Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διαχείριση χρηστών με το σύστημα Enterprise Single Sign-On – CAS

Φοιτητής:
Χρήστος Λορδόπουλος
Αριθμός Μητρώου: 174950

Επιβλέπων:
Αντώνης Σιδηρόπουλος

10 Σεπτεμβρίου 2024

Τίτλος Π.Ε. Διαχείριση χρηστών με το σύστημα Enterprise Single Sign-On – CAS
Κωδικός Π.Ε. 22298
Ονοματεπώνυμο φοιτητή/τών Χρήστος Λορδόπουλος
Ονοματεπώνυμο εισηγητή Αντώνης Σιδηρόπουλος
Ημερομηνία ανάληψης Π.Ε. 26-10-2022
Ημερομηνία περάτωσης Π.Ε. 10-09-2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Το Τμήμα Πληροφορικής και Ηλεκτρονικών Συστημάτων προσφέρει ένα ευρύ φάσμα ηλεκτρονικών υπηρεσιών, οι οποίες συχνά αποτελούν μέρος των πτυχιακών εργασιών των φοιτητών του. Κρίσιμος παράγοντας για τη σωστή λειτουργία και την ασφάλεια αυτών των εφαρμογών και των χρηστών τους, καθώς και για την ευχρηστία τους, είναι η ύπαρξη ενός καλά δομημένου συστήματος Single Sign-On (SSO). Στο πλαίσιο αυτό, η παρούσα εργασία έχει ως στόχο τη διερεύνηση εναλλακτικών λύσεων για την υλοποίηση του SSO του τμήματος, με σκοπό τον εκσυγχρονισμό, την ευκολότερη συντήρηση και τη μελλοντική επέκταση του συστήματος.

Abstract

This thesis explores the use of Central Authentication Service (CAS) as a Single Sign-On (SSO) solution in the Department of Computer Engineering and Electronic Systems. CAS allows users to log into multiple services with a single set of credentials, enhancing security and user experience.
The thesis's main objectives include evaluating the existing authentication system, investigating the features of CAS, and integrating it with the department's existing applications. Our work covers the installation, configuration, and performance evaluation of CAS and provides recommendations for future improvements.
Overall, we propose CAS as a robust and scalable solution for implementing a modern and secure SSO system, fulfilling the department's current requirements.

Περίληψη

Η εργασία αυτή διερευνά τη χρήση του Central Authentication Service (CAS) ως λύση Single Sign-On (SSO) για το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Το CAS επιτρέπει στους χρήστες να συνδέονται σε πολλές υπηρεσίες με ένα μόνο σύνολο διαπιστευτηρίων, βελτιώνοντας την ασφάλεια και την εμπειρία των χρηστών.
Οι κύριοι στόχοι της εργασίας περιλαμβάνουν την αξιολόγηση του υπάρχοντος συστήματος αυθεντικοποίησης, τη διερεύνηση των χαρακτηριστικών του CAS, και την ενσωμάτωσή του με τις υπάρχουσες εφαρμογές του τμήματος. Η εργασία καλύπτει τη διαδικασία εγκατάστασης, παραμετροποίησης και αξιολόγησης της απόδοσης του CAS, ενώ προσφέρει και συστάσεις για μελλοντικές βελτιώσεις.
Συνολικά, η εργασία προτείνει το CAS ως μια ισχυρή και επεκτάσιμη λύση για την υλοποίηση ενός σύγχρονου και ασφαλούς συστήματος SSO, εκπληρώνοντας τις υφιστάμενες απαιτήσεις του τμήματος.

Ευχαριστίες

Θα ήθελα να εκφράσω τις εγκάρδιες μου ευχαριστίες στους ανθρώπους που με στήριξαν κατά την διάρκεια της εκπόνησης αυτής της εργασίας. Συγκεκριμένα, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για όλα όσα έχουν κάνει για εμένα, όλα αυτά τα χρόνια.
Και κυρίως τους γονείς μου, Ιάκωβο και Σοφία, στους οποίους οφείλω πολλά περισσότερα από όσα λόγια ή πράξεις θα μπορούσαν ποτέ να αποδώσουν.
Επίσης, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Αντώνη Σιδηρόπουλο για την υπομονή, την εμπιστοσύνη και την καθοδήγηση που μου προσέφερε για την ολοκλήρωση της παρούσας εργασίας.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Καθώς τα εκπαιδευτικά ιδρύματα επεκτείνουν τις ψηφιακές τους εφαρμογές και υπηρεσίες, η ανάγκη για αποτελεσματικά και ασφαλή συστήματα αυθεντικοποίησης είναι επιτακτική. Οι τρέχουσες απαιτήσεις πολλαπλών συνδέσεων σε διάφορες εφαρμογές δημιουργούν μια αργή και κουραστική εμπειρία για τους χρήστες, οδηγώντας σε αναποτελεσματικότητα και πιθανά προβλήματα ασφαλείας. Το Central Authentication Service (CAS), που αποτελεί λογισμικό ανοικτού κώδικα, παρουσιάζει μια ευέλικτη λύση σε αυτές τις προκλήσεις μέσω του Single Sign-On (SSO). Επιτρέποντας στους χρήστες να έχουν πρόσβαση σε πολλές υπηρεσίες με ένα μόνο σετ username και password, το CAS απλοποιεί την εμπειρία των χρηστών και προσφέρει μεγαλύτερη ασφάλεια.
Αυτή η εργασία εξετάζει τη δυνατότητα υιοθέτησης του CAS ως λύση SSO για το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων, με στόχο την απλοποίηση της πρόσβασης σε ψηφιακούς πόρους και την αντιμετώπιση των υφιστάμενων αδυναμιών του συστήματος.

1.1 Στόχοι της Εργασίας και Περιορισμοί

Ο πρωταρχικός στόχος αυτής της εργασίας είναι να διερευνήσει τη σκοπιμότητα και τα οφέλη της υιοθέτησης του CAS ως λύση SSO της Σχολής. Αυτό περιλαμβάνει μια εις βάθος ανάλυση της αρχιτεκτονικής, των χαρακτηριστικών και της συμβατότητας του CAS με το υπάρχον σύστημα. Η εργασία στοχεύει στη δημιουργία μιας λειτουργικής έκδοσης του CAS που ικανοποιεί τις κύριες απαιτήσεις του τρέχοντος συστήματος αυθεντικοποίησης.
Οι ειδικοί στόχοι περιλαμβάνουν:

- Αξιολόγηση του τρέχοντος συστήματος αυθεντικοποίησης και εντοπισμός των περιορισμών του.
- Διερεύνηση της αρχιτεκτονικής και των λειτουργιών του CAS.
- Ενσωμάτωση του CAS με τις υπάρχουσες εφαρμογές και πλατφόρμες της σχολής.
- Παροχή συστάσεων για μία πλήρη ανάπτυξη και μελλοντικές βελτιώσεις.

Ωστόσο, η εργασία υπόκειται σε αρκετούς περιορισμούς. Ο πιο σημαντικός από αυτούς είναι ο χρόνος, ο οποίος περιορίζει το εύρος των δοκιμών και των χαρακτηριστικών που μπορούν να διερευνηθούν σε βάθος. Παρότι το CAS προσφέρει ένα ευρύ φάσμα λειτουργιών, δεν μπορούν να εφαρμοστούν ή να δοκιμαστούν όλες κατά τη διάρκεια αυτού του έργου. Ως εκ τούτου, το επίκεντρο θα είναι τα βασικά χαρακτηριστικά που καλύπτουν τις άμεσες ανάγκες του τμήματος, με συστάσεις για περαιτέρω εξερεύνηση και εφαρμογή πρόσθετων λειτουργιών στο μέλλον.

1.2 Δομή της Εργασίας

Η παρούσα εργασία αποτελείται από πολλά κεφάλαια, καθένα από τα οποία εξετάζει μια βασική πτυχή της:

- **Τεχνικό Πλαίσιο:** Αυτό το κεφάλαιο παρουσιάζει μια λεπτομερή εξέταση των τεχνολογιών SSO, συμπεριλαμβανομένων των πρωτοκόλλων αυθεντικοποίησης όπως τα SAML, OAuth 2.0 και OpenID Connect. Εξηγεί επίσης την αρχιτεκτονική του CAS και τα βασικά στοιχεία του.
- **Επισκόπηση Λύσεων Single Sign-On (SSO):** Αυτό το κεφάλαιο παρέχει μια επισκόπηση των λύσεων Single Sign-On (SSO). Ενώ παρουσιάζει το τρέχον σύστημα αυθεντικοποίησης του τμήματος καθώς και τις απαιτήσεις για τη νέα λύση SSO.
- **Εγκατάσταση και Παραμετροποίηση του CAS:** Αυτό το κεφάλαιο παρέχει έναν περιεκτικό οδηγό για την εγκατάσταση του CAS, συμπεριλαμβανομένων των προαπαιτούμενων, της ρύθμισης του περιβάλλοντος και της μετέπειτα διαμόρφωσης.
- **Συμπέρασμα και Συστάσεις:** Αυτό το κεφάλαιο συνοψίζει τα ευρήματα, παρέχει συστάσεις και προτείνει τομείς για μελλοντική έρευνα και ανάπτυξη.

Κεφάλαιο 2

Τεχνικό Πλαίσιο

2.1 Τεχνολογίες Single Sign-On (SSO)

Η ταχύρρυθμη ανάπτυξη του διαδικτύου και των υπηρεσιών που προσφέρονται μέσω αυτού, έχουν δημιουργήσει την ανάγκη για αποδοτικές και ασφαλείς λύσεις αυθεντικοποίησης. Για την πρόσβαση των χρηστών σε πλήθος εφαρμογών και υπηρεσιών, πολλές φορές απαιτούνται διαφορετικά διαπιστευτήρια (όνομα χρήστη και κωδικός πρόσβασης). Αυτή η διαδικασία μπορεί να είναι χρονοβόρα και συχνά προκαλεί σύγχυση και προβλήματα ασφαλείας. Οι τεχνολογίες Ενιαίας Σύνδεσης (Single Sign-On - SSO) προσφέρουν μια λύση σε αυτά τα προβλήματα, επιτρέποντας στους χρήστες να αποκτούν πρόσβαση σε πολλαπλές υπηρεσίες με ένα μόνο σύνολο διαπιστευτηρίων [1]. Δηλαδή κατά την είσοδο του στο σύστημα ο χρήστης χρειάζεται να αποδείξει την ταυτότητά του μόνο μία φορά στην κεντρική υπηρεσία, ενώ οι υπόλοιπες εφαρμογές θα διασταυρώσουν τα στοιχεία του μέσω αυτής. Δεν θα υπάρξουν επομένως πολλαπλές μεταφορές των κωδικών του χρήστη. Αυτό αρχικά έχει ως αποτέλεσμα την βελτίωση της εμπειρίας του χρήστη, διότι τον απαλλάσσει από την ανάγκη να θυμάται πολλαπλούς κωδικούς. Αλλά κυρίως συμβάλλει στην ενίσχυση της ασφάλειας, μειώνοντας τον κίνδυνο υποκλοπής κωδικών και άλλων ειδών απάτης.

2.1.1 Πρωτόκολλα Αυθεντικοποίησης

Για την υλοποίηση του SSO είναι αναγκαία η χρήση ενός πρωτοκόλλου αυθεντικοποίησης, το οποίο καθορίζει τον τρόπο με τον οποίο επιτυγχάνεται η επαλήθευση της ταυτότητας των χρηστών και η ασφαλής πρόσβαση σε ψηφιακούς πόρους. Στο πλαίσιο των τεχνολογιών SSO, πρωτόκολλα όπως το SAML, το OAuth, το OpenID Connect και το Kerberos παρέχουν διαφορετικές προσεγγίσεις για την επίτευξη αυθεντικοποίησης και εξουσιοδότησης, αντιμετωπίζοντας ποικίλες ανάγκες και απαιτήσεις ασφαλείας.

Kerberos

Το πρωτόκολλο Kerberos, το οποίο αναπτύχθηκε από το Τεχνολογικό Ινστιτούτο Μασαχουσέτης (MIT), είναι ένα από τα πιο διαδεδομένα και αξιόπιστα πρωτόκολλα αυθεντικοποίησης, ειδικά σε περιβάλλοντα δικτύων. Το Kerberos χρησιμοποιείται ευρέως σε εταιρικά δίκτυα για την εξασφάλιση ασφαλούς και αποδοτικής επικοινωνίας μεταξύ πελατών και υπηρεσιών[2].

Το Kerberos λειτουργεί με βάση ένα σύστημα ”εισιτηρίων” (tickets) για την παροχή ασφαλούς αυθεντικοποίησης. Ακολουθεί μια συνοπτική περιγραφή της διαδικασίας:

1. **Αίτηση Αρχικού Εισιτηρίου (TGT):** Ο χρήστης εισάγει τα διαπιστευτήριά του (π.χ. όνομα χρήστη και κωδικό πρόσβασης) και ζητά ένα Ticket Granting Ticket (TGT) από τον Κέντρο Διανομής Κλειδιών (Key Distribution Center - KDC).

2. **Έκδοση TGT:** Το KDC επαληθεύει τα διαπιστευτήρια και εκδίδει ένα κρυπτογραφημένο TGT, το οποίο αποστέλλεται στον χρήστη.

3. **Αίτηση Υπηρεσιακού Εισιτηρίου (ST):** Ο χρήστης χρησιμοποιεί το TGT για να ζητήσει ένα εισιτήριο πρόσβασης σε μια συγκεκριμένη υπηρεσία (Service Ticket) από το KDC.

4. **Έκδοση Υπηρεσιακού Εισιτηρίου:** Το KDC εκδίδει ένα υπηρεσιακό εισιτήριο (Service Ticket) το οποίο επιτρέπει στον χρήστη να έχει πρόσβαση στην υπηρεσία.

5. **Πρόσβαση στην Υπηρεσία:** Ο χρήστης παρουσιάζει το υπηρεσιακό εισιτήριο στην υπηρεσία, η οποία το επαληθεύει και επιτρέπει την πρόσβαση.

SAML (Security Assertion Markup Language)

Το SAML (Security Assertion Markup Language) είναι ένα πρωτόκολλο που χρησιμοποιείται για την ανταλλαγή δεδομένων αυθεντικοποίησης μεταξύ διαφορετικών πλευρών, όπως παρόχων ταυτότητας (Identity Providers - IdP) και παρόχων υπηρεσιών (Service Providers - SP). Αναπτύχθηκε από την OASIS (Organization for the Advancement of Structured Information Standards) και έχει υιοθετηθεί ευρέως σε εταιρικά και εκπαιδευτικά περιβάλλοντα για την υλοποίηση λύσεων Ενιαίας Σύνδεσης (Single Sign-On - SSO)[3]. Το SAML βασίζεται σε τρία κύρια στοιχεία:

- **Principal (Χρήστης):** Ο χρήστης που προσπαθεί να αποκτήσει πρόσβαση σε μια υπηρεσία.

- **Identity Provider (IdP):** Ο πάροχος ταυτότητας που επαληθεύει τα διαπιστευτήρια του χρήστη και εκδίδει δηλώσεις ταυτότητας.

- **Service Provider (SP):** Ο πάροχος υπηρεσιών που προσφέρει την εφαρμογή ή την υπηρεσία στην οποία ο χρήστης προσπαθεί να έχει πρόσβαση.

Η διαδικασία αυθεντικοποίησης με το SAML περιλαμβάνει τα εξής βήματα:

1. **Αίτηση Αυθεντικοποίησης:** Ο χρήστης προσπαθεί να προσπελάσει μια υπηρεσία από τον SP. Ο SP ανακατευθύνει τον χρήστη στον IdP για αυθεντικοποίηση.

2. **Αυθεντικοποίηση:** Ο IdP ζητά από τον χρήστη να εισάγει τα διαπιστευτήριά του (π.χ., όνομα χρήστη και κωδικό πρόσβασης). Εάν τα διαπιστευτήρια είναι έγκυρα, ο IdP δημιουργεί μια δήλωση αυθεντικοποίησης.

3. **Έκδοση Δήλωσης:** Η δήλωση αυθεντικοποίησης περιλαμβάνει πληροφορίες σχετικά με την ταυτότητα του χρήστη και άλλα δεδομένα εξουσιοδότησης. Αυτή η δήλωση είναι κρυπτογραφημένη και υπογεγραμμένη από τον IdP.

4. **Παράδοση Δήλωσης:** Η δήλωση αποστέλλεται στον SP, συνήθως μέσω του χρήστη (π.χ., μέσω ανακατεύθυνσης του προγράμματος περιήγησης).

5. **Επαλήθευση και Παροχή Πρόσβασης:** Ο SP επαληθεύει τη δήλωση και, εάν είναι έγκυρη, επιτρέπει στον χρήστη να αποκτήσει πρόσβαση στην υπηρεσία.

OAuth 2.0 (Open Authorization)

Το OAuth 2.0 είναι ένα πρωτόκολλο εξουσιοδότησης που επιτρέπει σε εφαρμογές τρίτων να αποκτούν περιορισμένη πρόσβαση σε πόρους ενός χρήστη χωρίς να χρειάζεται να μοιραστούν τον κωδικό πρόσβασης του χρήστη. Παρόλα αυτά, συχνά επιτελεί και τον ρόλο του πρωτοκόλλου αυθεντικοποίησης, ειδικά όταν συνδυάζεται με εφαρμογές κοινωνικών δικτύων (π.χ. Facebook, LinkedIn, Github κ.α). Αναπτύχθηκε ως διάδοχος του OAuth 1.0 και προσφέρει βελτιωμένη ευελιξία και ασφάλεια. Χρησιμοποιείται ευρέως σε εφαρμογές ιστού, κινητών και API για την παροχή εξουσιοδότησης μέσω μιας διαδικασίας που βασίζεται σε tokens[4]. Το OAuth 2.0 περιλαμβάνει τα εξής βασικά στοιχεία:

- **Resource Owner (Ιδιοκτήτης Πόρου):** Ο χρήστης που ελέγχει την πρόσβαση στους πόρους.
- **Client (Πελάτης):** Η εφαρμογή που ζητά πρόσβαση στους πόρους του χρήστη.
- **Authorization Server (Διακομιστής Εξουσιοδότησης):** Ο διακομιστής που εκδίδει τα tokens πρόσβασης μετά την επαλήθευση της ταυτότητας του χρήστη και της εξουσιοδότησης.
- **Resource Server (Διακομιστής Πόρων):** Ο διακομιστής που φιλοξενεί τους προστατευόμενους πόρους και δέχεται τα tokens πρόσβασης.

Η διαδικασία εξουσιοδότησης στο OAuth 2.0 περιλαμβάνει διάφορα βήματα και μπορεί να διαφοροποιείται ανάλογα με το grant type που χρησιμοποιείται, όπως τα Authorization Code, PKCE, Implicit, Device Code, Refresh Token και Client Credentials. Παρακάτω περιγράφεται η διαδικασία χρησιμοποιώντας το Authorization Code flow:

1. **Αίτηση Εξουσιοδότησης:** Ο χρήστης ξεκινά τη διαδικασία εξουσιοδότησης μέσω του client, ο οποίος ανακατευθύνει τον χρήστη στον διακομιστή εξουσιοδότησης.

2. **Συγκατάθεση Χρήστη:** Ο διακομιστής εξουσιοδότησης ζητά από τον χρήστη να παράσχει συγκατάθεση για την πρόσβαση στους πόρους. Εάν ο χρήστης συμφωνήσει, εκδίδεται ένας κωδικός εξουσιοδότησης (authorization code).

3. **Ανταλλαγή Κωδικού με Token:** Ο πελάτης χρησιμοποιεί τον κωδικό εξουσιοδότησης για να ζητήσει ένα token πρόσβασης (access token) από τον διακομιστή εξουσιοδότησης.

4. **Έκδοση Token:** Ο διακομιστής εξουσιοδότησης επαληθεύει τον κωδικό και εκδίδει ένα token πρόσβασης (και ενδεχομένως ένα refresh token) στον πελάτη.

5. **Πρόσβαση στους Πόρους:** Ο πελάτης χρησιμοποιεί το token πρόσβασης για να κάνει αιτήσεις στον διακομιστή πόρων, αποκτώντας πρόσβαση στους προστατευόμενους πόρους.

OpenID Connect

Το OpenID Connect είναι ένα πρωτόκολλο αυθεντικοποίησης που βασίζεται στο OAuth 2.0 και προσθέτει λειτουργίες ταυτοποίησης χρηστών στις δυνατότητες εξουσιοδότησης του OAuth 2.0. Αναπτύχθηκε από το OpenID Foundation και έχει υιοθετηθεί ευρέως για τη διασφάλιση της ταυτότητας των χρηστών και την παροχή ασφαλούς πρόσβασης σε εφαρμογές και υπηρεσίες στο διαδίκτυο[5].
Το OpenID Connect επιτρέπει στις εφαρμογές (που ονομάζονται Relying Parties - RP) να επαληθεύουν την ταυτότητα των χρηστών μέσω ενός τρίτου παρόχου ταυτότητας (Identity Provider - IdP) και να αποκτούν βασικές πληροφορίες γισ το προφίλ του χρήστη. Η διαδικασία περιλαμβάνει τα εξής βήματα:

1. **Αίτηση Αυθεντικοποίησης:** Ο χρήστης προσπαθεί να προσπελάσει μια υπηρεσία από την εφαρμογή RP. Η εφαρμογή ανακατευθύνει τον χρήστη στον IdP για αυθεντικοποίηση.

2. **Αυθεντικοποίηση από τον IdP:** Ο IdP ζητά από τον χρήστη να εισαγάγει τα διαπιστευτήριά του (π.χ. όνομα χρήστη και κωδικό πρόσβασης). Εάν τα διαπιστευτήρια είναι έγκυρα, ο IdP δημιουργεί ένα Access Token και ένα ID Token.

3. **Έκδοση των Tokens:** Το Access Token και το ID Token αποστέλλονται στην εφαρμογή RP μέσω του προγράμματος περιήγησης του χρήστη ή απευθείας.

4. **Επαλήθευση του ID Token:** Η εφαρμογή RP επαληθεύει το ID Token για να βεβαιωθεί ότι ο χρήστης έχει αυθεντικοποιηθεί επιτυχώς και να αποκτήσει τις πληροφορίες για το προφίλ του χρήστη.

5. **Παροχή Πρόσβασης:** Η εφαρμογή RP επιτρέπει στον χρήστη να αποκτήσει πρόσβαση στις προστατευμένες υπηρεσίες βάσει των πληροφοριών που παρέχονται στο ID Token.

Επίλογος

Το Kerberos, το SAML, το OAuth 2.0 και το OpenID Connect (OIDC) είναι σημαντικά πρωτόκολλα Single Sign-On (SSO), το καθένα με ξεχωριστές χρήσεις, πλεονεκτήματα και μειονεκτήματα. Το Kerberos χρησιμοποιείται ευρέως από τη Microsoft για την αυθεντικοποίηση των χρηστών της αλλά και στις υπηρεσίες του Active Directory. Αν και είναι γνωστό για την ανθεκτικότητά του και την αμοιβαία πιστοποίηση σε εσωτερικά δίκτυα, είναι περίπλοκο στην υλοποίηση και την διαχείριση του. Το SAML (Security Assertion Markup Language) χρησιμοποιείται ευρέως για SSO σε επιχειρήσεις, με εταιρείες όπως η Google να το χρησιμοποιούν για την πιστοποίηση web εφαρμογών, παρά την δυσκολία διαχείρισης των μεγάλων φορτίων XML που απαιτεί. Το OAuth 2.0, που χρησιμοποιείται από γίγαντες όπως το Facebook και το GitHub, χρησιμοποιείται για την παροχή πρόσβασης, σε τρίτες εφαρμογές, στους πόρους των χρηστών χωρίς την αποκάλυψη των διαπιστευτηρίων τους. Ωστόσο απαιτεί προσεκτική υλοποίηση για την αποφυγή κενών ασφάλειας. Το OIDC προσθέτει ένα επίπεδο ταυτότητας πάνω στο OAuth 2.0, παρέχοντας απλούστερη και πιο ασφαλή πιστοποίηση χρηστών για εφαρμογές web και κινητών. Ενώ έχει τα ίδια πιθανά προβλήματα με το OAuth 2.0.

2.1.2 JSON Web Tokens (JWTs)

Μια τεχνολογία που συχνά χρησιμοποιείται σε συνδυασμό με τα πρωτόκολλα αυθεντικοποίησης είναι τα JSON Web Tokens (JWTs). Τα JWTs είναι ένα ανοιχτό πρότυπο (RFC 7519) που καθορίζει έναν συμπαγή και αυτοδύναμο τρόπο για τη μεταφορά πληροφοριών μεταξύ δύο μερών με τη μορφή ενός JSON αντικειμένου[6]. Αυτή η μεταφορά μπορεί να είναι υπογεγραμμένη, ώστε να εξασφαλίζεται η αυθεντικότητα των δεδομένων, και κρυπτογραφημένη για την προστασία της εμπιστευτικότητας.
Τα JWTs χρησιμοποιούνται συχνά σε σενάρια SSO για την αποθήκευση και μεταφορά των claims που περιγράφουν την ταυτότητα του χρήστη και τα δικαιώματα που του έχουν ανατεθεί. Λόγω της ευελιξίας και της ευκολίας με την οποία μπορούν να ενσωματωθούν σε υπάρχουσες υποδομές, τα JWTs έχουν γίνει ιδιαίτερα δημοφιλή στις σύγχρονες εφαρμογές αυθεντικοποίησης και εξουσιοδότησης.

2.2 Αρχιτεκτονική του CAS

Το Central Authentication Service (CAS) είναι ένα πρωτόκολλο και πλατφόρμα ανοικτού κώδικα που αναπτύχθηκε από το Πανεπιστήμιο Yale και υποστηρίζεται από το Apereo Foundation. Σχεδιάστηκε για να παρέχει μια ενιαία, κεντρική υπηρεσία αυθεντικοποίησης (Single Sign-On - SSO) για διαδικτυακές εφαρμογές.
Η υιοθέτηση του CAS μπορεί να προσφέρει σημαντικά οφέλη στα εκπαιδευτικά ιδρύματα και τους οργανισμούς, συμπεριλαμβανομένης της μείωσης των λειτουργικών εξόδων, της βελτίωσης της ασφάλειας και της παροχής μιας πιο συνεκτικής και αποδοτικής εμπειρίας χρήστη.

2.2.1 Κύρια Μέρη

Η αρχιτεκτονική του CAS αποτελείται από δύο κύρια μέρη που συνεργάζονται για να επιτύχουν την ασφαλή διαχείριση ταυτοτήτων και την επαλήθευση των χρηστών[7].
Τον **CAS Server (Διακομιστής CAS):** Ο CAS Server είναι η καρδιά του συστήματος και είναι υπεύθυνος για την αυθεντικοποίηση των χρηστών, αλλά και για την έγκριση πρόσβασης σε προστατευόμενες από το CAS υπηρεσίες. Είναι γραμμένος στη γλώσσα προγραμματισμού Java και βασίζεται στο Spring Framework. Μία SSO session δημιουργείται όταν ο διακομιστής εκδίδει ένα ticket-granting ticket (TGT) στον χρήστη μετά την επιτυχή σύνδεση. Ένα service ticket (ST) δημιουργείται για μια υπηρεσία, κατόπιν αιτήματος του χρήστη, χρησιμοποιώντας το TGT ως token. Το ST επικυρώνεται στη συνέχεια από τον διακομιστή CAS μέσω επικοινωνίας back-channel.
Και τους **CAS Clients (Πελάτες CAS):** Οι CAS Clients αποτελούν οποιεσδήποτε εφαρμογές ή οι υπηρεσίες μπορούν να επικοινωνήσουν με τον CAS server μέσω ενός υποστηριζόμενου πρωτοκόλλου (π.χ. CAS, SAML, OAuth 2.0). Οι CAS clients μπορούν να υλοποιηθούν σε διάφορες γλώσσες προγραμματισμού και πλατφόρμες μέσω κατάλληλων βιβλιοθηκών και εργαλείων.

2.2.2 Ανάλυση Λογισμικού

Ο CAS server μπορεί να διαχωριστεί σε τρία διακριτά υποσυστήματα, ως εξής:

- **Web (Spring MVC/Spring Webflow):** Το Web tier είναι το σημείο επικοινωνίας με όλα τα εξωτερικά συστήματα, συμπεριλαμβανομένων των CAS clients. Αυτό το επίπεδο είναι υπεύθυνο για την αρχική επαλήθευση και την ανακατεύθυνση των χρηστών προς τα κατάλληλα υποσυστήματα για περαιτέρω επεξεργασία.
- **Ticketing:** Το υποσύστημα έκδοσης εισιτηρίων δημιουργεί τα εισιτήρια για την πρόσβαση των CAS clients. Η SSO session ξεκινά με την έκδοση ενός ticket-granting ticket (TGT) μετά από επιτυχή αυθεντικοποίηση. Το υποσύστημα έκδοσης εισιτηρίων συχνά συνεργάζεται με το υποσύστημα αυθεντικοποίησης για την επαλήθευση των διαπιστευτηρίων του χρήστη.
- **Authentication:** Το σύστημα αυθεντικοποίησης επεξεργάζεται τις αιτήσεις κατά την έναρξη της SSO session, αν και μπορεί να ενεργοποιηθεί και σε άλλες περιπτώσεις (π.χ. forced authentication).

Spring Framework

Το CAS χρησιμοποιεί πολλές πτυχές του Spring Framework, κυρίως το Spring MVC και το Spring Webflow. Το Spring παρέχει ένα ολοκληρωμένο και επεκτάσιμο πλαίσιο για τον βασικό κώδικα του CAS[8]. Η γενική γνώση του Spring είναι επωφελής για την κατανόηση της αλληλεπίδρασης μεταξύ των στοιχείων του πλαισίου, αν και δεν είναι απολύτως απαραίτητη.

Spring Boot

Το CAS βασίζεται επίσης σε μεγάλο βαθμό στο Spring Boot, το οποίο του επιτρέπει τη δημιουργία μιας αυτόνομης διαδικτυακής εφαρμογής χωρίς την ανάγκη για εκτεταμένη ρύθμιση μέσω XML[9]. Το Spring Boot επιτρέπει στο CAS να αποκρύπτει μεγάλο μέρος της εσωτερικής του πολυπλοκότητας, απλοποιώντας παράλληλα την διαδικασία προσαρμογής των λειτουργιών του.

Κεφάλαιο 3

Επισκόπηση λύσεων Single Sign-On (SSO)

3.1 Υπάρχον Σύστημα

Το υπάρχον σύστημα, το οποίο υλοποιεί το Single Sign-On, αποτελεί έργο φοιτητών του τμήματος ως μέρος των πτυχιακών τους εργασιών. Η λειτουργικότητα του SSO επιτυγχάνεται μέσω ενός RESTful API, το οποίο χρησιμοποιεί το πρωτόκολλο OAuth 2.0 για την προστασία και την εξουσιοδότηση των εφαρμογών, και ενός LDAP directory από όπου αντλούνται οι πληροφορίες για την αυθεντικοποίηση των χρηστών.
Παρότι το σύστημα είναι πλήρως λειτουργικό και καλύπτει τις ανάγκες της σχολής αυτή την δεδομένη στιγμή, η συντήρηση και η περαιτέρω ανάπτυξη του κώδικα καθιστούν πολύ δύσκολο έργο.
Για τον λόγο αυτόν πάρθηκε η απόφαση για τη διερεύνηση πιθανών εναλλακτικών, οι οποίες όχι μόνο θα εξυπηρετούν τις ήδη υπάρχουσες ανάγκες αλλά θα εγγυούνται και τη μελλοντική συντήρηση και επέκταση των τεχνολογιών που χρησιμοποιούν.

3.1.1 Απαιτήσεις Νέου Συστήματος

Κατά τη διαδικασία επιλογής ενός νέου συστήματος SSO, είναι σημαντικό να εξεταστούν βασικά χαρακτηριστικά όπως: τα υποστηριζόμενα πρωτόκολλα και οι Identity Providers, η ευκολία ενσωμάτωσης, οι δυνατότητες κλιμάκωσης και υψηλής διαθεσιμότητας, καθώς και η συχνότητα ενημερώσεων του λογισμικού. Επιπλέον, θα πρέπει να δοθεί έμφαση στην ύπαρξη καλού documentation και μιας ενεργής κοινότητας.
Συγκεκριμένα, το νέο σύστημα θα πρέπει να υποστηρίζει το πρωτόκολλο OAuth 2.0 και να επιτρέπει την ενσωμάτωσή του με το ήδη υπάρχον LDAP directory για την αυθεντικοποίηση των χρηστών. Επιπλέον, είναι σημαντικό να διασφαλίζεται ότι το σύστημα μπορεί να ανταποκριθεί σε απαιτήσεις υψηλής διαθεσιμότητας, με την εφαρμογή μηχανισμών αποθήκευσης δεδομένων (persistence) μέσω μιας αξιόπιστης βάσης δεδομένων, όπως η MongoDB.
Ιδανικά, το σύστημα θα πρέπει να προσφέρει δυνατότητες διαχείρισης χρηστών, επιτρέποντας τη διαχείριση δικαιωμάτων πρόσβασης, τον καθορισμό ρόλων και την παρακολούθηση της δραστηριότητας των χρηστών.
Παράλληλα, πιο γενικές αλλά κρίσιμες παράμετροι όπως η ασφάλεια, η επεκτασιμότητα, η αξιοπιστία και η απόδοση θα πρέπει να αποτελούν θεμελιώδη χαρακτηριστικά του συστήματος. Η διασφάλιση της ασφάλειας περιλαμβάνει την προστασία των δεδομένων από μη εξου-

σιοδοτημένη πρόσβαση και επιθέσεις, ενώ η επεκτασιμότητα αναφέρεται στη δυνατότητα του συστήματος να προσαρμόζεται σε αυξανόμενες ανάγκες χωρίς συμβιβασμούς στην απόδοση ή στην αξιοπιστία.

3.2 Shibboleth

Το Shibboleth είναι ένα ευρέως υιοθετημένο πρότζεκτ ανοιχτού κώδικα που παρέχει δυνατότητες Single Sign-On (SSO) και identity-based αυθεντικοποίησης, ιδιαίτερα κατάλληλο για εκπαιδευτικά ιδρύματα και ερευνητικούς οργανισμούς. Αναπτύσσεται και διατηρείται από το Shibboleth Consortium και χρησιμοποιεί την Security Assertion Markup Language (SAML) για να επιτρέψει την ταυτοποίηση μεταξύ διαφορετικών οργανισμών και εφαρμογών.
Στον πυρήνα του, το Shibboleth αποτελείται από δύο βασικά συστατικά: τον Identity Provider (IdP) και τον Service Provider (SP). Ο IdP αυθεντικοποιεί τους χρήστες και εκδίδει SAML assertions σχετικά με την ταυτότητά τους, ενώ ο SP χρησιμοποιεί αυτές τις δηλώσεις για να παρέχει πρόσβαση σε πόρους. Αυτή η δομή επιτρέπει έναν ευέλικτο και κλιμακούμενο μηχανισμό αυθεντικοποίησης, όπου οι χρήστες μπορούν να συνδέονται μία φορά και να αποκτούν πρόσβαση σε πολλές εφαρμογές χωρίς να χρειάζεται να επιβεβαιώνουν ξανά την ταυτότητά τους. Οι εκτεταμένες δυνατότητες προσαρμογής του Shibboleth και η υποστήριξη για πολιτικές attribute release παρέχουν λεπτομερή έλεγχο των πληροφοριών που κοινοποιούνται, ενισχύοντας την ασφάλεια[10].
Τεχνολογικά, το Shibboleth είναι χτισμένο χρησιμοποιώντας Java και απαιτεί ένα container servlet όπως το Apache Tomcat για την υλοποίηση του. Χρησιμοποιεί αρχεία διαμόρφωσης βασισμένα σε XML, τα οποία, αν και ισχυρά, είναι περίπλοκα στην διαχείριση τους. Ο IdP του Shibboleth υποστηρίζει διάφορους μηχανισμούς αυθεντικοποίησης, συμπεριλαμβανομένων των ονόματος χρήστη/κωδικού πρόσβασης, του multi-factor authentication (MFA) και της ενσωμάτωσης με υπηρεσίες καταλόγου όπως το LDAP και το Active Directory. Επιπλέον, ο IdP μπορεί να επεκταθεί με προσαρμοσμένους χειριστές αυθεντικοποίησης για να καλύψει συγκεκριμένες απαιτήσεις του οργανισμού.
Ωστόσο, η εξάρτηση του Shibboleth από το SAML το διαχωρίζει από άλλες λύσεις SSO που υποστηρίζουν ένα ευρύτερο φάσμα πρωτοκόλλων.

3.3 Keycloak

Το Keycloak είναι μια λύση identity and access management ανοιχτού κώδικα που αναπτύχθηκε από τη Red Hat, προσφέροντας δυνατότητες ενιαίας σύνδεσης (SSO). Σχεδιασμένο για να απλοποιεί τις διαδικασίες αυθεντικοποίησης και εξουσιοδότησης για εφαρμογές και υπηρεσίες, το Keycloak υποστηρίζει διάφορα πρωτόκολλα όπως το OAuth 2.0, το OpenID Connect και το SAML, καθιστώντας το μια ευέλικτη λύση για σύγχρονες διαδικτυακές εφαρμογές[11].
Το πρόγραμμα είναι χτισμένο πάνω στον application server WildFly, που αποτελεί μέρος της κοινότητας JBoss. Είναι γραμμένο κυρίως σε Java και παρέχει ένα πλούσιο σύνολο λειτουργιών μέσω του web-based admin console του. Η κονσόλα αυτή επιτρέπει στους διαχειριστές να διαχειρίζονται χρήστες, να διαμορφώνουν παρόχους ταυτότητας και να ρυθμίζουν τις ροές αυθεντικοποίησης με ευκολία. Η αρχιτεκτονική του Keycloak υποστηρίζει clustering και υψηλή

διαθεσιμότητα, διασφαλίζοντας ότι μπορεί να κλιμακωθεί για να καλύψει τις απαιτήσεις μεγάλων και πολύπλοκων περιβαλλόντων.
Ένα από τα ξεχωριστά χαρακτηριστικά του Keycloak είναι η υποστήριξη για την σύνδεση μέσω κοινωνικών δικτύων. Οι χρήστες μπορούν να αυθεντικοποιηθούν χρησιμοποιώντας τους λογαριασμούς τους από κοινωνικά δίκτυα όπως το Google, το Facebook, το Twitter και το GitHub, καθώς και μέσω εταιρικών παρόχων ταυτότητας μέσω SAML ή LDAP. Αυτή η ευελιξία μειώνει την τριβή για τους τελικούς χρήστες, επιτρέποντάς τους να χρησιμοποιούν υπάρχοντα διαπιστευτήρια, διευκολύνοντας έτσι τη διαδικασία σύνδεσης. Το Keycloak υποστηρίζει επίσης multifactor authentication (MFA) και identity brokering, επιτρέποντας στους χρήστες να συνδέονται μέσω ενός ενδιάμεσου παρόχου ταυτότητας.

3.4 Central Authentication Service (CAS)

Παρατηρώντας τις δυνατότητες που προσφέρουν άλλες λύσεις SSO, είναι λογικό να δημιουργηθεί το ερώτημα αν και κατά πόσο καλύτερη επιλογή αποτελεί το CAS. Με την πληθώρα επιλογών στην αγορά, ο καθορισμός της βέλτιστης λύσης μπορεί να είναι μια σύνθετη διαδικασία. Τι είναι αυτό που το κάνει να ξεχωρίζει από τον ανταγωνισμό;
Η απάντηση σε αυτή την ερώτηση βρίσκεται κυρίως στον αριθμό και το είδος των τεχνολογιών που υποστηρίζει, αλλά και στο πόσο απλή είναι η ενσωμάτωσή τους. Το CAS έχει εξελιχθεί σε μεγάλο βαθμό ανά τα χρόνια, και είναι πλέον μία από τις πιο ευέλικτες και ευπροσάρμοστες λύσεις στον τομέα της αυθεντικοποίησης. Στην τωρινή του έκδοση υποστηρίζει όλες τις τεχνολογίες και τις δυνατότητες που αναφέρθηκαν στα προηγούμενα κεφάλαια (π.χ εύρος πρωτοκόλλων αυθεντικοποίησης, MFA κλπ.), επιτρέποντας στους οργανισμούς να ανταποκρίνονται σε ένα συνεχώς μεταβαλλόμενο τοπίο ασφάλειας.
Για την επιλογή ενός συστήματος SSO, ο σημαντικότερος παράγοντας που πρέπει να ληφθεί υπόψη είναι το υπάρχον περιβάλλον. Τα περισσότερα έργα δεν ξεκινούν από το μηδέν, συνήθως αξιοποιούν έναν συνδυασμό υφιστάμενων τεχνολογιών. Αυτό σημαίνει ότι η συμβατότητα και η προσαρμοστικότητα της λύσης SSO είναι κρίσιμες παράμετροι για την επιτυχία της ενσωμάτωσης. Για παράδειγμα, μια εταιρεία μπορεί να χρησιμοποιεί το Redis σε CentOS, να ενσωματώνει την MongoDB και να βασίζεται σε ένα παλαιότερο σύστημα LDAP για την αποθήκευση χρηστών. Αυτοί οι περιορισμοί απαιτούν μια λύση SSO που μπορεί να ενσωματωθεί άψογα με ποικίλες τεχνολογίες.
Ο διακομιστής CAS (Central Authentication Service) είναι εξαιρετικά προσαρμοστικός και ταιριάζει σε μια ευρεία γκάμα περιβαλλόντων. Η προσαρμοστικότητα αυτή το καθιστά ιδανικό για οργανισμούς με πολύπλοκες υποδομές, καθώς μπορεί να καλύψει τις ανάγκες τόσο μικρών όσο και μεγάλων περιβαλλόντων. Υποστηρίζει πολλές μεθόδους αυθεντικοποίησης, όπως
LDAP, βάσεις δεδομένων, X509, Cassandra, Radius, Spnego, JWT και AWS Cloud Directory. Επιπλέον, μπορεί να χειριστεί διάφορα συστήματα αποθήκευσης, όπως Memcached, Redis, Oracle, Hazelcast, CouchDB και Couchbase. Η ευρεία υποστήριξη αυτών των τεχνολογιών επιτρέπει στο CAS να λειτουργεί σε περιβάλλοντα που απαιτούν υψηλή διαθεσιμότητα και κλιμακούμενη ασφάλεια.
Το πρότζεκτ προσφέρει επίσης την εφαρμογή διαχείρισης CAS Management Webapp, η οποία είναι μια φιλική προς το χρήστη διεπαφή, σχεδιασμένη για τη διαχείριση και διαμόρφωση του CAS. Απλοποιεί τη διαχείριση του CAS παρέχοντας εργαλεία για τη διαχείριση των προστα-

τευόμενων υπηρεσιών, τη ρύθμιση πολιτικών αυθεντικοποίησης και τον χειρισμό των metadata χωρίς την ανάγκη άμεσης τροποποίησης του κώδικα. Προσφέροντας έτσι βελτιωμένη συνολική χρηστικότητα και ευελιξία στην υποδομή του CAS.
Τέλος, εξίσου σημαντική είναι και η διαλειτουργικότητα του υπάρχοντος περιβάλλοντος, καθώς έργα τα οποία επιχειρούν να ανανεώσουν εντελώς ένα σύστημα συνήθως αποτυγχάνουν λόγω των δυσκολιών ενσωμάτωσης με το υπάρχον λογισμικό. Η ικανότητα μιας λύσης να συνεργάζεται με το υπάρχον σύστημα είναι συχνά καθοριστική για την επιτυχία της εφαρμογής της. Εδώ έγκειται μία από τις βασικές δυνάμεις του CAS, ο CAS server μπορεί να λειτουργήσει ως Πάροχος Ταυτότητας SAML (IdP), πάροχος OAuth, πάροχος OpenID Connect ή ως διακομιστής CAS. Ενώ μπορεί επίσης να λειτουργήσει ως client, αναθέτοντας την αυθεντικοποίηση σε άλλον διακομιστή CAS, σε έναν IdP SAML ή σε εξωτερικούς παρόχους ταυτότητας, όπως το Facebook, το Google και το Twitter. Αυτή η ευελιξία επιτρέπει στο CAS να λειτουργεί είτε ως ανεξάρτητο σύστημα είτε ως τμήμα μιας μεγαλύτερης λύσης αυθεντικοποίησης, ανάλογα με τις ανάγκες του οργανισμού.

Κεφάλαιο 4

Εγκατάσταση και Παραμετροποίηση του CAS

Στο κεφάλαιο αυτό παρουσιάζεται λεπτομερώς η εγκατάσταση και η παραμετροποίηση του Central Authentication Service (CAS), έκδοση 7.0.0. Στόχος του κεφαλαίου είναι να καθοδηγήσει τον αναγνώστη μέσα από τα βήματα που απαιτούνται για την επιτυχή εγκατάσταση του CAS, καθώς και τις βασικές παραμετροποιήσεις που είναι απαραίτητες για την προσαρμογή του στις ανάγκες του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων.

4.1 Προαπαιτούμενα και Προετοιμασία

Στην ενότητα αυτή, εξετάζονται τα προαπαιτούμενα και οι διαδικασίες προετοιμασίας που είναι απαραίτητες για την εγκατάσταση και παραμετροποίηση του CAS. Η σωστή προετοιμασία διασφαλίζει την ομαλή και αποτελεσματική ολοκλήρωση της διαδικασίας εγκατάστασης, ελαχιστοποιώντας πιθανά προβλήματα και καθυστερήσεις.

4.1.1 Απαιτήσεις Υλικού και Λογισμικού

Το σύστημα παρουσιάζει τις εξής προαπαιτήσεις :

- **Java:** Το CAS είναι μια web εφαρμογή βασισμένη στη γλώσσα προγραμματισμού Java. Επομένως, πριν από την εγκατάσταση θα χρειαστεί να υπάρχει εγκατεστημένο το κατάλληλο JDK. Για την έκδοση του CAS που θα χρησιμοποιηθεί, απαιτείται το JDK 21.
- **Servlet Containers:** Δεν υπάρχει επίσημα υποστηριζόμενο servlet container για το CAS, ωστόσο το Apache Tomcat είναι το πιο κοινά χρησιμοποιούμενο και αυτό που επιλέχθηκε για αυτήν την εργασία.
- **Εργαλεία Κατασκευής (Build Tools):** Το λογισμικό παρέχεται μέσω WAR overlays επιτρέποντας μια εύκολη και ευέλικτη λύση ανάπτυξης. Επιπλέον, δεν χρειάζεται να υπάρχει εγκατεστημένο το Gradle πριν από την εγκατάσταση, καθώς παρέχεται αυτόματα.
- **Git (Προαιρετικό):** Παρότι δεν αποτελεί αυστηρή απαίτηση, συνιστάται η εγκατάσταση του Git για τη διαχείριση της ανάπτυξης του CAS, καθώς και για τον έλεγχο όλων των

configuration files, των scripts και των ρυθμίσεων.

- **Λειτουργικό Σύστημα:** Δεν υπάρχει ιδιαίτερη προτίμηση όσον αφορά το λειτουργικό σύστημα, αν και οι εγκαταστάσεις σε Linux είναι συνήθως πιο κοινές από αυτές σε Windows. Η εγκατάστασή αυτή θα χρησιμοποιήσει το Debian GNU/Linux 11 (bullseye).
- **Υλικό:** Σύμφωνα με ενδείξεις από την κοινότητα, οι servers του CAS αποδίδουν καλά σε επεξεργαστή διπλού πυρήνα 3.00GHz με τουλάχιστον 4GB μνήμης RAM. Απαιτείται επίσης επαρκής χώρος στον δίσκο (κατά προτίμηση SSD) για την αποθήκευση των logs που παράγει το CAS, εάν αυτά διατηρούνται στον ίδιο τον server.

4.1.2 Ρύθμιση Περιβάλλοντος Εγκατάστασης

Πριν την έναρξη της εγκατάστασης του CAS, είναι απαραίτητη η κατάλληλη προετοιμασία του περιβάλλοντος, που στην προκείμενη περίπτωση είναι ένας Debian server, ακολουθώντας τα παρακάτω βήματα:

Εγκατάσταση Java

Το Java Development Kit (JDK) είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού που χρησιμοποιείται για την ανάπτυξη εφαρμογών και applets σε Java.
Το JDK περιλαμβάνει διάφορα εργαλεία που είναι απαραίτητα για τον προγραμματισμό σε Java, όπως τον μεταγλωττιστή (javac), τον διερμηνέα (java) και τον αποσφαλματωτή (jdb).
Επιπλέον, περιλαμβάνει ένα Java Runtime Environment (JRE), το οποίο περιλαμβάνει την Java Virtual Machine (JVM) και τις βιβλιοθήκες κλάσεων που είναι απαραίτητες για την εκτέλεση εφαρμογών Java.
Το JDK παρέχει επίσης διάφορα εργαλεία ανάπτυξης, όπως το εργαλείο javadoc για τη δημιουργία documentation, το jar για τη συσκευασία προγραμμάτων Java σε αρχεία JAR και το jarsigner για την υπογραφή αρχείων JAR.
Στην προκειμένη περίπτωση θα χρειαστεί να εγκαταστήσουμε το JDK 21[12], αυτό θα επιτευχθεί με τις παρακάτω εντολές (CLI commands):

```
curl -LO 'https://download.java.net/java/GA/jdk21.0.2/
   ↪ f2283984656d49d69e91c558476027ac/13/GPL/openjdk-21.0.2_linux-
   ↪ x64_bin.tar.gz'

tar xvf openjdk-21.0.2_linux-x64_bin.tar.gz
```

Η πρώτη εντολή χρησιμοποιεί το εργαλείο curl για να κατεβάσει το αρχείο από την ιστοσελίδα του παρόχου, ενώ η επόμενη μέσω του tar αποσυμπιέζει το αρχείο openjdk-21.0.2_linux-x64_bin.tar.gz και εξάγει τα περιεχόμενα του στον τρέχοντα κατάλογο.

Εγκατάσταση Apache Tomcat

Το Apache Tomcat είναι ένας ανοιχτού κώδικα διακομιστής ιστού και servlet container που αναπτύχθηκε από το Apache Software Foundation. Είναι σχεδιασμένο για να εκτελεί Java Servlets. Χρησιμοποιείται ευρέως λόγω της αξιοπιστίας, της κλιμακωσιμότητας και της υποστήριξης

του για διάφορες προδιαγραφές της Java EE. Για τις ανάγκες της έκδοσης του CAS που θα χρησιμοποιηθεί (7.0.0), θα χρειαστεί το Apache Tomcat 11[13]. Όπως και πριν, θα χρησιμοποιηθούν οι εντολές curl και tar για τη λήψη και εγκατάσταση του λογισμικού:

```
curl -LO 'https://dlcdn.apache.org/tomcat/tomcat-11/v11.0.0-M24/bin
   ↪ /apache-tomcat-11.0.0-M24.tar.gz'

tar xvf apache-tomcat-11.0.0-M24.tar.gz
```

Εγκατάσταση των Dependencies του Tomcat

Για να διασφαλιστεί η σωστή λειτουργία του Tomcat θα πρέπει να εγκατασταθούν μία σειρά από dependencies, κάποιες εκ των οποίων είναι μέρος του πακέτου του Tomcat 11 (Tomcat Native Library, Apache Commons Daemon), ενώ κάποιες άλλες θα πρέπει να ληφθούν από το διαδίκτυο (OpenSSL, Apache Portable Runtime).

OpenSSL Το OpenSSL είναι μια βιβλιοθήκη ανοιχτού κώδικα που παρέχει εργαλεία και πρωτόκολλα για ασφαλή επικοινωνία μέσω του διαδικτύου. Χρησιμοποιείται ευρέως για την υλοποίηση κρυπτογραφικών λειτουργιών όπως το SSL/TLS, εξασφαλίζοντας την ασφάλεια της μεταφοράς δεδομένων μεταξύ συστημάτων. Επιπλέον, το OpenSSL περιλαμβάνει πλήθος αλγορίθμων κρυπτογράφησης, γεννήτριες κλειδιών, και ψηφιακές υπογραφές, καθιστώντας το ένα από τα πιο σημαντικά εργαλεία για την ασφάλεια στο διαδίκτυο. Για το πρότζεκτ αυτό χρησιμοποιήθηκε η έκδοση 3.3.0[14], η οποία αποκτήθηκε ως εξής:

```
curl -LO 'https://www.openssl.org/source/openssl-3.3.0g.tar.gz'

tar xzf openssl-3.3.0g.tar.gz

cd openssl-3.3.0g

./config --prefix=/opt/openssl/openssl-3.3.0g shared

make depend

make

make test

make install_sw
```

Apache Portable Runtime Το Apache Portable Runtime (APR) είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα που παρέχει ένα σύνολο λειτουργιών προγραμματισμού για τη δημιουργία φορητών εφαρμογών. Το APR διευκολύνει την ανάπτυξη εφαρμογών που μπορούν να λειτουργούν σε πολλαπλά λειτουργικά συστήματα, προσφέροντας μια ομοιόμορφη διεπαφή για συνηθισμένες εργασίες, όπως διαχείριση αρχείων, διαχείριση μνήμης και χειρισμό νημάτων. Παραθέτονται οι εντολές για την λήψη και την εγκατάσταση της έκδοσης 1.7.4 του APR[15]:

```
curl -LO 'http://apache.cs.utah.edu//apr/apr-1.7.4.tar.gz'

tar xzf apr-1.7.4.tar.gz

cd apr-1.7.4.tar.gz

./configure --prefix=/opt/apr/apr-1.7.4

make depend

make

make test

make install
```

Tomcat Native Library Το Tomcat Native Library είναι μια βιβλιοθήκη που επιτρέπει την ενσωμάτωση της απόδοσης και των δυνατοτήτων του Apache Tomcat με το Apache Portable Runtime (APR). Με τη χρήση του Tomcat Native, ο διακομιστής Tomcat μπορεί να εκμεταλλευτεί βελτιωμένες δυνατότητες όπως ταχύτερη διαχείριση συνδέσεων, καλύτερη υποστήριξη SSL και βελτιωμένη απόδοση I/O. Ακολουθούν οι εντολές εγκατάστασης:

```
./configure --prefix=/opt/apr/apr-1.7.4

cd /opt/tomcat//apache-tomcat-11.0.0-M24/bin

tar xzf tomcat-native.tar.gz

cd tomcat-native-*-src/native

/configure \
--with-java-home=/usr/lib/jvm/java-openjdk \
--with-apr=/opt/apr/latest/bin/apr-1-config \
--with-ssl=/opt/openssl/latest \
--prefix=/opt/tomcat//apache-tomcat-11.0.0-M24

make

make install
```

Apache Commons Daemon Το Apache Commons Daemon είναι μια βιβλιοθήκη λογισμικού που επιτρέπει τη διαχείριση Java εφαρμογών ως υπηρεσίες ή δαίμονες (daemons) στο λειτουργικό σύστημα. Με το Apache Commons Daemon, οι εφαρμογές μπορούν να εκκινούνται αυτόματα κατά την εκκίνηση του συστήματος και να λειτουργούν στο παρασκήνιο χωρίς ανθρώπινη παρέμβαση. Η εγκατάσταση επιτυγχάνεται με παρόμοιο τρόπο με αυτόν του Tomcat

Native:

```
cd /opt/tomcat//apache-tomcat-11.0.0-M24/bin

tar xzf commons-daemon-native.tar.gz

cd commons-daemon-*-native-src/unix

./configure --with-java=/usr/lib/jvm/java-openjdk

make
```

Παραμετροποίηση των TLS/SSL Settings

Το Transport Layer Security/Secure Sockets Layer (TLS/SSL) κρυπτογραφεί τις επικοινωνίες μεταξύ πελάτη και διακομιστή, όπως αυτές που πραγματοποιούνται μεταξύ προγραμμάτων περιήγησης και ιστοσελίδων ή εφαρμογών. Η σωστή παραμετροποίηση των ρυθμίσεων TLS/SSL διασφαλίζει την ασφαλή κρυπτογράφηση της μεταφοράς δεδομένων, προστατεύοντας τόσο την ακεραιότητα όσο και την εμπιστευτικότητα των πληροφοριών. Για να επιτευχθεί αυτό, είναι απαραίτητο να ακολουθηθούν τα εξής βήματα:

Δημιουργία ιδιωτικού κλειδιού και πιστοποιητικού Τα ιδιωτικά κλειδιά (private keys) αποτελούν θεμελιώδη στοιχεία στην κρυπτογραφία, καθώς χρησιμοποιούνται για την αποκρυπτογράφηση δεδομένων που έχουν κρυπτογραφηθεί με το αντίστοιχο δημόσιο κλειδί. Στο πλαίσιο του TLS/SSL, τα ιδιωτικά κλειδιά διασφαλίζουν την ασφαλή επικοινωνία μεταξύ διακομιστή και πελάτη.
Τα πιστοποιητικά (certificates) είναι ψηφιακά έγγραφα που επιβεβαιώνουν την ταυτότητα ενός διακομιστή ή χρήστη και επιτρέπουν την ασφαλή, κρυπτογραφημένη επικοινωνία. Ένα πιστοποιητικό περιέχει πληροφορίες όπως το δημόσιο κλειδί, το όνομα του κατόχου, και την υπογραφή μιας αξιόπιστης αρχής πιστοποίησης (CA). Η αξιόπιστη αρχή πιστοποίησης εγγυάται την αυθεντικότητα του πιστοποιητικού, διασφαλίζοντας ότι ο διακομιστής ή ο χρήστης είναι πραγματικά αυτός που ισχυρίζεται ότι είναι, και επιτρέποντας έτσι την ασφαλή ανταλλαγή δεδομένων μέσω του δικτύου. Να σημειωθεί ότι για τις ανάγκες της δοκιμαστικής αυτής υλοποίησης επαρκεί η χρήση ενός self-signed certificate, δηλαδή ενός πιστοποιητικού το οποίο δεν έχει υπογραφεί από κάποια αρχή πιστοποίησης αλλά από εμάς τους ίδιους.
Για την ορθή λειτουργία του CAS server, είναι απαραίτητο να δημιουργηθούν τα παραπάνω στοιχεία, κάτι που θα επιτευχθεί με την παρακάτω εντολή.

```
openssl req -newkey rsa:2048 -keyout domain.key -x509 -days 365 -
   ↪ out domain.crt
```

Η εντολή καλεί το εργαλείο OpenSSL για δημιουργία αιτήματος υπογραφής πιστοποιητικού (CSR), ενώ οι παράμετροι επιτυγχάνουν τα εξής:

- -newkey rsa:2048: Δημιουργεί ένα νέο ιδιωτικό κλειδί RSA με μήκος 2048 bit.
- -keyout domain.key: Αποθηκεύει το ιδιωτικό κλειδί που δημιουργήθηκε στο αρχείο domain.key.

- -x509: Δημιουργεί ένα αυτο-υπογεγραμμένο πιστοποιητικό αντί για ένα αίτημα υπογραφής πιστοποιητικού (CSR).
- -days 365: Ορίζει τη διάρκεια ισχύος του πιστοποιητικού σε 365 ημέρες.
- -out domain.crt: Αποθηκεύει το αυτο-υπογεγραμμένο πιστοποιητικό στο αρχείο domain.crt.

Εισαγωγή του πιστοποιητικού σε ένα Java keystore Επόμενο βήμα αποτελεί η αλλαγή του format του πιστοποιητικού σε PKCS12, ξανά με την χρήση του εργαλείου του openssl, και η εισαγωγή του στο Java keystore του tomcat (keystore.jks).
Όταν είναι αναγκαία η εισαγωγή ενός πιστοποιητικού (.crt) σε ένα Java Keystore, είναι συχνά απαραίτητη η μετατροπή του πρώτα σε μορφή PKCS12 (.p12). Η μορφή PKCS12 περιλαμβάνει τόσο το πιστοποιητικό όσο και το ιδιωτικό κλειδί σε ένα ενιαίο αρχείο, το οποίο είναι απαραίτητο για την ασφαλή αποθήκευση και χρήση τους από το Java Keystore.
Τα Java Keystores είναι αποθετήρια που χρησιμοποιούνται για την αποθήκευση κρυπτογραφικών κλειδιών και πιστοποιητικών, τα οποία απαιτούνται για την ασφαλή επικοινωνία μέσω πρωτοκόλλων όπως το TLS/SSL.

```
openssl pkcs12 -inkey domain.key -in domain.crt -export -out domain
   ↪ .pfx

keytool -importkeystore -srckeystore domain.pfx -srcstoretype
   ↪ pkcs12 -destkeystore keystore.jks
```

Διαμόρφωση των ρυθμίσεων του Tomcat server Τέλος πρέπει να γίνουν ορισμένες αλλαγές στο αρχείο server.xml του Tomcat, για την σωστή παραμετροποίηση και λειτουργία του TLS/SSL connector.
Πρώτα θα πρέπει να εντοπιστεί ο HTTP connector της πύλης 8080 και να απενεργοποιηθεί, καθώς όλες οι μεταφορές δεδομένων που σχετίζονται με το CAS πρέπει να πραγματοποιούνται μέσω ενός ασφαλούς καναλιού.

```
<!--
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
```

Ενώ αντίστοιχα, θα πρέπει να ενεργοποιηθεί και να παραμετροποιηθεί με τον παρακάτω τρόπο ο HTTPS connector της πύλης 8443.

```
<Connector port="8443" protocol="org.apache.coyote.http11.
   ↪ Http11NioProtocol"
    sslImplementationName="org.apache.tomcat.util.net.openssl.
       ↪ OpenSSLImplementation"
    SSLEnabled="true" connectionTimeout="20000" maxThreads="150">
    <SSLHostConfig
            ciphers="ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128
               ↪ -GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-
               ↪ RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
               ↪ POLY1305:ECDHE-RSA-CHACHA220-POLY1305:DHE-RSA-
               ↪ AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-
               ↪ RSA-CHACHA20-POLY1305"
            honorCipherOrder="false" protocols="TLSv1.2, TLSv1.3"
            disableSessionTickets="true">
            <Certificate
                    certificateKeystoreFile="/opt/tomcat/keystore.
                       ↪ jks"
                    certificateKeystorePassword="changeit"
                    certificateKeyAlias="cas.iee.ihu.gr"
                    type="RSA" />
    </SSLHostConfig>
    <UpgradeProtocol className="org.apache.coyote.http2.
       ↪ Http2Protocol" />
 </Connector>
```

Ακόμη θα απενεργοποιηθεί και ο AJP (Apache JServ Protocol) connector της πύλης 8009, μιας και δεν θα χρησιμοποιηθεί στην παρούσα υλοποίηση.

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<!--
<Connector protocol="AJP/1.3"
           address="::1"
           port="8009"
           redirectPort="8443" />
-->
```

4.2 Λήψη και Εγκατάσταση του CAS Server

Η διαδικασία εγκατάστασης του CAS Server ξεκινά με την επιλογή της κατάλληλης έκδοσης, η οποία για την παρούσα εργασία είναι η 7.0.0. Στη συνέχεια, προχωράμε στη λήψη του πρότζεκτ από το GitHub, εξασφαλίζοντας έτσι ότι διαθέτουμε το πιο πρόσφατο και συμβατό λογισμικό για τις ανάγκες μας.

```
git clone https://github.com/apereo/cas-overlay-template.git

git checkout 7.0.x
git checkout -b cas.iee.ihu.gr
```

4.2.1 WAR Overlay

Το πρότζεκτ προσφέρεται στην μορφή ενός WAR (Web Application Archive) overlay. Η τεχνολογία των WAR overlays αναφέρεται στη διαδικασία υλοποίησης και ανάπτυξης web εφαρμογών χρησιμοποιώντας αρχεία WAR σε περιβάλλοντα όπως το Apache Maven και το Gradle. Τα αρχεία WAR είναι αρχεία συμπιεσμένα σε μορφή JAR που περιέχουν όλα τα στοιχεία μιας web εφαρμογής, όπως ο κώδικας, οι βιβλιοθήκες και οι πόροι, καθιστώντας τα έτοιμα για ανάπτυξη σε ένα servlet container όπως το Apache Tomcat.
Στην περίπτωση του Maven και του Gradle, η διαδικασία δημιουργίας ενός WAR αρχείου μπορεί να αυτοματοποιηθεί μέσω κατάλληλων plugins, τα οποία αναλαμβάνουν να συλλέξουν όλα τα απαραίτητα αρχεία και τις εξαρτήσεις, να τα τοποθετήσουν στις κατάλληλες θέσεις και να δημιουργήσουν το τελικό WAR αρχείο. Τα WAR overlays επιτρέπουν στους προγραμματιστές να επαναχρησιμοποιούν και να επεκτείνουν υπάρχοντα WAR αρχεία, προσθέτοντας ή τροποποιώντας συγκεκριμένα στοιχεία, χωρίς να χρειάζεται να αναπτύξουν ολόκληρη την εφαρμογή από την αρχή. Αυτό προσφέρει ευελιξία και αποδοτικότητα στη διαχείριση και την ανάπτυξη μεγάλων web εφαρμογών.
Το WAR overlay, χωρίς καμία επιπλέον ρύθμιση, θα δημιουργήσει έναν βασικό “προεπιλεγμένο” CAS server. Αυτός ο server μπορεί να μην προσφέρει πολλές δυνατότητες, αλλά θα επιτρέψει την επιβεβαίωση της μέχρι τώρα σωστής υλοποίησης. Επίσης, αποτελεί ένα σημείο εκκίνησης για περαιτέρω ρυθμίσεις και εξατομίκευση. Για την δημιουργία του προεπιλεγμένου server, θα πρέπει να εκτελεστούν οι παρακάτω εντολές.

```
cd cas-overlay-template

./gradlew clean build
```

4.2.2 Αρχική Παραμετροποίηση

Το CAS, βάσει των προεπιλεγμένων ρυθμίσεων, αναζητά τα αρχεία διαμόρφωσης του στον κατάλογο /etc/cas. Σχεδόν κάθε πτυχή της παραμετροποίησης του διακομιστή CAS μπορεί να ελεγχθεί μέσω μεταβλητών, οι οποίες αποθηκεύονται στο αρχείο cas.properties, το οποίο βρίσκεται στον φάκελο /etc/cas/config. Η αρχική μορφή του αρχείου μοιάζει ως εξής:

```
#Server conf
cas.server.name=https://cas.iee.ihu.gr:8443
cas.server.prefix=${cas.server.name}/cas
logging.config=file:/etc/cas/config/log4j2.xml

#Ticket granting cookie encryption
cas.tgc.secure=true
cas.tgc.crypto.signing.key-size=512
cas.tgc.crypto.encryption.key-size=256

cas.tgc.crypto.signing.key=kQU**********************
```

```
cas.tgc.crypto.encryption.key=740**********

#Spring Webflow encryption
cas.webflow.crypto.signing.key=mf-******************
cas.webflow.crypto.encryption.key=1nt*******
```

Οι πρώτες δύο μεταβλητές αναφέρονται στο URL του server (με την σειρά protocol, domain name, και port) και το URL της web εφαρμογής αντίστοιχα.
Η μεταβλητή logging.config ορίζει το αρχείο log4j2.xml ως την πηγή των ρυθμίσεων για την καταγραφή των logs. Το Log4j2 αποτελεί framework ανοιχτού κώδικα που χρησιμοποιείται για την καταγραφή (logging) συμβάντων σε εφαρμογές Java. Προσφέρει υψηλή ευελιξία και επεκτασιμότητα, επιτρέποντας στους προγραμματιστές να καταγράφουν μηνύματα διαφόρων επιπέδων (όπως error, warning, info) και να τα αποθηκεύουν σε πολλαπλές εξόδους, όπως αρχεία, βάσεις δεδομένων ή κονσόλες[16].
Στη συνέχεια, πρέπει να ρυθμιστεί η κρυπτογράφηση τόσο των Ticket Granting Cookies όσο και του Spring Webflow. Ο διακομιστής CAS χρησιμοποιεί ένα TGT, προκειμένου να διατηρεί την κατάσταση σύνδεσης κατά τη διάρκεια των συνεδριών Single Sign-On. Ο client μπορεί να παρουσιάσει αυτό το cookie στον CAS αντί για τα πρωτεύοντα διαπιστευτήρια και, εφόσον είναι έγκυρο, να πιστοποιηθεί. Ο CAS χρησιμοποιεί το Spring Webflow για τη διαχείριση της διαδικασίας αυθεντικοποίησης, κάτι το οποίο επίσης απαιτεί κρυπτογράφηση.
Τα παραπάνω επιτυγχάνονται με την δημιουργία των κατάλληλων JSON Web Keys, ανάλογα με το προαπαιτούμενο μέγεθος των κλειδιών (512-bit για το signing key και 256-bit για το encryption key αντίστοιχα). Τα JSON Web Keys (JWK) αποτελούν πρότυπο για την αναπαράσταση κρυπτογραφικών κλειδιών σε μορφή JSON[17]. Χρησιμοποιούνται συνήθως σε εφαρμογές που βασίζονται σε πρωτόκολλα ασφαλείας, όπως το OAuth 2.0 και το OpenID Connect, για την ασφαλή ανταλλαγή και επικύρωση κλειδιών μεταξύ δύο πλευρών. Ένα JWK περιέχει τις πληροφορίες του κλειδιού όπως τον αλγόριθμο, τον τύπο του κλειδιού και το ίδιο το κλειδί, σε μορφή που είναι εύκολα κατανοητή και επεξεργάσιμη από μηχανές.
Η δημιουργία των κλειδιών μπορεί να επιτευχθεί μέσω κάποιου εργαλείου όπως το mkjwk-JSON Web Key Generator[18]. Να σημειωθεί ότι το encryption key του Spring WebFlow δεν αποτελεί JSON Web Key, αλλά μία τυχαία δημιουργημένη συμβολοσειρά 16 οκτάδων με κωδικοποίηση Base64, και μπορεί να δημιουργηθεί με το εργαλείο του openssl με την την παρακάτω εντολή.

```
openssl rand -base64 16
```

4.2.3 Εγκατάσταση μέσω του Tomcat

Για την εγκατάσταση της εφαρμογής CAS, πρέπει να μεταφερθεί η εφαρμογή που μόλις δημιουργήθηκε με το Gradle στον κατάλογο webapps του Tomcat, καθώς και να δημιουργηθεί αντίγραφο των περιεχομένων του καταλόγου etc/cas στον κατάλογο /etc/cas.
Σύμφωνα με τις οδηγίες ασφαλείας, οι διαδικτυακές εφαρμογές πρέπει να αναπτύσσονται ως αποσυμπιεσμένοι κατάλογοι αντί για αρχεία WAR. Όλα τα αρχεία πρέπει να ανήκουν στον χρήστη root και στην ομάδα tomcat, ενώ οι άδειες αρχείων πρέπει να είναι ρυθμισμένες ώστε ο κάτοχος να έχει δικαιώματα ανάγνωσης/εγγραφής, η ομάδα να έχει μόνο δικαιώματα ανάγνω-

σης και κανένα δικαίωμα για τους υπόλοιπους χρήστες. Για να επιτευχθούν όλα τα παραπάνω, θα πρέπει να εκτελεστούν οι ακόλουθες εντολές.

```
cd cas-overlay-template

./gradlew clean copyCasConfiguration build
./gradlew unzipWAR

cd cas-overlay-template/build

tar czf /tmp/cassrv-files.tgz --exclude app/META-INF --owner=root
   ↪ --group=tomcat --mode=g-w,o-rwx app

systemctl stop tomcat

cd /opt/tomcat/latest/webapps
tar xzf /tmp/cassrv-files.tgz
mv app cas

systemctl start tomcat
```

Η παραπάνω διαδικασία πρέπει να επαναλαμβάνεται κάθε φορά που γίνονται προσθήκες ή αλλαγές στην εφαρμογή.

4.3 Προσθήκη Επιπρόσθετων Δυνατοτήτων

Η διαδικασία προσθήκης επιπρόσθετων δυνατοτήτων στον CAS server αποτελείται κυρίως από την προσθήκη κάποιου καινούργιου dependency στο αρχείο build.gradle, το οποίο βρίσκεται στον κύριο κατάλογο της εφαρμογής, και την ρύθμιση των κατάλληλων μεταβλητών στο αρχείο cas.properties.
Τα Gradle dependencies είναι βιβλιοθήκες ή άλλα εξωτερικά αρχεία που χρειάζεται ένα πρότζεκτ για να λειτουργήσει σωστά. Μέσω του αρχείου build.gradle, καθορίζεται ποιες εξαρτήσεις απαιτούνται και το Gradle αναλαμβάνει αυτόματα να τις κατεβάσει και να τις ενσωματώσει στο πρότζεκτ. Αυτό απλοποιεί τη διαχείριση των βιβλιοθηκών και εξασφαλίζει ότι το πρότζεκτ παραμένει ενημερωμένο με τις τελευταίες εκδόσεις τους.

4.3.1 Service Registry

Ο διακομιστής CAS περιλαμβάνει μια δομή διαχείρισης υπηρεσιών που επιτρέπει στους διαχειριστές του διακομιστή να δηλώνουν και να διαμορφώνουν ποιες υπηρεσίες (CAS clients) μπορούν να τον χρησιμοποιούν, αλλά και με ποιον τρόπο. Ο βασικός πυλώνας της δομής διαχείρισης υπηρεσιών είναι το μητρώο υπηρεσιών (service regisrty), το οποίο αποθηκεύει πληροφορίες για τις καταχωρημένες υπηρεσίες. Σε αυτές τις πληροφορίες συμπεριλαμβάνονται οι τρόποι με τους οποίους οι υπηρεσίες πρέπει να αυθεντικοποιούν τους χρήστες, ποιοι χρήστες μπορούν να έχουν πρόσβαση στην υπηρεσία και υπό ποιες συνθήκες, καθώς και δεδομένα για τους εξουσιοδοτημένους χρήστες που οι υπηρεσίες μπορούν να προσπελάσουν.

Ο βασικός διακομιστής CAS που δημιουργήθηκε στην προηγούμενη ενότητα δεν περιλαμβάνει μητρώο υπηρεσιών. Πριν καταστεί δυνατή η δημιουργία και η χρήση οποιουδήποτε δοκιμαστικού client, είναι απαραίτητο να προστεθεί ένα μητρώο υπηρεσιών στον διακομιστή. Αρχικά, θα προστεθεί ένα απλό μητρώο που χρησιμοποιεί αρχεία JSON για την περιγραφή των υπηρεσιών. Αργότερα, θα αντικατασταθεί με ένα πιο ανθεκτικό μητρώο, όταν διαμορφώσουμε τον διακομιστή για την παροχή υψηλής διαθεσιμότητας.
Για την προσθήκη του μητρώου υπηρεσιών θα πρέπει να υλοποιηθούν οι παρακάτω ενέργειες. Πρώτα. θα πρέπει να συμπεριληφθεί το ανάλογο dependency στο αρχείο build.gradle.

```
implementation "org.apereo.cas:cas-server-support-json-service-
    ↪ registry:${project.'cas.version'}"
```

Καθώς επίσης και η παρακάτω γραμμή στο αρχείο cas.properties, η οποία καθορίζει την τοποθεσία στην οποία θα αναζητήσει η εφαρμογή τα JSON αρχεία με τις δηλώσεις των υπηρεσιών. Στην προκειμένη περίπτωση, επιλέχθηκε ο κατάλογος /etc/cas/services.

```
cas.serviceRegistry.json.location:    file:/etc/cas/services
```

Τα JSON αρχεία που περιέχουν τις δηλώσεις των υπηρεσιών, έχουν την εξής μορφή.

```
{
    "@class" : "org.apereo.cas.services.CasRegisteredService",
    "serviceId" : "testId",
    "name" : "testJsonFile",
    "id" : 103935657744185,
    "evaluationOrder" : 10
}
```

Κάθε πεδίο ορίζει μία ιδιότητα της υπηρεσίας. Το παραπάνω παράδειγμα δεν είναι εξαντλητικό, καθώς υπάρχουν πολλές άλλες διαθέσιμες ιδιότητες, όμως τα παραπάνω πεδία αποτελούν τα βασικά.

- @class: Καθορίζει το πρωτόκολλο που αναμένεται να χρησιμοποιηθεί για την αυθεντικοποίηση από τον server. Στο παράδειγμα αυτό, η τιμή δηλώνει ότι ο server θα χρησιμοποιήσει το πρωτόκολλο CAS.
- serviceId: Αποτελεί μία regular expression που περιγράφει το URL(s) στα οποία βρίσκεται η υπηρεσία. Είναι σημαντικό να αποφεύγονται μοτίβα που μπορεί να ταιριάζουν με περισσότερα από τα επιθυμητά URLs, καθώς αυτό μπορεί να δημιουργήσει κενά ασφαλείας.
- name: Περιγράφει το όνομα της υπηρεσίας.
- id: Ορίζει έναν μοναδικό αριθμό ταυτοποίησης για τον ορισμό της υπηρεσίας.
- evaluationOrder: Είναι μία τιμή που καθορίζει τη σχετική σειρά αξιολόγησης των καταχωρημένων υπηρεσιών (οι χαμηλότερες τιμές προηγούνται των υψηλότερων). Αυτό είναι ιδιαίτερα σημαντικό όταν περισσότερες από μία εκφράσεις serviceId μπορεί να ταιριάζουν με την ίδια υπηρεσία, το evaluationOrder καθορίζει ποια έκφραση θα αξιολογηθεί πρώτη.

4.3.2 Lightweight Directory Access Protocol

Το Lightweight Directory Access Protocol (LDAP) είναι ένα πρωτόκολλο που χρησιμοποιείται για την πρόσβαση και τη διαχείριση καταλόγων πληροφοριών σε ένα δίκτυο. Οι κατάλογοι αυτοί οργανώνουν δεδομένα σε μια ιεραρχική δομή και χρησιμοποιούνται συχνά για την αποθήκευση πληροφοριών όπως οι λογαριασμοί χρηστών, ρυθμίσεις ασφαλείας και άλλες υπηρεσίες καταλόγου[19].
Το LDAP είναι σχεδιασμένο να είναι ελαφρύ και αποδοτικό, καθιστώντας το ιδανικό για χρήση σε εφαρμογές που έχουν ως σκοπό την διαχείριση ταυτοτήτων και πρόσβασης. Υποστηρίζει βασικές λειτουργίες όπως αναζήτηση, προσθήκη, τροποποίηση και διαγραφή δεδομένων, διευκολύνοντας την κεντρική διαχείριση πληροφοριών σε ένα δίκτυο.
Στην παρούσα εργασία, το LDAP αποτελεί σημαντικό κομμάτι της υλοποίησης, όχι μόνο γιατί παρέχει μία αξιόπιστη λύση αλλά κυρίως επειδή τα δεδομένα των χρηστών αποθηκεύονται ήδη σε έναν LDAP κατάλογο. Κατά συνέπεια, η ενσωμάτωση του στο CAS ήταν απαραίτητη προϋπόθεση. Στην συνέχεια, παρουσιάζεται αναλυτικά η διαδικασία ενσωμάτωσης.
Η διαδικασία ξεκινά με τον ορισμό του LDAP dependency στο αρχείο build.gralde του πρότζεκτ.

```
    implementation "org.apereo.cas:cas-server-support-ldap:${project.'
       ↪ cas.version'}"
```

Και συνεχίζεται με την παραμετροποίηση των cas properties.

```
    #LDAP Authentication
    cas.authn.accept.users=
    cas.authn.ldap[0].type=AUTHENTICATED
    cas.authn.ldap[0].ldap-url=ldap://ldap.it.teithe.gr:389
    cas.authn.ldap[0].base-dn=ou=people,dc=it,dc=teithe,dc=gr
    cas.authn.ldap[0].search-filter=uid={user}
    cas.authn.ldap[0].bind-dn=cn=casadmin,dc=it,dc=teithe,dc=gr
    cas.authn.ldap[0].bind-credential=*****
    cas.authn.ldap[0].principal-attribute-list=cn,sn,uid,givenName,mail
```

Οι παραπάνω μεταβλητές αφορούν τη χρήση του LDAP για την αυθεντικοποίηση των χρηστών. Η πρώτη μεταβλητή απενεργοποιεί τη χρήση των ενσωματωμένων διαπιστευτηρίων που προσφέρει το CAS, κάτι που αποτελεί κρίσιμο βήμα, καθώς η εφαρμογή θα συνεχίσει να τα χρησιμοποιεί ανεξάρτητα από την περαιτέρω διαμόρφωση εάν δεν απενεργοποιηθούν.
Τα υπόλοιπα properties καθοδηγούν τον διακομιστή CAS στη διεύθυνση του LDAP directory και ορίζουν τα διαπιστευτήρια που θα χρησιμοποιηθούν για την πρόσβαση στις πληροφορίες. Επιπλέον, προσδιορίζεται ο τρόπος πλοήγησης και αναζήτησης χρηστών μέσα στον κατάλογο.

```
    #LDAP Attribute resolution
    cas.authn.attribute-repository.ldap[0].ldap-url=ldap://ldap.it.
       ↪ teithe.gr:389
    cas.authn.attribute-repository.ldap[0].search-filter=uid={user}
    cas.authn.attribute-repository.ldap[0].base-dn=ou=people,dc=it,dc=
       ↪ teithe,dc=gr
    cas.authn.attribute-repository.ldap[0].bind-dn=cn=casadmin,dc=it,dc
       ↪ =teithe,dc=gr
    cas.authn.attribute-repository.ldap[0].bind-credential=******
```

```
cas.authn.attribute-repository.core.aggregation=MERGE
cas.authn.attribute-repository.ldap[0].attributes.cn=cn
cas.authn.attribute-repository.ldap[0].attributes.displayName=
   ↪ displayName
cas.authn.attribute-repository.ldap[0].attributes.givenName=
   ↪ givenName
cas.authn.attribute-repository.ldap[0].attributes.mail=mail
cas.authn.attribute-repository.ldap[0].attributes.sn=sn
cas.authn.attribute-repository.ldap[0].attributes.udcid=
   ↪ UDC_IDENTIFIER
cas.authn.attribute-repository.ldap[0].attributes.uid=uid
cas.authn.attribute-repository.ldap[0].attributes.group=group
cas.authn.attribute-repository.ldap[0].attributes.am=am
cas.authn.attribute-repository.ldap[0].attributes.
   ↪ eduPersonAffiliation=eduPersonAffiliation
cas.authn.attribute-repository.ldap[0].attributes.
   ↪ eduPersonScopedAffiliation=eduPersonScopedAffiliation
cas.authn.attribute-repository.ldap[0].attributes.
   ↪ eduPersonEntitlement=eduPersonEntitlement
cas.authn.attribute-repository.ldap[0].attributes.
   ↪ eduPersonPrimaryAffiliation=eduPersonPrimaryAffiliation
cas.authn.attribute-repository.ldap[0].attributes.cn;lang-el=cn;
   ↪ lang-el
cas.authn.attribute-repository.ldap[0].attributes.description=
   ↪ description
cas.authn.attribute-repository.ldap[0].attributes.description;lang-
   ↪ el=description;lang-el
cas.authn.attribute-repository.ldap[0].attributes.fathersname=
   ↪ fathersname
cas.authn.attribute-repository.ldap[0].attributes.fathersname;lang-
   ↪ el=fathersname;lang-el
cas.authn.attribute-repository.ldap[0].attributes.givenName;lang-el
   ↪ =givenName;lang-el
cas.authn.attribute-repository.ldap[0].attributes.regsem=regsem
cas.authn.attribute-repository.ldap[0].attributes.regyear=regyear
cas.authn.attribute-repository.ldap[0].attributes.labeledURI=
   ↪ labeledURI
cas.authn.attribute-repository.ldap[0].attributes.pwdChangedTime=
   ↪ pwdChangedTime
cas.authn.attribute-repository.ldap[0].attributes.secondarymail=
   ↪ secondarymail
cas.authn.attribute-repository.ldap[0].attributes.sem=sem
cas.authn.attribute-repository.ldap[0].attributes.sn;lang-gr=sn;
   ↪ lang-el
cas.authn.attribute-repository.ldap[0].attributes.telephoneNumber=
   ↪ telephoneNumber
cas.authn.attribute-repository.ldap[0].attributes.title=title
cas.authn.attribute-repository.ldap[0].attributes.title;lang-el=
```

```
    ↪ title;lang-el
cas.authn.attribute-repository.ldap[0].attributes.profilePhoto=
    ↪ profilePhoto
cas.authn.attribute-repository.ldap[0].attributes.socialMedia=
    ↪ socialMedia
```

Τέλος, σε αυτή την ενότητα, το παραπάνω απόσπασμα παραμετροποιεί το attribute resolution του LDAP. Το attribute resolution περιλαμβάνει την αναζήτηση και εξαγωγή των απαραίτητων χαρακτηριστικών από τον LDAP κατάλογο, βάσει των αιτημάτων μιας εφαρμογής. Για παράδειγμα, όταν ένας χρήστης συνδέεται σε μια εφαρμογή, το σύστημα μπορεί να χρειαστεί να ανακτήσει το όνομα, την ομάδα ή άλλα στοιχεία του χρήστη από το LDAP. Οι μεταβλητές που αναφέρονται παραπάνω χρησιμοποιούνται για την αντιστοίχηση του LDAP directory με τον CAS server.
Η μεταβλητή cas.authn.attribute-repository.core.aggregation=MERGE καθορίζει πώς ο CAS server διαχειρίζεται περιπτώσεις όπου το ίδιο attribute εμφανίζεται σε περισσότερους από έναν καταλόγους. Η τιμή ”MERGE” συνιστά τον συνδυασμό των αποτελεσμάτων, δημιουργώντας μια λίστα στην οποία οι τιμές διαχωρίζονται με κόμμα.

4.3.3 OAuth 2.0

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, η ενσωμάτωση και χρήση του πρωτοκόλλου OAuth 2.0 για την επικοινωνία με τον CAS server αποτελεί απαραίτητη προϋπόθεση για το καινούργιο σύστημα SSO του τμήματος.
Για την επίτευξη της παραπάνω χρηστικότητας, θα χρειαστεί να γίνουν οι εξής προσθήκες στα αρχεία build.gradle και cas.properties αντίστοιχα.
Αρχικά η δήλωση του κατάλληλου dependency.

```
    implementation "org.apereo.cas:cas-server-support-oauth-webflow:${
    ↪ project.'cas.version'}"
```

Έπειτα ο ορισμός των παρακάτω μεταβλητών στο αρχείο cas.properties.

```
    #OAUTH settings
    cas.authn.oauth.access-token.crypto.encryption.key=qbS**********
    cas.authn.oauth.access-token.crypto.signing.key=WjS************

    cas.authn.oauth.crypto.encryption.key=xjp********
    cas.authn.oauth.crypto.signing.key=PVjM**********

    cas.authn.oauth.session-replication.cookie.crypto.encryption.key=
    ↪ M3h****
    cas.authn.oauth.session-replication.cookie.crypto.signing.key=QAz
    ↪ ******
```

Όλες οι μεταβλητές αναφέρονται στην δημιουργία κρυπτογραφικών κλειδιών για τα διάφορα συστατικά στοιχεία του πρωτοκόλλου, κάθε κλειδί αποτελεί ένα JWK.
Μετά την επιτυχή ενσωμάτωση των modules που επιτρέπουν την χρήση του OAuth 2.0 από τον CAS server, προσφέρονται τα εξής endpoints για την λειτουργία του πρωτοκόλλου:

- /oauth2.0/authorize: Εξουσιοδότηση του χρήστη και εκκίνηση της διαδικασίας αυθεντικοποίησης του CAS.
- /oauth2.0/accessToken,/oauth2.0/token: Λήψη ενός access token σε απλό κείμενο ή JSON.
- /oauth2.0/profile: Λήψη του προφίλ του αυθεντικοποιημένου χρήστη σε μορφή JSON μέσω της παραμέτρου access_token.
- /oauth2.0/introspect: Αναζήτηση στο CAS για την κατάσταση ενός access token μέσω introspection.
- /oauth2.0/device: Έγκριση κωδικών συσκευής μέσω του device code flow.
- /oauth2.0/revoke: Απόρριψη access ή refresh token. Για τη χρήση αυτού του endpoint είναι αναγκαία η αυθεντικοποίηση με τη χρήση των client_id και client_secret της υπηρεσίας OAuth2 ως όνομα χρήστη και κωδικός πρόσβασης.

4.3.4 High Availability

Η υποστήριξη της υψηλής διαθεσιμότητας (High Availability - HA) για τον CAS server είναι κρίσιμη για την εξασφάλιση της συνεχούς πρόσβασης των χρηστών σε υπηρεσίες που απαιτούν έλεγχο ταυτότητας. Η αρχιτεκτονική υψηλής διαθεσιμότητας διασφαλίζει ότι ο CAS server παραμένει λειτουργικός και διαθέσιμος, ακόμη και σε περιπτώσεις βλαβών ή αυξημένων φορτίων.
Η έκδοση 7.0.x του CAS υποστηρίζει ποικιλία από μνήμες cache και βάσεις δεδομένων για την υλοποίηση των μητρώων εισιτηρίων (ticket registries), των μητρώων υπηρεσιών (service registries) και της αποθήκευσης αρχείων καταγραφής (logging files).

Service Registry

Όπως αναφέρθηκε σε προηγούμενη ενότητα, το μητρώο υπηρεσιών αποτελεί κρίσιμο στοιχείο για τη διαλειτουργικότητα ενός διακομιστή CAS. Για την εξασφάλιση υψηλής διαθεσιμότητας και απόδοσης, η χρήση μιας βάσης δεδομένων για την αποθήκευση των αρχείων του Service Registry είναι απαραίτητη, αντί της αποθήκευσης στο σύστημα αρχείων του διακομιστή. Μια βάση δεδομένων προσφέρει αυξημένη αξιοπιστία, ευκολότερη διαχείριση και βελτιωμένη κλιμάκωση, ιδίως σε περιβάλλοντα με πολλούς διακομιστές. Παράλληλα, διευκολύνει την αναζήτηση, την ενημέρωση και τη διαχείριση των καταχωρήσεων υπηρεσιών, μειώνοντας τον κίνδυνο απώλειας δεδομένων και ενισχύοντας την ασφάλεια.
Το πρότζεκτ προσφέρει μία πληθώρα επιλογών από βάσεις δεδομένων όπως σχεσιακές βάσεις δεδομένων μέσω του Java Persistence API (JPA), NoSQL Databases αλλά και επιλογές όπως το Git και το LDAP.

Java Persistence API Το Java Persistence API χρησιμοποιείται για τον ορισμό του persistence και του object-relational mapping (ORM), κυρίως με την μορφή annotations ή XML configuration files, τα οποία δίνουν την δυνατότητα αντιστοίχησης αντικειμένων του κώδικα με μία σχεσιακή βάση δεδομένων και την αποθήκευση τους. Το CAS υποστηρίζει δύο JPA implementations,

το Hibernate και το EclipseLink, μέσω των οποίων δίνεται η δυνατότητα χρήσης διάφορων σχεσιακών βάσεων όπως η MySQL και η Postgress.

NoSQL Databases Οι βάσεις δεδομένων NoSQL είναι μη σχεσιακές βάσεις δεδομένων που δεν χρησιμοποιούν την παραδοσιακή δομή των σχεσιακών βάσεων δεδομένων που βασίζεται σε πίνακα. Αντίθετα, αποθηκεύουν δεδομένα σε μορφή που είναι πιο ευέλικτη και ευκολότερα επεκτάσιμη όπως σε documents, key-value pairs, ή graph structures. Αρχικά υπάρχει επιλογή του Redis (Remote Dictionary Server), το οποίο είναι ένα in-memory data structure store και αποθηκεύει στην RAM τα δεδομένα ως key-value pairs. Ωστόσο, για την υποστήριξη του high availability είναι δυνατή η εγγραφή των δεδομένων στον δίσκο. Τα keys είναι σε μορφή String, ενώ τα values μπορούν να πάρουν πολλές μορφές (όπως strings, hashes, lists, sets, sorted sets, streams κ.α), στην προκειμένη περίπτωση στα values θα τοποθετούνται τα service definitions. Έπειτα υπάρχουν τα document stores, τα οποία αποθηκεύουν τα δεδομένα σε μορφή αρχείων JSON ή κάποια όμοια με αυτήν (π.χ BSON). Σε αυτήν την κατηγορία το CAS υποστηρίζει τις MongoDB, CouchDB, CouchBase και Apache Cassandra. Όλες οι παραπάνω βάσεις αποθηκεύουν τα δεδομένα στον δίσκο, ενώ έχουν τη δυνατότητα να λειτουργήσουν και ως key-value stores. Η χρήση ενός document-oriented database θα ήταν αρκετά βολική για την αποθήκευση των service definitions.

Git Τα αρχεία των service definitions μπορούν να αποθηκευτούν σε κάποιο remote git repository, σε μορφή JSON. Τα περιεχόμενα του repository γίνονται pull σε προκαθορισμένες στιγμές, ενώ οι αλλαγές γίνονται committed και pushed.

LDAP Υπάρχει η δυνατότητα να γίνει mapping των service definitions σε ένα LDAP schema και να αποθηκευτούν σε ένα ανάλογο directory. Στην παραπάνω περίπτωση θα πρέπει το objectclass να οριστεί ως casRegisteredService, ενώ η αναζήτηση θα γίνεται με το uid attribute.

Επίλογος Μετά από προσεκτική αξιολόγηση των διαθέσιμων επιλογών, λαμβάνοντας υπόψη κριτήρια όπως η ευκολία εγκατάστασης, χρήσης και συντήρησης, καθώς και η δυνατότητα ενσωμάτωσης της βάσης δεδομένων με άλλα συστήματα, η MongoDB επιλέχθηκε ως η καταλληλότερη λύση.
Η MongoDB είναι μια ανοιχτού κώδικα βάση δεδομένων τύπου NoSQL. Αποθηκεύει δεδομένα σε έγγραφα που μοιάζουν με JSON, τα οποία περιέχουν ζεύγη πεδίων και τιμών. Η τιμή ενός πεδίου μπορεί να είναι ένας από διάφορους τύπους δεδομένων, όπως αριθμοί, αλφαριθμητικά, λογικές τιμές, ημερομηνίες, αντικείμενα και πίνακες. Στη MongoDB, οι βάσεις δεδομένων περιέχουν συλλογές εγγράφων. Οι συλλογές μοιάζουν με πίνακες σε σχεσιακές βάσεις δεδομένων, με τη διαφορά ότι δεν απαιτείται τα έγγραφα σε μια συλλογή να έχουν το ίδιο schema. Δηλαδή, τα έγγραφα σε μια συλλογή δεν χρειάζεται να περιέχουν τα ίδια πεδία, και ο τύπος δεδομένων ενός πεδίου μπορεί να διαφέρει από έγγραφο σε έγγραφο μέσα στην ίδια συλλογή[20].
Για την εγκατάσταση της έκδοσης 6.0.4 της MongoDB, ακολουθήθηκαν τα εξής βήματα.

```
sudo apt-get update

sudo apt-get install -y mongodb-org
```

Έπειτα πρέπει να διευθετηθεί το θέμα των Transparent Huge Pages (THP). Οι Transparent Huge Pages είναι μια δυνατότητα διαχείρισης μνήμης στο Linux που έχει σχεδιαστεί για να μειώνει το κόστος των αναζητήσεων στον translation lookaside buffer (page table) σε συστήματα με μεγάλη ποσότητα μνήμης, χρησιμοποιώντας μεγαλύτερες σελίδες εικονικής μνήμης. Ωστόσο, οι THP συχνά προκαλούν προβλήματα απόδοσης σε βάσεις δεδομένων, καθώς αυτές συνήθως έχουν διάσπαρτα μοτίβα πρόσβασης στη μνήμη αντί για συνεχόμενα. Η MongoDB, Inc. συνιστά την απενεργοποίηση των THP σε διακομιστές που εκτελούν την MongoDB.
Η απενεργοποίηση των THP επιτυγχάνεται με την δημιουργία του αρχείου /etc/systemd/system /mongodb-disable-thp.service και την εκτέλεση των εντολών restorecon και chmod, οι οποίες χρησιμοποιούνται για τη διαχείριση των αδειών και της ασφάλειας των αρχείων.

```
[Unit]
Description="Disable Transparent Huge Pages (THP) before mongodb
   ↪ starts"
Before=mongodb.service

[Service]
Type=oneshot
ExecStart=/bin/sh -c 'echo never > /sys/kernel/mm/
   ↪ transparent_hugepage/enabled'
ExecStart=/bin/sh -c 'echo never > /sys/kernel/mm/
   ↪ transparent_hugepage/defrag'

[Install]
RequiredBy=mongodb.service
```

```
restorecon /etc/systemd/system/mongodb-disable-thp.service

chmod 644 /etc/systemd/system/mongodb-disable-thp.service
```

Στην συνέχεια ακολουθεί το άνοιγμα του port 27017, το οποίο χρησιμοποιεί η MongoDB, με την δημιουργία του αρχείου /etc/firewalld/services/mongodb.xml και την εγκατάσταση του ως service του συστήματος.

```
<?xml version="1.0" encoding="utf-8"?>
<service>
    <short>mongodb</short>
    <description>MongoDB default port for mongodb and mongosh
       ↪ instances.</description>
<port protocol="tcp" port="27017"/>
</service>
```

```
restorecon /etc/firewalld/services/mongodb.xml

chmod 640 /etc/firewalld/services/mongodb.xml

firewall-cmd --reload
```

Τέλος, δημιουργήθηκε μια βάση δεδομένων ειδικά για το CAS, όπου ο διακομιστής CAS θα μπορεί να αποθηκεύει όλα τα δεδομένα του. Κάθε ένα από τα διαφορετικά modules (όπως το ticket registry, το service registry κ.λπ.) θα αποθηκεύει τις πληροφορίες του σε ξεχωριστή συλλογή μέσα σε αυτή τη βάση δεδομένων. Επιπλέον, δημιουργήθηκε ένας χρήστης, χωρίς δικαιώματα διαχειριστή, που θα χρησιμοποιείται από τον διακομιστή CAS για την πρόσβαση σε αυτές τις συλλογές.

```
sudo mongosh

use casdb
db.createUser( { user: "mongocas", pwd: "******", roles: [ { role:
   ↪ "readWrite", db: "casdb" } ] } )
```

Εφόσον ολοκληρωθεί επιτυχώς η εγκατάσταση της MongoDB, επόμενο βήμα αποτελεί η ενσωμάτωση του module στον CAS server, με την εισαγωγή του dependecy και την παραμετροποίηση των κατάλληλων properties στο αρχείο cas.properties.

```
implementation "org.apereo.cas:cas-server-support-mongo-service-
   ↪ registry:${project.'cas.version'}"
```

```
#MongoDB Service Registry
cas.service-registry.mongo.client-uri= mongodb:// mongocas:*****
   ↪ @cas.iee.ihu.gr:27017/casdb?directConnection=true&ssl=false

cas.service-registry.mongo.database-name=                casdb
cas.service-registry.mongo.host=                         cas.iee.
   ↪ ihu.gr
cas.service-registry.mongo.password=                     *****
cas.service-registry.mongo.port=                         27017
cas.service-registry.mongo.user-id=                      mongocas
cas.service-registry.mongo.collection:
   ↪ casServiceRegistry
cas.service-registry.mongo.ssl-enabled=
   ↪ false
cas.service-registry.mongo.authentication-database-name= casdb
```

Ticket Registry

Το ticket registry είναι μία βάση η οποία περιέχει όλα τα tickets και tokens που δημιουργούνται για την λειτουργία του SSO (Ticket-Granting Tickets, Service Tickets, Access Tokens κλπ.) και χρησιμοποιούνται για την αυθεντικοποίηση.
Οι διαθέσιμες επιλογές για την βάση στην οποία θα αποθηκεύονται τα tickets είναι παρόμοιες με αυτές που προσφέρονται και για το service registry. Ωστόσο, η χρήση μιας σχεσιακής βάσης δεδομένων μέσω του JPA μπορεί να αποδειχθεί ιδιαίτερα απαιτητική, λόγω της φύσης των δεδομένων που πρέπει να αποθηκευτούν. Με βάση αυτά τα κριτήρια, η επιλογή της MongoDB για τη διαχείριση και του μητρώου των εισιτηρίων ήταν η πιο λογική, καθώς απλοποιεί τη διαδικασία, εξαλείφοντας την ανάγκη για τη δημιουργία και συντήρηση μιας διαφορετικής βάσης δεδομένων.

Η διαδικασία ενσωμάτωσης του MongoDB ticket registry είναι όμοια με αυτήν της προηγούμενης ενότητας, και υλοποιείται ως εξής.

```
implementation "org.apereo.cas:cas-server-support-mongo-ticket-
    ↪ registry:${project.'cas.version'}"
```

```
#MongoDB Ticket Registry
cas.ticket.registry.mongo.client-uri= mongodb://mongocas:*****@cas.
    ↪ iee.ihu.gr:27017/casdb?directConnection=true

cas.ticket.registry.mongo.database-name=                casdb
cas.ticket.registry.mongo.host=                         cas.iee.ihu
    ↪ .gr
cas.ticket.registry.mongo.password=                     ******
cas.ticket.registry.mongo.port=                         27017
cas.ticket.registry.mongo.user-id=                      mongocas
```

Audit Registry

Το Audit Registry ενός διακομιστή CAS είναι ένα σημαντικό εργαλείο για την παρακολούθηση και καταγραφή των γεγονότων που σχετίζονται με την ασφάλεια και την πρόσβαση. Καταγράφει λεπτομερώς τις δραστηριότητες που σχετίζονται με την αυθεντικοποίηση, τις εξουσιοδοτήσεις και τις αιτήσεις υπηρεσιών, προσφέροντας έτσι μια σαφή εικόνα της χρήσης του συστήματος. Αυτό επιτρέπει στους διαχειριστές να εντοπίζουν και να αναλύουν πιθανά προβλήματα ασφάλειας, διασφαλίζοντας τη διαφάνεια και τη συμμόρφωση με τις πολιτικές ασφαλείας του οργανισμού. Το CAS, ως προεπιλογή, χρησιμοποιεί το σύστημα αρχείων του διακομιστή για την αποθήκευση των audits. Αυτή η υλοποίηση κρίνεται επαρκής για διακομιστές μικρού μεγέθους, όπως αυτός της παρούσας εργασίας, ωστόσο παρέχεται η δυνατότητα αποθήκευσης σε διάφορες βάσεις δεδομένων όπως αυτές που αναφέρθηκαν στις προηγούμενες ενότητες.

Log Registry

Το CAS παρέχει την δυνατότητα καταγραφής σημαντικών ενημερωτικών γεγονότων σε μορφή αρχείων, όπως επιτυχία και αποτυχία αυθεντικοποίησης, και μπορεί να προσαρμοστεί ώστε να παράγει πρόσθετες πληροφορίες για την αντιμετώπιση προβλημάτων. Τα παραπάνω αποθηκεύονται στο log registry, το οποίο βρίσκεται τοπικά στο file system του server. Deployments σαν το συγκεκριμένο, όπου δεν υπάρχουν πολλαπλοί διακομιστές ή load balancers, δεν χρειάζονται απαραίτητα τη χρήση κάποιας τρίτης υπηρεσίας η οποία θα αναλαμβάνει τη συγκέντρωση και την καλύτερη διαχείριση των αρχείων.

4.3.5 CAS Management Webapp

Η τελευταία σημαντική προσθήκη στο πρότζεκτ είναι το CAS Management Webapp, μια εφαρμογή σχεδιασμένη για τη διαχείριση και διαμόρφωση του CAS, όπως έχει ήδη αναφερθεί.
Το CAS Management Webapp είναι μια διαδικτυακή διεπαφή (GUI) που επιτρέπει στους διαχειριστές του CAS να δημιουργούν, να τροποποιούν και να διαγράφουν ορισμούς υπηρεσιών

στο service registry. Παράλληλα, προσφέρει μια οπτική αναπαράσταση διαφόρων μετρήσεων σχετικά με την κατάσταση του CAS server. Πρόκειται για μια ανεξάρτητη εφαρμογή, η οποία αναπτύσσεται ξεχωριστά από τον CAS server. Μπορεί να αναπτυχθεί είτε στον ίδιο Java Servlet container που εκτελεί τον διακομιστή CAS, είτε σε έναν εντελώς ξεχωριστό container. Σε κάθε περίπτωση, η λειτουργία του CAS δεν εξαρτάται από την κατάσταση του management webapp. Η εφαρμογή αυτή είναι ιδιαίτερα σημαντική όταν χρησιμοποιούνται βάσεις δεδομένων ή άλλα συστήματα αποθήκευσης για το service registry. Σε αυτήν τη διαμόρφωση, το management webapp λειτουργεί ως η διεπαφή για τις λειτουργίες CRUD (δημιουργία, ανάγνωση, ενημέρωση, διαγραφή) που αφορούν το σύστημα αποθήκευσης στο παρασκήνιο.
Για την εγκατάσταση και παραμετροποίηση της εφαρμογής θα ακολουθηθούν παρόμοια βήματα με αυτά που ακολουθήθηκαν για τον CAS server. Η διαδικασία ξεκινά με την επιλογή της κατάλληλης έκδοσης, η οποία για να είναι συμβατή με την έκδοση 7.0.0 του διακομιστή, θα πρέπει να είναι η 7.0.0. Στη συνέχεια, ακολουθεί η λήψη του πρότζεκτ από το GitHub, το οποίο επίσης προσφέρεται στην μορφή ενός WAR (Web Application Archive) overlay.

```
git clone https://github.com/apereo/cas-management-overlay.git

git checkout 7.0.x
git checkout -b cas.iee.ihu.gr
```

Αρχική Παραμετροποίηση

Η παραμετροποίηση του CAS management webapp μπορεί να ελεγχθεί σχεδόν εξ ολοκλήρου μέσω μεταβλητών, οι οποίες αποθηκεύονται στο αρχείο management.properties, το οποίο βρίσκεται στον φάκελο /etc/cas/config. Το αρχικό περιεχόμενο του αρχείου έχει την εξής μορφή:

```
#Server conf
cas.server.name=https://cas.iee.ihu.gr:8443
cas.server.prefix=${cas.server.name}/cas
mgmt.server-name=${cas.server.name}
mgmt.server-prefix=https://cas.iee.ihu.gr:8443/cas-management

logging.config=file:/etc/cas/config/log4j2-management.xml
```

Οι πρώτες τέσσερις μεταβλητές δηλώνουν τα URLs του CAS server (με τη σειρά protocol, domain name, και port) και του management webapp αντίστοιχα. Στην προκειμένη περίπτωση, η εφαρμογή θα εγκατασταθεί στον ίδιο διακομιστή μέσω του Tomcat, επομένως οι τιμές των URLs θα είναι οι ίδιες.
Η μεταβλητή logging.config ορίζει το αρχείο log4j2-management.xml ως την πηγή των ρυθμίσεων για την καταγραφή των logs.

Εγκατάσταση μέσω του Tomcat

Παρακάτω αναγράφεται η διαδικασία εγκατάστασης μέσω του Apache Tomcat 11, ακολουθώντας την ίδια διαδικασία αποσυμπίεσης του WAR και την μεταφορά των αρχείων στον κατάλογο webapps του Tomcat .

```
cd cas-management-overlay

./gradlew clean copyCasConfiguration build
./gradlew unzipWAR

cd cas-management-overlay/build

tar czf /tmp/casmngt-files.tgz --exclude app/META-INF --owner=root
   ↪ --group=tomcat --mode=g-w,o-rwx app

systemctl stop tomcat

cd /opt/tomcat/latest/webapps
tar xzf /tmp/casmngt-files.tgz
mv app cas

systemctl start tomcat
```

Όπως και πριν, η επανάληψη της διαδικασίας κάθε φορά που γίνονται προσθήκες ή αλλαγές στην εφαρμογή, είναι αναγκαία.

Προσθήκη του MongoDB Service Registry

Ένας από τους κύριους ρόλους του webapp είναι η διαχείριση του μητρώου των υπηρεσιών, επομένως είναι αυτονόητο ότι θα πρέπει να έχει πρόσβαση και να χρησιμοποιεί την ίδια βάση με τον CAS server. Για την υλοποίηση των παραπάνω, θα χρειαστούν οι εξής προσθήκες στα αρχεία build.gradle και management.properties αντίστοιχα.

```
implementation "org.apereo.cas:cas-server-support-mongo-service-
   ↪ registry:${project.'cas.version'}"
```

```
#MongoDB Service Registry
cas.service-registry.mongo.client-uri= mongodb:// mongocas:*****
   ↪ @cas.iee.ihu.gr:27017/casdb?directConnection=true&ssl=false

cas.service-registry.mongo.database-name=                casdb
cas.service-registry.mongo.host=                         cas.iee.
   ↪ ihu.gr
cas.service-registry.mongo.password=                     *****
cas.service-registry.mongo.port=                         27017
cas.service-registry.mongo.user-id=                      mongocas
cas.service-registry.mongo.collection:
   ↪ casServiceRegistry
cas.service-registry.mongo.ssl-enabled=
   ↪ false
cas.service-registry.mongo.authentication-database-name= casdb
```

Προσθήκη ενός Stub Attribute Repository

Η εφαρμογή διαχείρισης επιτρέπει τη ρύθμιση πολιτικών έκδοσης attributes για κάθε υπηρεσία ξεχωριστά. Για να γνωρίζει ποια attributes είναι διαθέσιμα προς έκδοση, πρέπει να διαμορφωθεί ένα ειδικό ”stub” attribute repository, το οποίο θα περιλαμβάνει τα ονόματα όλων των διαθέσιμων attributes.
Χωρίς αυτή τη ρύθμιση, η εφαρμογή διαχείρισης θα μπορεί να διαμορφώσει μόνο πολιτικές έκδοσης ”όλων” ή ”κανενός”, χωρίς τη δυνατότητα επιλογής συγκεκριμένων attributes προς έκδοση.
Οι παρακάτω προσθήκες είναι αναγκαίες στο αρχείο management.properties.

```
#STUB Repository
cas.authn.attributeRepository.stub.attributes.cn=                cn
cas.authn.attributeRepository.stub.attributes.displayName=
   ↪ displayName
cas.authn.attributeRepository.stub.attributes.givenName=
   ↪ givenName
cas.authn.attributeRepository.stub.attributes.mail=
   ↪ mail
cas.authn.attributeRepository.stub.attributes.sn=                sn
cas.authn.attributeRepository.stub.attributes.uid=               uid
cas.authn.attributeRepository.stub.attributes.group=group
cas.authn.attributeRepository.stub.attributes.am=am
cas.authn.attributeRepository.stub.attributes.eduPersonAffiliation=
   ↪ eduPersonAffiliation
cas.authn.attributeRepository.stub.attributes.
   ↪ eduPersonScopedAffiliation=eduPersonScopedAffiliation
cas.authn.attributeRepository.stub.attributes.eduPersonEntitlement=
   ↪ eduPersonEntitlement
cas.authn.attributeRepository.stub.attributes.
   ↪ eduPersonPrimaryAffiliation=eduPersonPrimaryAffiliation
cas.authn.attributeRepository.stub.attributes.cn;lang-el=cn;lang-el
cas.authn.attributeRepository.stub.attributes.description=
   ↪ description
cas.authn.attributeRepository.stub.attributes.description;lang-el=
   ↪ description;lang-el
cas.authn.attributeRepository.stub.attributes.fathersname=
   ↪ fathersname
cas.authn.attributeRepository.stub.attributes.fathersname;lang-el=
   ↪ fathersname;lang-el
cas.authn.attributeRepository.stub.attributes.givenName;lang-el=
   ↪ givenName;lang-el
cas.authn.attributeRepository.stub.attributes.regsem=regsem
cas.authn.attributeRepository.stub.attributes.regyear=regyear
cas.authn.attributeRepository.stub.attributes.labeledURI=labeledURI
cas.authn.attributeRepository.stub.attributes.pwdChangedTime=
   ↪ pwdChangedTime
cas.authn.attributeRepository.stub.attributes.secondarymail=
```

```
    ↪ secondarymail
cas.authn.attributeRepository.stub.attributes.sem=sem
cas.authn.attributeRepository.stub.attributes.sn;lang-gr=sn;lang-el
cas.authn.attributeRepository.stub.attributes.telephoneNumber=
    ↪ telephoneNumber
cas.authn.attributeRepository.stub.attributes.title=title
cas.authn.attributeRepository.stub.attributes.title;lang-el=title;
    ↪ lang-el
cas.authn.attributeRepository.stub.attributes.profilePhoto=
    ↪ profilePhoto
cas.authn.attributeRepository.stub.attributes.socialMedia=
    ↪ socialMedia
```

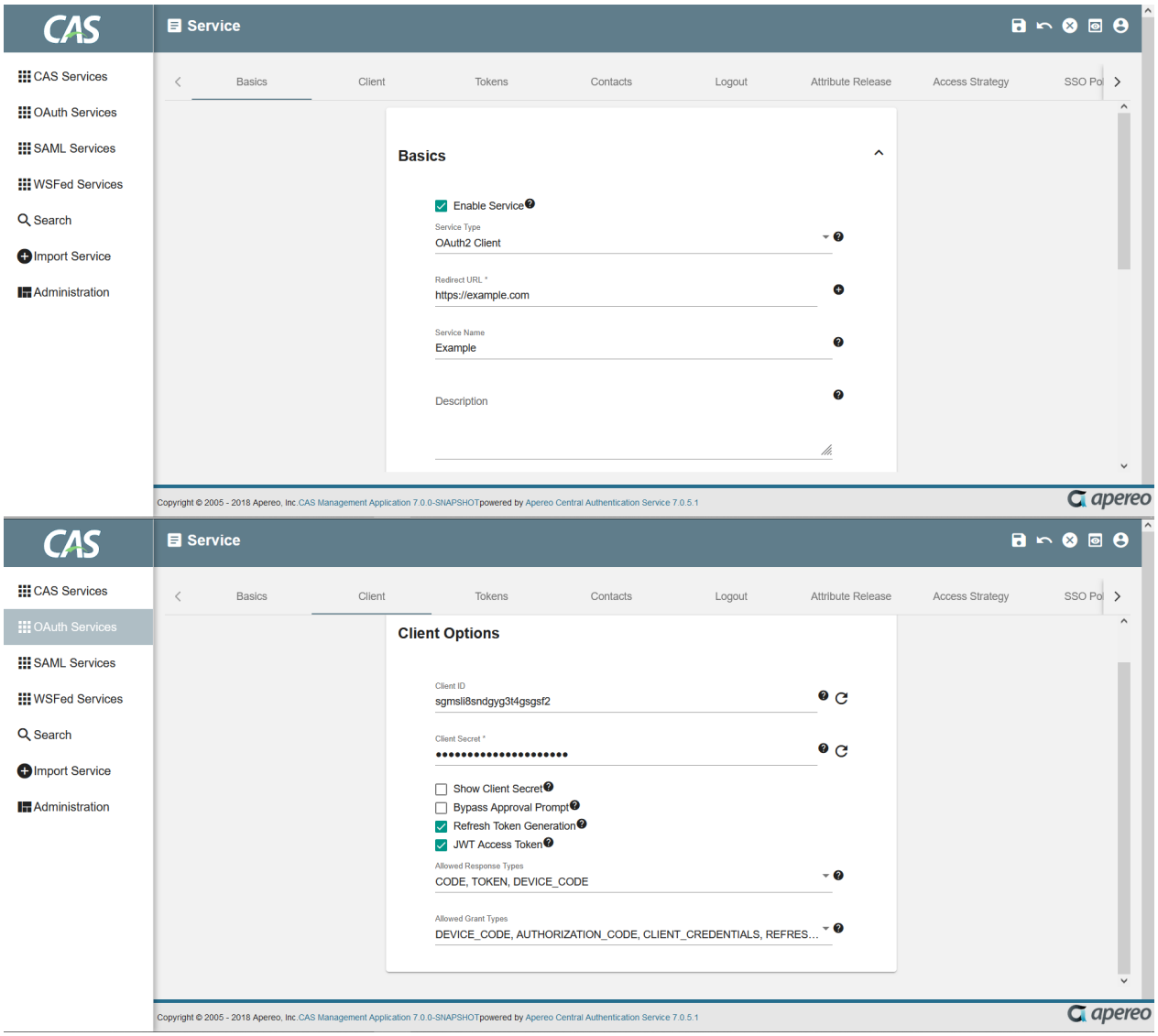
Δημιουργία Δήλωσης Υπηρεσίας για το Management Webapp

Η πρόσβαση στην εφαρμογή θα γίνεται μέσω του συστήματος SSO του CAS, επομένως είναι απαραίτητο να δημιουργηθεί η κατάλληλη εγγραφή στο μητρώο υπηρεσιών. Σε αντίθεση με άλλες εφαρμογές, το Management Webapp θα χρησιμοποιεί το πρωτόκολλο CAS για την επικοινωνία του με τον διακομιστή, αντί του OAuth 2.0. Αυτό οφείλεται στο γεγονός ότι η εφαρμογή αποτελεί "γηγενές" μέρος του οικοσυστήματος του CAS, καθιστώντας την υλοποίηση πολύ πιο απλή, καθώς λειτουργεί άμεσα χωρίς πρόσθετες ρυθμίσεις.

```
{
    "@class" : "org.apereo.cas.services.CasRegisteredService",
    "serviceId" : "^https://cas.iee.ihu.gr:8443/cas-management.*",
    "name" : "CAS Services Management",
    "id" : 1713004100,
    "description" : "CAS services management webapp",
    "attributeReleasePolicy" : {
        "@class" : "org.apereo.cas.services.
            ↪ ReturnAllAttributeReleasePolicy"
    },
    "evaluationOrder" : 5500
}
```

4.4 Ενοποίηση του CAS με τις Υπάρχουσες Εφαρμογές

Η διαδικασία της ενοποίησης του CAS με τις υπάρχουσες εφαρμογές αποτελεί μία απλή αλλά χρονοβόρα διαδικασία, η οποία αποτελείται από δύο βήματα.
Το πρώτο βήμα περιλαμβάνει τη δημιουργία μιας εγγραφής στο service registry μέσω του γραφικού περιβάλλοντος διαχείρισης που προσφέρεται από το management webapp, το οποίο απλοποιεί τη διαδικασία και διασφαλίζει τη σωστή διαμόρφωση της εγγραφής.
Το δεύτερο βήμα αφορά την παραμετροποίηση των εφαρμογών, έτσι ώστε να χρησιμοποιούν τα endpoints του CAS server για την υλοποίηση του πρωτοκόλλου OAuth 2.0, αντί για αυτά του



Σχήμα 4.1: Management App GUI για την δημιουργία service definitions

προηγούμενου συστήματος. Η πολυπλοκότητα των αλλαγών αυτών εξαρτάται από την γλώσσα προγραμματισμού και τις τεχνολογίες τις οποίες χρησιμοποιεί η εκάστοτε εφαρμογή.
Ανάμεσα στην υλοποίηση του OAuth 2.0 από τον CAS και το υπάρχον σύστημα του τμήματος, υπάρχουν ορισμένες διαφορές. Οι διαφορές αυτές οφείλονται κυρίως στη μορφή του access token που παράγει το API, το οποίο είναι αρκετά απλό και στερείται ορισμένων παραμέτρων. Παρακάτω αναφέρονται οι δύο σημαντικότερες από αυτές:

- Απουσία της παραμέτρου ”type” από το access token που δημιουργεί το API, η οποία θα πρέπει να λαμβάνει την τιμή ”bearer”.
- Απαίτηση για προσθήκη του -H ”x-access-token: ACCESS_TOKEN” ως header στην εντολή POST κατά την κλήση του API με το access token.

Παρά τις διαφορές αυτές, η μετάβαση στο CAS, το οποίο υλοποιεί το πρωτόκολλο με έναν λιγότερο ”απαιτητικό” τρόπο, δεν αναμένεται να προκαλέσει σημαντικά προβλήματα. Ωστόσο, ενδέχεται να χρειαστούν κάποιες προσαρμογές στις εφαρμογές για την καλύτερη αξιοποίηση των διαφορετικών access tokens που παράγει ο διακομιστής CAS.

Κεφάλαιο 5

Συμπεράσματα και Συστάσεις

5.1 Συμπεράσματα

Το Central Authentication Service (CAS) αποδείχθηκε μια αποτελεσματική και ευέλικτη λύση για την υλοποίηση ενός συστήματος Single Sign-On (SSO) στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Οι δοκιμές απέδειξαν ότι το CAS μπορεί να ενσωματωθεί με επιτυχία στις υπάρχουσες εφαρμογές και πλατφόρμες της σχολής, καλύπτοντας πλήρως τις υφιστάμενες ανάγκες αυθεντικοποίησης αλλά και την ανάγκη εκσυχρονισμού και συντήρησης του υπάρχοντος συστήματος. Με βάση την πορεία και εξέλιξη του CAS έως σήμερα, η υιοθέτησή του αποτελεί μια εξαιρετική επένδυση για το μέλλον. Καθώς διαθέτει μια δυναμική και ενεργή κοινότητα ανάπτυξης, η οποία με τη συνεχή βελτίωση και την προσθήκη νέων λειτουργιών, εξασφαλίζει ότι το CAS θα παραμείνει στην αιχμή της τεχνολογίας.

5.2 Συστάσεις για τη Μελλοντική Ανάπτυξη

Κατά τη διάρκεια της εργασίας διερευνήθηκαν διάφορες δυνατότητες του CAS οι οποίες δεν υλοποιήθηκαν, κυρίως λόγω του χρονικού περιορισμού.

Διαμόρφωση του User Interface (UI) Ένα από τα βασικά παραδείγματα, τέτοιων δυνατοτήτων, αποτελεί η διαμόρφωση του User Interface (UI) έτσι ώστε να αντικατοπτρίζει την ταυτότητα του τμήματος.
Ο διακομιστής CAS χρησιμοποιεί ένα σύνολο αρχείων HTML, Thymeleaf templates, αρχεία CSS και JavaScript, καθώς και πακέτα μηνυμάτων Java για να υλοποιήσει τη σελίδα σύνδεσης και τις υπόλοιπες σελίδες. Το CAS αξιοποιεί το Spring Web Flow για την επεξεργασία των αιτημάτων σύνδεσης και αποσύνδεσης, καθώς και προσθήκες όπως ο έλεγχος ταυτότητας πολλών παραγόντων (multi-factor authentication).
Η διεπαφή χρήστη του CAS χωρίζεται σε τρία κύρια μέρη:

- Views: Μια σειρά αρχείων HTML, ένα για κάθε κατάσταση προβολής. Οι σελίδες προβολής επεξεργάζονται μέσω του Thymeleaf template engine, επιτρέποντάς τους να έχουν δυναμική πρόσβαση σε ρυθμίσεις και μεταβλητές ιδιοτήτων, καθώς και να αξιολογούν

Σχήμα 5.1: Πρότυπο login view

λογικές εκφράσεις με βάση τις τιμές τους. Το Thymeleaf δίνει επίσης τη δυνατότητα ορισμού ενός κοινού προτύπου διάταξης (για το φόντο της σελίδας, την κεφαλίδα, το υποσέλιδο, τα στοιχεία πλοήγησης, την ενσωμάτωση style sheets, κ.λπ.) που θα μοιράζεται από όλες τις σελίδες προβολής.

- Themes: Μια συλλογή από style sheets CSS, αρχεία JavaScript και εικόνες που περιλαμβάνονται στις σελίδες προβολής. Τα αρχεία CSS ελέγχουν τα χρώματα, τις γραμματοσειρές και άλλες στιλιστικές πτυχές της διεπαφής, ενώ τα αρχεία JavaScript ορίζουν τις διαδικασίες που εκτελούνται στο πρόγραμμα περιήγησης του χρήστη, όπως ανίχνευση ενεργοποίησης Caps Lock, έλεγχο ταυτότητας πολλών παραγόντων κ.ά. Τα αρχεία εικόνων περιλαμβάνουν λογότυπα, κουμπιά, γραμμές κ.λπ., που χρησιμοποιούνται από τα style sheets.

- Πακέτα μηνυμάτων (Message bundles): Αρχεία ιδιοτήτων Java που ορίζουν όλα τα κείμενα που εμφανίζονται στα διάφορα views. Η μηχανή προτύπων Thymeleaf εισάγει αυτόματα τα μηνύματα από το κατάλληλο πακέτο μηνυμάτων, βάσει των τοπικών ρυθμίσεων, στα HTML views.

Κάθε φορά που το web flow εισέρχεται σε μια κατάσταση προβολής, το CAS εμφανίζει μια δυναμικά δημιουργημένη ιστοσελίδα ή ένα συγκεκριμένο view. Το CAS χρησιμοποιεί το Thymeleaf template engine για τη διαχείριση αυτής της δυναμικής δημιουργίας περιεχομένου. Συγκεκριμένα, το Thymeleaf Layout Dialect είναι μια επέκταση του Thymeleaf που επιτρέπει την κατασκευή διατάξεων και επαναχρησιμοποιούμενων προτύπων, διευκολύνοντας την επαναχρησιμοποίηση κώδικα. Η επέκταση αυτή χρησιμοποιεί δύο βασικά στοιχεία: τις διατάξεις (layouts) και τα πρότυπα περιεχομένου (content templates).
Μια διάταξη είναι ένα αρχείο HTML που ορίζει κοινά στοιχεία για όλες τις ιστοσελίδες που χρησιμοποιούν αυτή τη διάταξη, όπως η κεφαλίδα, το υποσέλιδο, το μενού, η πλοήγηση κ.λπ. Ενώ τα πρότυπα περιεχομένου παρέχουν το περιεχόμενο που αντικαθίσταται μέσα στη διάταξη.

Για την επίτευξη της εξατομίκευσης του user interface θα πρέπει να δημιουργηθεί ένα καινούργιο theme, με την εξής εντολή.

```
cd cas-overlay-template

./gradlew createTheme -Ptheme=iee
```

Το οποίο θα περιλαμβάνει τα παρακάτω αρχεία, μέσω της κατάλληλης διαμόρφωσης των οποίων θα επιτευχθεί η επιθυμητή εμφάνιση του UI.

```
├── iee.properties
├── static
│   └── themes
│       └── iee
│           ├── css
│           │   └── cas.css
│           └── js
│               └── cas.js
└── templates
    └── iee
        └── fragments
```

Εφόσον τελειώσει η διαδικασία διαμόρφωσης, θα χρειαστεί η προσθήκη της παρακάτω μεταβλητής στο αρχείο cas.properties, για την χρήση του καινούργιου theme από τον CAS server.

```
cas.theme.defaultThemeName=iee
```

Δυνατότητα Password Management Μια ακόμη δυνατότητα που διερευνήθηκε, αν και δεν ενσωματώθηκε στην πιλοτική εφαρμογή του CAS, είναι η διαχείριση κωδικών πρόσβασης (password management). Κύριος ανασταλτικός παράγοντας για την προσθήκη της δυνατότητας αποτέλεσε η αδυναμία παροχής των κατάλληλων δικαιωμάτων στον λογαριασμό που χρησιμοποιεί το CAS για την προσπέλαση των καταλόγων του LDAP. Η προσθήκη του password management κρίνεται ιδιαίτερα σημαντική, καθώς οι κωδικοί πρόσβασης που είναι αποθηκευμένοι στον LDAP έχουν ημερομηνία λήξης και χρειάζονται τακτική ενημέρωση. Ακόμη, μέσω των ίδιων ρυθμίσεων καθίσταται δυνατή και η ανάκτηση του κωδικού πρόσβασης, σε περίπτωση που ο χρήστης τον έχει ξεχάσει, λειτουργία που κρίνεται εξαιρετικής σημασίας.
Τα βήματα για την ενσωμάτωση του password management ακολουθούν την ήδη καθιερωμένη διαδικασία της προσθήκης των κατάλληλων dependencies στο πρότζεκτ και την παραμετροποίηση ορισμένων ιδιοτήτων στο αρχείο cas.properties.

```
implementation "org.apereo.cas:cas-server-support-pm-webflow:${
   ↪ project.'cas.version'}"
implementation "org.apereo.cas:cas-server-support-pm-ldap:${project
   ↪ .'cas.version'}"
```

Οι παρακάτω ιδιότητες ενεργοποιούν το module του password management, θέτουν ένα ισχυρό password policy (τουλάχιστον οκτώ χαρακτήρες εκ των οποίων ένας πεζός χαρακτήρας, ένας

κεφαλαίος, ένα νούμερο και ένα ειδικό σύμβολο) και ορίζουν τα κλειδιά της κρυπτογραφικής διαδικασίας για την αλλαγή των κωδικών πρόσβασης. Παρέχεται επίσης η δυνατότητα ορισμού ερωτήσεων ασφαλείας, αν και στην τρέχουσα διαμόρφωση αυτή η λειτουργία είναι απενεργοποιημένη. Ενώ ορίζονται ξανά οι λεπτομέρειες του καταλόγου του LDAP σε σχέση με το password management. Τέλος, παραμετροποιούνται οι μεταβλητές που χρησιμοποιούνται για τη σύνταξη και την αποστολή των αυτοματοποιημένων emails για την ανάκτηση των χαμένων κωδικών πρόσβασης.

```
#LDAP Password Management
CasFeatureModule.AccountManagement.enabled=true
cas.authn.pm.reset.security-questions-enabled=false
cas.authn.pm.core.enabled=true
cas.authn.pm.core.password-policy-pattern=^(?=.*[a-z])(?=.*[A-Z])
   ↪ (?=.*\d)(?=.*[$@$!%*?&])[A-Za-z\d$@$!%*?&]{8,}$
cas.authn.pm.reset.crypto.encryption.key=dPj**************
cas.authn.pm.reset.crypto.signing.key=mUF****************

cas.authn.pm.reset.mail.attribute-name=mail
cas.authn.pm.reset.mail.from=admin@cas.iee.ihu.gr
cas.authn.pm.reset.mail.subject=Password Reset
#cas.authn.pm.reset.mail.text=
spring.mail.host=cas.iee.ihu.gr
spring.mail.username=casadmin
spring.mail.password=*******
spring.mail.port=587
spring.mail.protocol=smtp

cas.authn.pm.ldap[0].ldap-url=ldap://ldap.it.teithe.gr:389
cas.authn.pm.ldap[0].account-locked-attribute=pwdLockout
cas.authn.pm.ldap[0].base-dn=ou=people,dc=it,dc=teithe,dc=gr
cas.authn.pm.ldap[0].bind-dn=cn=casadmin,dc=it,dc=teithe,dc=gr
cas.authn.pm.ldap[0].bind-credential=*****
cas.authn.pm.ldap[0].search-filter=uid={user}
cas.authn.pm.ldap[0].type=GENERIC
cas.authn.pm.ldap[0].username-attribute=uid
```

Μελλοντική Έρευνα και Ανάπτυξη Επιπροσθέτως για την μελλοντική έρευνα και ανάπτυξη, προτείνεται η περαιτέρω εξερεύνηση των δυνατοτήτων του CAS όπως ο έλεγχος ταυτότητας πολλών παραγόντων (Multi Factor Authentication - MFA), το GeoLocation Tracking και η ενσωμάτωση κάποιου Identity Management tool.
Ο έλεγχος ταυτότητας πολλών παραγόντων (Multi-Factor Authentication - MFA) είναι μια μέθοδος ασφαλείας που απαιτεί περισσότερα από ένα στοιχεία για να επιβεβαιώσει την ταυτότητα ενός χρήστη κατά την είσοδό του σε μια εφαρμογή ή σύστημα. Εκτός από τον παραδοσιακό κωδικό πρόσβασης, το MFA ζητά πρόσθετες αποδείξεις όπως έναν κωδικό που αποστέλλεται στο κινητό, βιομετρικά δεδομένα (π.χ. δακτυλικό αποτύπωμα), ή έναν φυσικό κωδικό ασφαλείας. Με αυτόν τον τρόπο μειώνεται σημαντικά ο κίνδυνος παραβίασης λογαριασμών, καθώς

ακόμη και αν έχει κλαπεί ο κωδικός πρόσβασης, καθίσταται πολύ πιο δύσκολο για κακόβουλους χρήστες να αποκτήσουν πρόσβαση.
Παράλληλα, το GeoLocation Tracking προσφέρει επιπλέον δυνατότητες ασφάλειας μέσω της παρακολούθησης της γεωγραφικής θέσης ενός ατόμου ή συσκευής σε πραγματικό χρόνο. Αξιοποιώντας δεδομένα από Wi-Fi, κινητά δίκτυα ή IP διευθύνσεις, η τεχνολογία αυτή επιτρέπει τον ακριβή προσδιορισμό της τοποθεσίας του χρήστη. Η ενσωμάτωση του στο πρότζεκτ θα συμβάλει στην ενίσχυση της ασφάλειας. παρέχοντας τη δυνατότητα καταγραφής της τοποθεσίας κάθε σύνδεσης. Παράλληλα, θα ειδοποιεί τους χρήστες για τυχόν νέες συνδέσεις από άγνωστες τοποθεσίες, οι οποίες ενδέχεται να μην προέρχονται από τους ίδιους, εξασφαλίζοντας έτσι την ασφάλεια των λογαριασμών τους.
Μία δυνατότητα η οποία θα προσέθετε ακόμη περισσότερη χρηστικότητα στο σύστημα του CAS, θα ήταν η ενσωμάτωση του με κάποιο Identity Management tool. Μία εφαρμογή δηλαδή που θα επέτρεπε την διαχείριση των λογαριασμών των χρηστών σε administrative επίπεδο, δυνατότητα υψηλής σημασίας για την σωστή λειτουργία του συστήματος, την οποία δυστυχώς το ίδιο το CAS δεν παρέχει. Η υλοποίηση μίας δυνατότητας όπως η παραπάνω, πιθανότατα θα επιτευχθεί με την χρήση του CAS server ως SAML2.0 identity provider, θα χρησιμοποιηθεί δηλαδή το πρωτόκολλο SAML για την επικοινωνία ανάμεσα στον διακομιστή και τον client, και την εγκατάσταση της ανάλογης εφαρμογής που θα επιτρέπει την διαχείριση των λογαριασμών των χρηστών.
Τέλος, θα πρέπει να γίνει μία αναφορά στο CAS Management Webapp, το οποίο κατά την διάρκεια ανάπτυξης της συγκεκριμένης εργασίας υπέστη σημαντικές αλλαγές. Αποφασίστηκε ότι η λειτουργικότητα του θα μεταφερθεί στο κύριο πρότζεκτ του CAS, με την μορφή του module ”Palantir”, από την έκδοση 7.1.x και έπειτα. Μία από τις κύριες αλλαγές τις οποίες έφερε η παραπάνω αλλαγή, ακόμα και στην έκδοση 7.0.x, ήταν η δυνατότητα πρόσβασης στην εφαρμογή μόνο με την χρήση των διαπιστευτηρίων του spring admin. Απενεργοποιήθηκε δηλαδή η δυνατότητα σύνδεσης μέσω του συστήματος SSO, κάτι το οποίο αποτελεί πρόβλημα για την υλοποίηση που επιθυμούσε η παρούσα εργασία. Η δυνατότητα χρήσης του GUI για τη δημιουργία των service definitions από αυθεντικοποιημένους χρήστες, οι οποίοι επιθυμούσαν τη χρήση του CAS για την προστασία των εφαρμογών τους, ήταν σημαντική για την ευκολία της διαχείρισης του συστήματος. Η πρόσβαση στην εφαρμογή ελεγχόταν από ένα σύστημα ρόλων, το οποίο επέτρεπε στους απλούς χρήστες (ROLE_USER) να δημιουργούν καινούργιες δηλώσεις εφαρμογών, οι οποίες θα έπρεπε να εγκριθούν από κάποιον διαχειριστή (ROLE_ADMIN) πριν περάσουν σε ισχύ, ενώ επίσης περιόριζε την πρόσβαση στο πάνελ της διαχείρισης. Εφόσον η παραπάνω υλοποίηση δεν είναι δυνατόν να πραγματοποιηθεί, θα πρέπει να διερευνηθούν πιθανές εναλλακτικές που θα παρέχουν παρόμοιες δυνατότητες.

Βιβλιογραφία

[1] Shaikh, Naim and Kasat, Kishori and Jadhav, Smita, "Secured Authentication by Single Sign On (SSO): A Big Picture", στο *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2022, σσ. 951–955. doi: 10.1109/ICCCIS56430.2022.10037708.

[2] Neuman, B.C. and Ts'o, T., "Kerberos: an authentication service for computer networks", *IEEE Communications Magazine*, τόμ. 32, αρθμ. 9, σσ. 33–38, 1994. doi: 10.1109/35.312841.

[3] Hughes, John and Maler, Eve, "Security assertion markup language (saml) v2. 0 technical overview", *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, τόμ. 13, σ. 12, 2005.

[4] Hardt, Dick, "The OAuth 2.0 authorization framework", αδημοσίευτη ερευνητική εργασία, 2012.

[5] Fett, Daniel and Küsters, Ralf and Schmitz, Guido, "The web sso standard openid connect: In-depth formal security analysis and security guidelines", στο *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, IEEE, 2017, σσ. 189–202.

[6] M. Jones, J. Bradley και N. Sakimura, "Json web token (jwt)", αδημοσίευτη ερευνητική εργασία, 2015.

[7] Apereo Foundation, *Central Authentication Service (CAS)*, https://www.apereo.org/projects/cas, Accessed: 2024-08-08, 2024.

[8] Johnson, Rod and Hoeller, Juergen and Donald, Keith and Sampaleanu, Colin and Harrop, Rob and Risberg, Thomas and Arendsen, Alef and Davison, Darren and Kopylenko, Dmitriy and Pollack, Mark and others, "The spring framework-reference documentation", *interface*, τόμ. 21, σ. 27, 2004.

[9] Walls, Craig, *Spring Boot in action*. Simon και Schuster, 2015.

[10] Cantor, Scott and Scavo, Tom, "Shibboleth architecture", *Protocols and Profiles*, τόμ. 10, αρθμ. 16, σ. 29, 2005.

[11] Thorgersen, Stian and Silva, Pedro Igor, *Keycloak-identity and access management for modern applications: harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications*. Packt Publishing Ltd, 2021.

[12] Oracle, *Java Development Kit, Version 21.0.2*, https://www.oracle.com/java/technologies/javase-downloads.html, Accessed: 2024-08-12, 2024.

[13] Apache Software Foundation, *Apache Tomcat, Version 11*, https://tomcat.apache.org/download-11.cgi, Accessed: 2024-08-12, 2024.

[14] OpenSSL Project, *OpenSSL: The Open Source toolkit for SSL/TLS*, https://www.openssl.org, Version 3.3.0, 2023.

[15] Apache Software Foundation, *Apache Portable Runtime (APR) Library*, https://apr.apache.org, Version 1.7.4, 2022.

[16] T. A. S. Foundation, *Apache Log4j 2*, https://logging.apache.org/log4j/2.x/, Version 2.23.0, 2024.

[17] M. Jones, "JSON web key (JWK)", αδημοσίευτη ερευνητική εργασία, 2015.

[18] mkjwk.org, *Online JSON Web Key Generator*, https://mkjwk.org/, Accessed: 2024-08-28, 2024.

[19] S. Tuttle, A. Ehlenberger, R. Gorthi κ.ά., *Understanding LDAP-design and implementation*. IBM Redbooks, 2006.

[20] A. Chauhan, "A review on various aspects of MongoDB databases", *International Journal of Engineering Research & Technology (IJERT)*, τόμ. 8, αρθμ. 05, σσ. 90–92, 2019.