catch (UnknownHostException e)
        {
            System.out.println ("@@@ Error: Invalid Servers'
Initialization. Should Never Happen.");
        }
    }

}
```

DatabaseSQL.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package org.marg.pada.marvinauth.data;

/**
 *
 * @author Marga
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class DatabaseSQL {
    // JDBC URL, username, and password of the database
    private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/root";
    private static final String USERNAME = "username";
    private static final String PASSWORD = "password";
```

```
    // SQL queries to insert data into the Requests, Tickets, and SAML
Assertions tables
    private static final String INSERT_REQUEST_SQL = "INSERT INTO
Requests (IPAddress, ServiceName, Timestamp) VALUES (?, ?, ?)";
    private static final String INSERT_TICKET_SQL = "INSERT INTO
Tickets (RequestID, Username, Roles, Attributes, ExpirationTime)
VALUES (?, ?, ?, ?, ?)";
    private static final String INSERT_ASSERTION_SQL = "INSERT INTO
SAMLAssertions (TicketID, AssertionXML, Issuer, Subject, IssueInstant,
NotBefore, NotOnOrAfter) VALUES (?, ?, ?, ?, ?, ?, ?)";

    // Method to store request information in the Requests table
    public void storeRequest(String ipAddress, String serviceName,
long timestamp) throws SQLException {
        try (Connection connection =
DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
             PreparedStatement statement =
connection.prepareStatement(INSERT_REQUEST_SQL)) {
            statement.setString(1, ipAddress);
            statement.setString(2, serviceName);
            statement.setLong(3, timestamp);
            statement.executeUpdate();
        }
    }

    // Method to store ticket information in the Tickets table
    public void storeTicket(int requestID, String username, String
roles, String attributes, long expirationTime) throws SQLException {
        try (Connection connection =
DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
             PreparedStatement statement =
connection.prepareStatement(INSERT_TICKET_SQL)) {
            statement.setInt(1, requestID);
            statement.setString(2, username);
            statement.setString(3, roles);
            statement.setString(4, attributes);
            statement.setLong(5, expirationTime);
            statement.executeUpdate();
        }
    }

    // Method to store SAML assertion information in the SAML
Assertions table
    public void storeAssertion(int ticketID, String assertionXML,
String issuer, String subject, long issueInstant, long notBefore, long
notOnOrAfter) throws SQLException {
        try (Connection connection =
DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
```

```
             PreparedStatement statement =
connection.prepareStatement(INSERT_ASSERTION_SQL)) {
            statement.setInt(1, ticketID);
            statement.setString(2, assertionXML);
            statement.setString(3, issuer);
            statement.setString(4, subject);
            statement.setLong(5, issueInstant);
            statement.setLong(6, notBefore);
            statement.setLong(7, notOnOrAfter);
            statement.executeUpdate();
        }
    }
}
```

PC.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package org.marg.pada.marvinauth.data;

import java.net.Inet4Address;

/**
 * Βασικές πληροφορίες ενός PC. Μπορεί να είναι Server, μπορεί να
είναι ένας client
 * @author marg
 */
public class PC
{
    private String ID;
    private String Project;
    private Inet4Address Addr;

    public PC (String ID, String Name, Inet4Address Addr)
    {
        this.ID = ID;
        this.Project = Name;
        this.Addr = Addr;
    }

    public String getID ()
    {
        return ID;
    }
```

```
    public void setID (String ID)
    {
        this.ID = ID;
    }

    public String getProject ()
    {
        return Project;
    }

    public void setProject (String Project)
    {
        this.Project = Project;
    }

    public Inet4Address getAddr ()
    {
        return Addr;
    }

    public void setAddr (Inet4Address Addr)
    {
        this.Addr = Addr;
    }
}
```

Ticket.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
to edit this template
 */
package org.marg.pada.marvinauth.data;

import java.time.LocalDateTime;

/**
 * Παριστά ένα Cas ticket και τις συνοδευτικές πληροφορίες
 * @author marg
 */
public class Ticket
{
    private String CasTicket;
    private String Server;
    private LocalDateTime Creation;
```

```
    private LocalDateTime LastAccess;
    private LocalDateTime Expire;

    public Ticket (String CasTicket, String Server)
    {
        this.CasTicket = CasTicket;
        this.Server = Server;
        Creation = null;
        LastAccess = null;
        Expire = null;
    }

    public String getCasTicket ()
    {
        return CasTicket;
    }

    public void setCasTicket (String CasTicket)
    {
        this.CasTicket = CasTicket;
    }

    public String getServer ()
    {
        return Server;
    }

    public void setServer (String Server)
    {
        this.Server = Server;
    }

    public LocalDateTime getCreation ()
    {
        return Creation;
    }

    public void setCreation (LocalDateTime Creation)
    {
        this.Creation = Creation;
    }

    public LocalDateTime getLastAccess ()
    {
        return LastAccess;
    }

    public void setLastAccess (LocalDateTime LastAccess)
```

```
    {
        this.LastAccess = LastAccess;
    }

    public LocalDateTime getExpire ()
    {
        return Expire;
    }

    public void setExpire (LocalDateTime Expire)
    {
        this.Expire = Expire;
    }
}
```

Authent.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click
nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to
edit this template
 */
package org.marg.pada.marvinauth.endpoints;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.time.Clock;
import java.time.Instant;
import java.time.LocalTime;
import java.util.Base64;
import java.util.Enumeration;
import java.util.List;
import java.util.Map;
import javax.net.ssl.HttpsURLConnection;
```

```
/**
 *
 * @author marg
 */
@WebServlet(name = "authent", urlPatterns = {"/authent"})
public class Authent extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest (HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException
    {


        response.setContentType("text/plain;charset=UTF-8");
        String CasTicket = request.getParameter ("ticket");
        PrintWriter out = response.getWriter();
        out.println ("Ticket : " + CasTicket);
        Enumeration enu = request.getHeaderNames();
        while (enu.hasMoreElements())
        {
            String headerName = (String) enu.nextElement();
            String headerValue = request.getHeader(headerName);
            out.print(headerName + ":");
            out.println(headerValue);
        }

        Enumeration<String> attributes =
request.getSession().getAttributeNames();
        while (attributes.hasMoreElements())
        {
            String attribute = (String) attributes.nextElement();
            out.println("$" + attribute+" :
"+request.getSession().getAttribute(attribute));
        }

        BufferedReader BR = new BufferedReader (new InputStreamReader
```

```
(request.getInputStream()));
        String str;
        while ((str = BR.readLine()) != null)
            out.println ("!" + str);


        out.println ("--------");
        String CookieLine = request.getHeader ("Cookie");
        out.println ("!" + CookieLine);
        System.out.println ("EPLogin:Validate Ticket : " + CasTicket +
" " +LocalTime.now ());
        System.out.flush ();


        String ValidateUrl = "https://sso.uniwa.gr/validate?ticket=" +
CasTicket
            +
"&service=https%3A%2F%2Fmarvin.ice.uniwa.gr%3A40032%2Fauthent";
        URL url = new URL(ValidateUrl);
        System.out.println ("!!!" + url.toString ());
        response.sendRedirect (ValidateUrl);


        HttpsURLConnection connection = (HttpsURLConnection)
url.openConnection();

        connection.setRequestMethod("GET");
        connection.setInstanceFollowRedirects(false);
        connection.setDoInput (true);
        connection.setDoOutput (true);
        connection.setRequestProperty ("Cookie", CookieLine);
        connection.connect();

        int ResponseCode = connection.getResponseCode();
        System.out.println("EPLogin: Response Code = " +
ResponseCode);
        System.out.flush();
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        out.println("EPLogin: Response Code = " + ResponseCode);
        for (int i = 0;; i++)
        {
            String headerName = connection.getHeaderFieldKey(i);
            String headerValue = connection.getHeaderField(i);
            out.println (headerName + " : " + headerValue);

            if (headerName == null && headerValue == null) {
                out.println("No more headers");
```

```
                break;
            }
        }


        BufferedReader BR1 = new BufferedReader (new InputStreamReader
(connection.getInputStream()));
        String str1;
        while ((str1 = BR1.readLine()) != null)
            out.println ("!!" + str1);

        out.println ("==================================");


        byte[] PostBytes = CreateSAMLPostData3 (CasTicket);
        out.println (new String (PostBytes, StandardCharsets.UTF_8));


        //String UrlParams = "ticket=" + CasTicket +
"&service=https%3A%2F%2Fmarvin.ice.uniwa.gr%3A40032%2Fauthent";
        //byte[] PostBytes = UrlParams.getBytes (
StandardCharsets.UTF_8 );
        //SAMLOut.write (PostBytes);


        URL SamlUrl = new URL
("https://sso.uniwa.gr:443/samlValidate");
        HttpsURLConnection SAMLCon = (HttpsURLConnection)
SamlUrl.openConnection ();
        SAMLCon.setRequestMethod ("POST");
        SAMLCon.setDoInput (true);
        SAMLCon.setDoOutput (true);
        SAMLCon.setRequestProperty ("Authorization", new
String(Base64.getEncoder().encode("marg:osfp!!!!".getBytes())));
        SAMLCon.setRequestProperty ("Host", "sso.uniwa.gr:443");
        SAMLCon.setRequestProperty ("User-Agent", "Java/15.0.1");
        SAMLCon.setRequestProperty ("TARGET",
"https%3A%2F%2sso.uniwa.gr%2FsamlValidate");
        SAMLCon.setRequestProperty ("ticket", CasTicket);
        SAMLCon.setRequestProperty ("Content-Length", Integer.toString
(PostBytes.length));
        SAMLCon.setRequestProperty ("Content-Language", "en-US");
        SAMLCon.setRequestProperty ("Cookie", CookieLine.replace
("cookie:", CasTicket));
        SAMLCon.setRequestProperty ("upgrade-insecure-requests", "1");
        SAMLCon.setRequestProperty ("sec-fetch-dest", "document");
        SAMLCon.setRequestProperty ("sec-fetch-mode", "navigate");
        SAMLCon.setRequestProperty ("sec-fetch-site", "same-site");
```

```
        SAMLCon.setRequestProperty ("sec-fetch-user", "?1");
        SAMLCon.setRequestProperty ("te", "trailers");
        SAMLCon.setRequestProperty ("soapaction", "http://www.oasis-
open.org/committees/security");
        SAMLCon.setRequestProperty ("cache-control", "no-cache");
        //SAMLCon.setRequestProperty ("cache-control", "private");
        SAMLCon.setRequestProperty ("pragma", "no-cache");
        SAMLCon.setRequestProperty ("accept", "text/xml");
        SAMLCon.setRequestProperty ("connection", "keep-alive");
        SAMLCon.setRequestProperty ("content-type", "text/xml");


        Map <String, List<String>> Head = SAMLCon.getRequestProperties
();
        for (String tmp: Head.keySet ())
        {
            out.println ("!!" + tmp);
            for (String tmp1: Head.get (tmp))
                out.println ("  " + tmp1);
        }


        DataOutputStream SAMLOut = new DataOutputStream
(SAMLCon.getOutputStream ());
        SAMLOut.write(PostBytes);
        SAMLOut.close ();
        SAMLCon.connect ();



        int SAMLResCode = SAMLCon.getResponseCode();
        System.out.println("EPLogin: SAML Response Code = " +
SAMLResCode);
        System.out.flush();
        //response.setContentType("text/plain");
        //PrintWriter out = response.getWriter();
        out.println("EPLogin: SAML Response Code = " + SAMLResCode);
        out.flush();

        for (int i = 0;; i++)
        {
            String headerName = SAMLCon.getHeaderFieldKey(i);
            String headerValue = SAMLCon.getHeaderField(i);
            out.println (headerName + " : " + headerValue);

            if (headerName == null && headerValue == null) {
                out.println("No more headers");
                break;
```

```
            }
        }

        InputStream SS;
        if (SAMLResCode == 412)
            SS = SAMLCon.getErrorStream ();
        else
            SS = SAMLCon.getInputStream ();

        if (SS != null)
        {
            BufferedReader BR2 = new BufferedReader (new
InputStreamReader (SS));
            String str2;
            while ((str2 = BR2.readLine()) != null)
                out.println ("##" + str2);
        }
        else
            out.println ("NULL STREAM");



        out.println ("==================================");

    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet
methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
```

```
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>


    /**
     * Soap Only XML
     * @param Ticket
     * @return
     * @throws UnsupportedEncodingException
     */
    byte[] CreateSAMLPostData1 (String Ticket) throws
UnsupportedEncodingException
    {
        StringBuilder SB = new StringBuilder ();
        SB.append ("ticket=" + Ticket +
"&TARGET=https%3A%2F%2Fmarvin.ice.uniwa.gr%3A40032%2Fauthent" + "\n");
        //SB.append ("<?xml version=\"1.0\">" + "\n");
        SB.append ("<SOAP-ENV:Envelope xmlns:SOAP-
ENV=\"http://schemas.xmlsoap.org/soap/envelope/\">" + "\n");
        SB.append ("<SOAP-ENV:Header/>" + "\n");
        SB.append ("<SOAP-ENV:Body>" + "\n");
        SB.append ("<samlp:Request
xmlns:samlp=\"urn:oasis:names:tc:SAML:1.1:protocol\"
MajorVersion=\"1\" ");
        SB.append ("MinorVersion=\"1\"
RequestID=\"_195.130.106.91.102458806224022\" ");
        SB.append ("IssueInstant=\"").append (Instant.now ()).append
("\">\n");
        SB.append ("<samlp:AssertionArtifact> ");
        SB.append (Ticket);
        SB.append (" </samlp:AssertionArtifact>\n");
```

```
        SB.append ("</samlp:Request>" + "\n");
        SB.append ("</SOAP-ENV:Body>" + "\n");
        SB.append ("</SOAP-ENV:Envelope>" + "\n");
        String Data = SB.toString ();
        byte[] Bytes1 = Data.getBytes("utf-8");
        return Bytes1;

    }

    byte[] CreateSAMLPostData2 (String Ticket) throws
UnsupportedEncodingException
    {
        StringBuilder SB = new StringBuilder ();
        //SB.append ("<?xml version=\"1.0\">" + "\n");
        SB.append ("<SOAP-ENV:Envelope xmlns:SOAP-
ENV=\"http://schemas.xmlsoap.org/soap/envelope/\">" + "\n");
        SB.append ("<SOAP-ENV:Header/>" + "\n");
        SB.append ("<SOAP-ENV:Body>" + "\n");
        SB.append ("<samlp:ArtifactResolve
xmlns:samlp=\"urn:oasis:names:tc:SAML:1.0:protocol\" ");
        SB.append ("IssueInstant=\"").append (Instant.now ()).append
("\">\n");
        SB.append ("ID=\"123gg456\">" + "\n");
        SB.append ("<samlp:Artifact> ");
        //SB.append (Ticket +"\n");
        SB.append ("marg2");
        SB.append (" </samlp:Artifact>\n");
        SB.append ("</samlp:ArtifactResolve>" + "\n");
        SB.append ("</SOAP-ENV:Body>" + "\n");
        SB.append ("</SOAP-ENV:Envelope>" + "\n");
        String Data = SB.toString ();
        byte[] Bytes1 = Data.getBytes("utf-8");
        return Bytes1;

    }

    byte[] CreateSAMLPostData3 (String Ticket) throws
UnsupportedEncodingException
    {
        StringBuilder SB = new StringBuilder ();
        //SB.append ("TARGET=https%3A%2F%2sso.uniwa.gr&SAMLart=" +
Ticket +"\n");
        SB.append
("TARGET=https%3A%2F%2sso.uniwa.gr%2FsamlValidate&ticket=" + Ticket
+"\n");
        //SB.append
("TARGET=https%3A%2F%2marvin.ice.uniwa.gr%2Fauthent&ticket=" + Ticket
+"\n");
```

```
        //SB.append
("TARGET=https%3A%2F%2Fsso.uniwa.gr%2FsamlValidate" + "\n");
        SB.append ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
        SB.append ("<SOAP-ENV:Envelope xmlns:SOAP-
ENV=\"http://schemas.xmlsoap.org/soap/envelope/\">" + "\n");
        SB.append ("<SOAP-ENV:Header/>" + "\n");
        SB.append ("<SOAP-ENV:Body>" + "\n");
        SB.append ("<samlp:Request
xmlns:samlp=\"urn:oasis:names:tc:SAML:1.0:protocol\"
MajorVersion=\"1\" ");
        SB.append ("MinorVersion=\"1\"
RequestID=\"_195.130.106.91.1024506224022\" ");
        SB.append ("IssueInstant=\"").append (Instant.now
(Clock.systemDefaultZone ())).append ("\">\n");
        SB.append ("<samlp:AssertionArtifact>");
        SB.append (Ticket);
        SB.append ("</samlp:AssertionArtifact>\n");
        SB.append ("</samlp:Request>" + "\n");
        SB.append ("</SOAP-ENV:Body>" + "\n");
        SB.append ("</SOAP-ENV:Envelope>" + "\n");
        String Data = SB.toString ();
        System.out.println (Data);
        byte[] Bytes1 = Data.getBytes("utf-8");
        return Bytes1;

    }
}
```

EPValidate.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click
nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to
edit this template
 */
package org.marg.pada.marvinauth.endpoints;


import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.util.Enumeration;


/**
 *
 * @author marg
 */
public class EPValidate extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println ("EPValidate: Here !!!!" +
request.getParameter ("ticket"));
        System.out.flush ();
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter())
        {
            response.setContentType("text/plain");
            out.println ("--------");
            BufferedReader BR = new BufferedReader (new
InputStreamReader (request.getInputStream()));
            String str;
            while ((str = BR.readLine()) != null)
                out.println (str);
            out.println ("--------");
            Enumeration enu = request.getHeaderNames();
            while (enu.hasMoreElements())
            {
                String headerName = (String) enu.nextElement();
                String headerValue = request.getHeader(headerName);
                out.print(headerName + "  :  ");
                out.println(headerValue);
            }
            out.println ("--------");
        }
```

```
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet
methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.marg.pada</groupId>
    <artifactId>MarvinAuthent</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>MarvinAuthent</name>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>

<endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <failOnMissingWebXml>false</failOnMissingWebXml>
        <jakartaee>10.0.0</jakartaee>
    </properties>

    <dependencies>
        <dependency>
            <groupId>jakarta.platform</groupId>
            <artifactId>jakarta.jakartaee-api</artifactId>
            <version>${jakartaee}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <type>jar</type>
        </dependency>
        <dependency>
        <groupId>org.apereo.cas</groupId>
        <artifactId>cas-server-support-saml</artifactId>
        <version>1</version>
    </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
```

```
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                    <compilerArguments>
                        <endorseddirs>${endorsed.dir}</endorseddirs>
                    </compilerArguments>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.1</version>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <version>2.6</version>
                <executions>
                    <execution>
                        <phase>validate</phase>
                        <goals>
                            <goal>copy</goal>
                        </goals>
                        <configuration>

<outputDirectory>${endorsed.dir}</outputDirectory>
                            <silent>true</silent>
                            <artifactItems>
                                <artifactItem>

<groupId>jakarta.platform</groupId>
                                    <artifactId>jakarta.jakartaee-
api</artifactId>
                                    <version>${jakartaee}</version>
                                    <type>jar</type>
                                </artifactItem>
                            </artifactItems>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
```

```
    </build>
</project>
```

Βιβλιογραφία και Πηγές

[1] Github, «CAS Protocol», 2023. [Ηλεκτρονικό]. Available: https://apereo.github.io/cas/6.6.x/protocol/CAS-Protocol.html), Author: Drew Mazurek, 01-12-2017.

[2] Github, «CAS Protocol 3.0 Specification», 2023. [Ηλεκτρονικό]. Available: https://apereo.github.io/cas/7.0.x/protocol/CAS-Protocol-Specification.html#head3.6), Author: Drew Mazurek, 01-12-2017.

[3] Github, «SAML v1.1 Protocol», 2023. [Ηλεκτρονικό]. Available: https://apereo.github.io/cas/6.6.x/protocol/SAML-Protocol.html), Author: Drew Mazurek, 01-12-2017.

[4] Wikipedia, «Central Authentication Service», 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Central_Authentication_Service), Author: Unknown, 15-12-2023.

[5] UCONN, «The CAS Protocol for Application Owners», 2023. [Ηλεκτρονικό]. Available: https://iam.uconn.edu/the-cas-protocol-for-application-owners/), Author: UCONN, 2022.

[6] Texas A&M University, «CAS Architecture, Technical Requirements and Client Setup», 2023. [Ηλεκτρονικό]. Available: https://docs.identity.tamu.edu/help/help.html), Author: TAMU, 2023.

[7] UC Berkeley, «How CAS Works», 2023. [Ηλεκτρονικό]. Available: https://calnet.berkeley.edu/calnet-technologists/cas/how-cas-works), Author: UC Berkeley, 2021.

[8] Spring Documentation, «CAS Authentication», 2023. [Ηλεκτρονικό]. Available: https://docs.spring.io/spring-security/site/docs/3.1.x/reference/cas.html), Author: Spring Documentation, 2021.

[9] IEEE, «Research and Design of CAS Protocol Identity Authentication», 2023. [Ηλεκτρονικό]. Available: https://ieeexplore.ieee.org/document/9270440), Author: HaoYuan Zhou, LIGU ZHU, 2020.

[10] AudioCodes Digital VoIP Media Gateways, «CAS Protocol Tables», 2023. [Ηλεκτρονικό]. Available: https://www.audiocodes.com/media/15588/cas-protocol-table-configuration-note-ver-74.pdf), Author: Documentation, November - 2020.

[11] Indiana University, «Connect to UI Login with the CAS Protocol», 2023. [Ηλεκτρονικό]. Available: https://kb.iu.edu/d/bfpq), Author: University Information Technology Services, 2015.

[12] M. Amin Saghizadeh, «Introducing CAS 3.0 protocol», 2023. [Ηλεκτρονικό]. Available: https://www.slideshare.net/MohammadAminSaghizad/introducing-cas-30-protocol-security-and-performance), Author: M. Amin Saghizadeh, September - 2016.

[13] OASIS, «Security Assertion Markup Language (SAML) v2.0 Technical Overview», 2023. [Ηλεκτρονικό]. Available: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html), Author: Hal Lockhart, Brian Campbell, March - 2008.

[14] OKTA, «SAML», 2023. [Ηλεκτρονικό]. Available: https://developer.okta.com/docs/concepts/saml/), Author: OKTA Documentation, 2017.

[15] Cloudfare, «SAML», 2023. [Ηλεκτρονικό]. Available: https://www.cloudflare.com/learning/access-management/what-is-saml/), Author: Cloudfare Documentation, 2017.

[16] IBM, «SAML 2.0 bindings», 2023. [Ηλεκτρονικό]. Available: https://www.ibm.com/docs/en/sva/9.0.4?topic=federations-saml-20-bindings), Author: IBM Security Verify Access Documentation, 2021.

[17] Oracle, «What is Security Assertion Markup Language (SAML)?», 2023. [Ηλεκτρονικό]. Available: https://www.oracle.com/security/cloud-security/what-is-saml/), Author: Oracle Documentation, 2023.

[18] SAML Community, «SAML Specifications», 2023. [Ηλεκτρονικό]. Available: https://www.oracle.com/security/cloud-security/what-is-saml/), Author: SAML Community OASIS Standard, 2013.

[19] WSO2, «Introduction to Security Assertion Markup Language 2.0», 2023. [Ηλεκτρονικό]. Available: https://wso2.com/library/articles/2014/02/introduction-to-security-assertion-markup-language-2.0/), Author: Senior Software Engineer – WSO2, 2017.

[20] Jetbrains, «SAML 2.0», 2023. [Ηλεκτρονικό]. Available: https://www.jetbrains.com/help/hub/saml-management.html), Author: Hub Documentation–Jetbrains, 2023.

[21] Jetbrains, «What is SAML? SAML Authentication & Security Assertion Markup Language Explained», 2023. [Ηλεκτρονικό]. Available: https://www.splunk.com/en_us/blog/learn/saml-authentication.html), Author: Splunk, May 2023.

[22] loginradius, «SAML Overview », 2023. [Ηλεκτρονικό]. Available: https://www.loginradius.com/docs/single-sign-on/tutorial/federated-sso/saml/overview/), Author: loginradius Documentation, 2023.

[23] the art of service, «Will you be using SAML Authentication with a trusted identity provider? », 2023. [Ηλεκτρονικό]. Available: https://theartofservice.com/how-do-you-convert-a-saml-protocol-message-to-and-from-xml.html), Author: poster, May 2022.

[24] flexera, «SAML Overview», 2023. [Ηλεκτρονικό]. Available: https://docs.flexera.com/flexera/EN/Administration/SAMLConcepts.htm), Author: Flexera Documentation,2022.

[25] LDAPCON Brussels, «Understanding the main SSO protocols: CAS, SAML and OpenID Connect», 2023. [Ηλεκτρονικό]. Available: https://ldapcon.org/2017/wp-content/uploads/2017/08/16_Cl%C3%A9ment-Oudot_PRE_LDAPCon2017_SSO-1.pdf), Author: Clement Oudot, 2017.

[26] Yale University, «Yale API Portal - CAS – Central Authentication Service», 2023. [Ηλεκτρονικό]. Available: https://developers.yale.edu/cas-central-authentication-service), Author: Documentation, 2019.

[27] Harvard University, «Authentication How-To Guide: CAS Integration », 2023. [Ηλεκτρονικό]. Available: https://www.iam.harvard.edu/resources/cas-integration), Author: HUIT, 2020.

[28] One Identity, «SAML Explained in Plain English », 2023. [Ηλεκτρονικό]. Available: https://www.onelogin.com/learn/saml), Author: onelogin, 2020.

[29] VARONIS, «What is SAML and how does it work? », 2023. [Ηλεκτρονικό]. Available: https://www.varonis.com/blog/what-is-saml), Author: Michael Buckbee, 2022.

[30] Github, «GlassFish », 2023. [Ηλεκτρονικό]. Available: https://javaee.github.io/glassfish/), Author: Github, 2019.

[31] Wikipedia, «GlassFish », 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/GlassFish), Author: Wikipedia, 2019.

[32] Khan Academy, «Digital certificates and CAS », 2023. [Ηλεκτρονικό]. Available: https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:secure-internet-protocols/a/digital-public-key-certificates), Author: Khan Academy, 2019.

[33] Wikipedia, «Certificate Authority », 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Certificate_authority), Author: Wikipedia, 2017.

[34] Wikipedia, «Netbeans», 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/NetBeans), Author: Wikipedia, 2023.

[35] Wikipedia, «Remmina», 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Remmina), Author: Wikipedia, 2023.

[36] SDXCENTRAL, «Client-Server Model», 2023. [Ηλεκτρονικό]. Available: https://www.sdxcentral.com/resources/glossary/client-server-model/), Author: sdx central, 2023.

[37] Wikipedia, «Client-server model», 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Client%E2%80%93server_model), Author: Wikipedia, 2023.