BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

ISO 9001 : 2008

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

HẢI PHÒNG 2013

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

TÌM HIỂU CƠ CHẾ ĐĂNG NHẬP MỘT LẦN (SINGLE SIGN ON) VÀ THỬ NGHIỆM DỰA TRÊN THƯ VIỆN PHPCAS

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công Nghệ Thông Tin

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

TÌM HIỂU CƠ CHẾ ĐĂNG NHẬP MỘT LẦN (SINGLE SIGN ON) VÀ THỬ NGHIỆM DỰA TRÊN THƯ VIỆN PHPCAS

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công Nghệ Thông Tin

Giáo viên hướng dẫn: Th.s Bùi Huy Hùng

Sinh viên thực hiện: Đào Văn Phong

Mã số sinh viên: 1351010001

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

CỘNG HÒA XA HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

-------o0o-------

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Sinh viên: Đào Văn Phong Mã SV: 1351010001

Lớp: CT1301 Ngành: Công Nghệ Thông Tin

Tên đề tài:Tìm hiểu cơ chế đăng nhập một lần (single sign on) và thử nghiệm dựa trên thư viện phpCAS.

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Nội dung

- Tìm hiểu về đăng nhập một lần (Single Sign On).
- Tìm hiểu về CAS (Central Authentication Service).
- Thử nghiệm, cài đặt CAS, kiểm thử với website PHP dựa trên thư viện phpCAS.
- Nghiêm túc thực hiện các nhiệm vụ và nội dung giáo viên hướng dẫn.

b. Các yêu cầu cần giải quyết

- Lý thuyết
  - Nắm được cơ sở lý thuyết của đăng nhập một lần (Single Sign On).
  - Nắm được quá trình cài đặt CAS và các thức triển khai Single Sign On.
- Thực nghiệm (chương trình)

Cài đặt CAS và thực nghiệm với website PHP

2. Các số liệu cần thiết để tính toán.

……

……

3. Địa điểm thực tập.

……

……

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Người hướng dẫn thứ nhất:

Họ và tên: Bùi Huy Hùng

Học hàm, học vị: Thạc sĩ

Cơ quan công tác: Trường Đại Học Dân Lập Hải Phòng

Nội dung hướng dẫn:

- Tìm hiểu về Single Sign On dựa trên Central Authentication Service
- Thử nghiệm với website PHP sử dụng thư viện phpCAS

Người hướng dẫn thứ hai:

Họ và tên: …….

Học hàm, học vị: …….

Cơ quan công tác: ……

Nội dung hướng dẫn: ……..

……

……

Đề tài tốt nghiệp được giao ngày….tháng….năm 2013.

Yêu cầu phải hoàn thành trước ngày….tháng….năm 2013.

Đã nhận nhiệm vụ: Đ.T.T.N	Đã nhận nhiệm vụ: Đ.T.T.N
Sinh viên	Cán bộ hướng dẫn Đ.T.T.N
Đào Văn Phong	Th.s Bùi Huy Hùng

Hải Phòng, ngày .....tháng.....năm 2013

HIỆU TRƯỞNG

GS.TS.NGƯT Trần Hữu Nghị

PHẦN NHẬN XÉT TÓM TẮT CỦA CÁN BỘ HƯỚNG DẪN

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp:

.....

.....

.....

.....

.....

Đánh giá chất lượng của đề tài tốt nghiệp (so với nội dung yêu cầu đã đề ra trong nhiệm vụ đề tài tốt nghiệp)

.....

.....

.....

.....

.....

.....

.....

3. Cho điểm của cán bộ hướng dẫn:

(Điểm ghi bằng số và chữ)

.....

.....

.....

Ngày.....tháng.....năm 2013

Cán bộ hướng dẫn chính

(Ký, ghi rõ họ tên)

PHẦN NHẬN XÉT ĐÁNH GIÁ CỦA CÁN BỘ CHẤM PHẢN BIỆN ĐỀ TÀI TỐT NGHIỆP

1. Đánh giá chất lượng đề tài tốt nghiệp (về các mặt như cơ sở lý luận, thuyết minh chương trình, giá trị thực tế).

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

2. Cho điểm của cán bộ phản biện

(*Điểm ghi bằng số và chữ*)

.....
.....

Ngày.....tháng.....năm 2013

Cán bộ chấm phản biện

(Ký, ghi rõ họ tên)

LỜI CẢM ƠN

Trước hết em xin chân thành cám ơn các thầy giáo, cô giáo Khoa Công nghệ thông tin Trường Đại học Dân lập Hải Phòng, những người đã dạy dỗ, trang bị cho chúng em những kiến thức cơ bản, cần thiết trong những năm học vừa qua để em có đủ điều kiện hoàn thành đề tài tốt nghiệp của mình.

Đặc biệt em xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy giáo Ths. Bùi Huy Hùng, thầy đã hướng dẫn, chỉ bảo tận tình trong suốt thời gian làm đề tài tốt nghiệp.

Em xin cảm ơn hai thầy Đoàn Quang Hưng và thầy Trương Hoàng Dũng bên trung tâm thư viện ICT đã hỗ trợ em rất nhiều trong quá trình làm đồ án.

Con xin gửi đến cha mẹ lời ghi ơn sâu sắc, những người đã sinh ra và dạy bảo con trưởng thành đến ngày hôm nay. Cảm ơn người tôi yêu đã động viên cho tôi những lúc tôi mệt mỏi. Em là động lực để tôi cố gắng.

Mặc dù đã hết sức cố gắng để hoàn thiện báo cáo tốt nghiệp song do khả năng còn hạn chế nên bài báo cáo vẫn còn nhiều thiếu sót. Vì vậy em rất mong nhận được những đóng góp chân tình của các thầy cô và bạn bè.

Một lần nữa em xin chân thành cảm ơn!

Hải Phòng, Ngày 04 tháng 11 năm 2013.

Sinh viên

Đào Văn Phong

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

DANH SÁCH CHỮ VIẾT TẮT

SSO	Single Sign On
CAS	Central Authentication Service
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SSL	Secure Sockets Layer
ST	Service Ticket
PT	Proxy Ticket
LT	Login Ticket
PGT	Proxy-granting ticket
PGTIOU	Proxy-granting ticket IOU
TGTIOU	Ticket -granting ticket IOU
TGT	Ticket-granting ticket
TGC	Ticket-granting cookie
CSDL	Cơ sở dữ liệu

LỜI NÓI ĐẦU

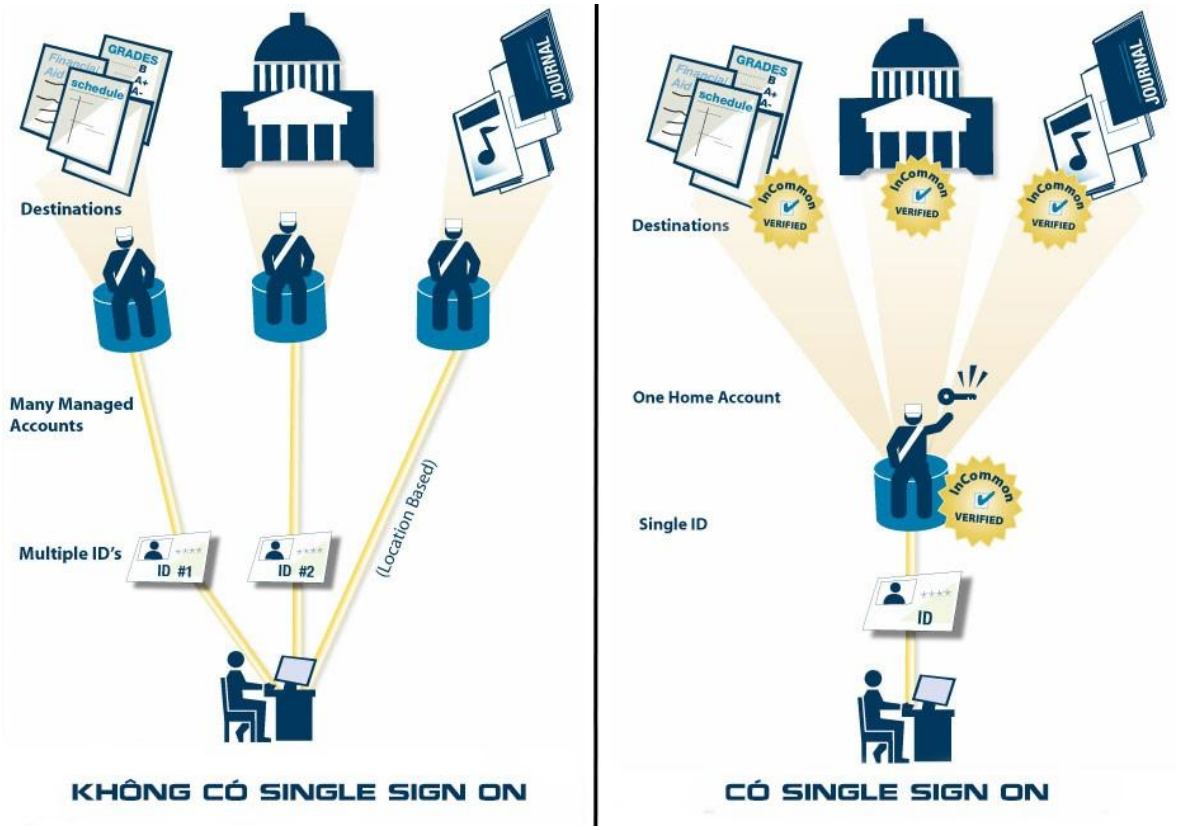
Khuynh hướng các dịch vụ cùng nhau chia sẽ dữ liệu người dùng đang là hướng phát triển chung của công nghệ thông tin,một người dùng phải quản lý rất nhiều tài khoản, mật khẩu cho các dịch vụ họ tham gia. Điều này sẽ xảy ra nhiều rủi ro do người dùng phải ghi nhớ các tài khoản khác nhau. Và hơn nữa, các ứng dụng và dịch vụ công nghệ thông tin ngày càng nhiều và đa dạng. Do vậy nhu cầu đăng nhập một lần cho các ứng dụng và dịch vụ này là không thể thiếu.Đăng nhập một lần (Single Sign On) đã được nhiều tổ chức, công ty trên thế giới nghiên cứu và phát triển, tuy nhiên tại Việt Nam đây vẫn là lĩnh vực còn khá mới. Trước vấn đề đó, em mong muốn được tìm hiểu và thực nghiệm hệ thống đăng nhập một lần. Với những gì đã nghiên cứu được, em hy vọng sẽ được đóng góp một phần nhỏ vào công tác phát triển khoa học. Mục đích: Tìm hiểu cơ chế đăng nhập 1 lần và nghiên cứu kỹ thuật Single Sign On để áp dụng đăng nhập một lần dựa trên thư viện phpCAS.

Xin chân thành cảm ơn !

CHƯƠNG IGIỚI THIỆU VỀ CƠ CHẾ ĐĂNG NHẬP 1 LẦN (SINGLE SIGN ON).

1.1. Tổng quan về SSO.[1]

SSO là một cơ chế xác thực yêu cầu người dùng đăng nhập vào chỉ một lần với một tài khoản và mật khẩu để truy cập vào nhiều ứng dụng trong 1 phiên làm việc (session).



Hình 1.1: Single sign on là gì?

1.2. Lợi ích mà SSO mang lại.

Trước khi có đăng nhập một lần (SSO), một người sử dụng đã phải nhập các tài khoản và mật khẩu cho từng ứng dụng mỗi khi họ đăng nhập vào các ứng dụng khác nhau hoặc các hệ thống trong cùng một phiên (session). Điều này rõ ràng có thể tốn nhiều thời gian, đặc biệt là trong môi trường doanh nghiệp, nơi mà thời gian là tiền bạc nhưng thời gian là lãng phí bởi vì nhân viên phải đăng nhập mỗi khi họ truy cập vào một hệ thống mới từ máy tính của họ.

SSO thường được thực hiện thông qua một mô-đun xác thực phần mềm riêng biệt hoạt động như một cửa ngõ vào tất cả các ứng dụng yêu cầu đăng nhập. Các

mô-đun xác thực người sử dụng và sau quản lý truy cập vào các ứng dụng khác. Nó hoạt động như một kho dữ liệu chung cho tất cả các thông tin đăng nhập được yêu cầu.

Ví dụ:

Một ví dụ về một module SSO là hệ thống của Google khi mà người dùng chỉ cần đăng nhập 1 lần thì họ có thể sử dụng các dịch vụ của Google hay Yahoo mà không đòi hỏi đăng nhập 1 lần nữa như Gmail, Google Plus, Youtube…..

Trong khi SSO là rất tiện lợi, một số nhận thấy nó như là một vấn đề an ninh của riêng mình. Nếu hệ thống SSO bị tổn thương, một kẻ tấn công có quyền truy cập không giới hạn cho tất cả các ứng dụng chứng thực của các module SSO.SSO thường là một dự án lớn cần lập kế hoạch cẩn thận trước khi thực hiện.

1.3. Một số vấn đề thường gặp khi triển khai SSO.

- Có phải nếu sử dụng SSO sẽ cải thiện vấn đề bảo mật?

Xin trả lời rằng:

Đăng nhập một lần (SSO) là một con dao hai lưỡi. SSO tự nó không thực sự cải thiện bảo mật và trên thực tế, nếu không triển khai đúng cách có thể làm giảm bảo mật. SSO được sử dụng nhiều hơn cho người sử dụng thuận tiện.

Như hệ thống của công ty nhân, với mỗi một yêu cầu mật khẩu riêng của mình, SSO giúp giảm bớt gánh nặng phải dành thời gian đăng nhập vào từng hệ thống riêng. Nhưng đồng thời, nếu SSO bị tổn thương, nó mang lại cho tin tặc khả năng truy cập vào toàn bộ hệ thống sử dụng SSO. Mặt khác, SSO có những lợi ích nhiều hơn những rủi ro nó mang lại.

Vì vậy, mặc dù SSO không phải là thuốc chữa bách bệnh bảo mật trong và của chính nó, nhưng nó có thể đóng góp tích cực vào một chương trình bảo mật thông tin doanh nghiệp. Dưới đây là đề cập cụ thể.

Hệ thống SSO thường dựa trên các ứng dụng phức tạp hệ thống quản lý như IBM Tivoli (http://en.wikipedia.org/wiki/IBM_Tivoli_Directory_Server), hoặc dựa trên phần cứng thiết bị từ hãng Imprivata Inc(1 hãng cung cấp giải pháp SSO nổi tiếng http://www.imprivata.com). Kết quả là, hệ thống SSO có thể tập trung xác thực trên các máy chủ đặc biệt. Họ làm điều này bằng cách sử dụng các máy chủ chuyên dụng để giữ các module SSO. Các máy chủ hoạt động như SSO người gác

cổng, đảm bảo tất cả các xác thực đi đầu tiên thông qua máy chủ SSO, sau đó đi dọc theo các chứng chỉ đã được lưu trữ để xác thực các ứng dụng cụ thể đã đăng ký với hệ thống SSO. Hệ thống đòi hỏi phải lập kế hoạch cụ thể và chi tiết để kiểm toán để ngăn chặn truy cập độc hại hơn so với các hệ thống SSO làm(Có nghĩa là nếu được đầu tư về phần cứng thích hợp thì nó sẽ tăng bảo mật).

Ngoài ra, hệ thống SSO thường có lưu trữ an toàn hơn các thông tin xác thực và các khóa mã hóa, làm cho chúng là một thách thức đối với tin tặc. Hệ thống SSO nằm sâu trong kiến trúc IT của công ty. Nó thường giấu một cách an toàn sau nhiều bức tường lửa. Điều này sẽ giúp SSO an toàn hơn.

- *Các yếu tố cần xem xét trước khi triển khai SSO là gì?*

Đăng nhập một lần (SSO) có thể là một giải pháp cho tình hình của bạn, nhưng tất cả phụ thuộc vào hoàn cảnh của đơn vị triển khai, đặc biệt là nhu cầu bảo mật và ngân sách. SSO có ưu điểm và những rủi ro của nó.

Hai ưu điểm chính là:

- Thuận tiện: Người sử dụng chỉ cần đăng nhập 1 lần để sử dụng nhiều ứng dụng.
- Bảo mật: Bởi vì chỉ có một đăng nhập một lần, SSO có thể loại bỏ những rủi ro vốn có trong việc ghi nhớ nhiều username/password.

Hai rủi ro chính là:

- Bảo mật: Nếu một kẻ xâm nhập làm tổn hại tài khoản của người dùng hoặc mật khẩu, kẻ xâm nhập có thể có rộng rãi và dễ dàng truy cập vào rất nhiều ứng dụng.
- Chi phí: triển khai SSO có thể tốn kém, cả về chi phí để mua và nguồn nhân lực để triển khai.

Hai yếu tố SSO là tốt nhất, nơi truy cập được cấp dựa trên sự kết hợp đối với những gì người sử dụng biết (mật khẩu hoặc mã PIN)

1.4. Các giải pháp SSO hiện nay.[2]

Dưới đây là các giải pháp SSO hiện có sẵn.

Bảng 1.1: Danh sách các giải pháp SSO.

Tên sản phẩm	Nhà phát triển	Loại hình	Nền tảng	Mô tả
Accounts & SSO	Nokia, Intel, …	Miễn phí		Client-side implementation with plugins for various services/protocols
Novell Access Manager	NetIQ	Thương mại		webSSO to browser based applications with rules, policies and methods to be complied to access-event.
Active Directory Federation Services	Microsoft	Commercial		Claims-based system and application federation
Athens access and identity management	Eduserv UK	Thương mại	Yes	
CAS / Central Authentication Service	Jasig	Mã nguồn mở		Protocol and SSO server/client implementation
CoSign single	University of	Tổ chức riêng		SSO for

Tên sản phẩm	Nhà phát triển	Loại hình	Nền tảng	Mô tả
sign on	Michigan			Michigan University
Distributed Access Control System (DACS)	Distributed Systems Software	Miễn phí		
Enterprise Sign On Engine	Queensland University of Technology	Miễn phí		
Facebook connect	Facebook	Facebook specific SSO		Facebook SSO to third parties enabled by Facebook
Forefront Identity Manager	Microsoft	Thương mại	Yes	State-based identity life-cycle management
FreeIPA	Red Hat	Miễn phí		
HP IceWall SSO	Hewlett-Packard Development Company, L.P.	Thương mại		Web and Federated Single Sign-On Solution

Tên sản phẩm	Nhà phát triển	Loại hình	Nền tảng	Mô tả
LTPA	IBM	Thương mại		
IBM Tivoli Identity Manager	IBM	Thương mại	Yes	Identity life-cycle management product
Janrain Federate SSO	Janrain	Thương mại	Yes	Social and conventional user SSO
JBoss SSO	Red Hat	Miễn phí		Federated Single Sign-on
JOSSO	JOSSO	Miễn phí		Open Source Single Sign-On Server
Kerberos	M.I.T.	Protocol		Computer network authentication protocol
Microsoft account	Microsoft	Miễn phí và thương mại (Microsoft bây giờ thu hút các trang web mới để sử dụng hệ thống)		Microsoft single sign-on web service
myOneLogin	VMware	Thương mại		Cloud single

Tên sản phẩm	Nhà phát triển	Loại hình	Nền tảng	Mô tả
				sign-on
Numina Application Framework	Numina Solutions	Thương mại	Yes	Single sign-on system for Windows (OpenID RP & OP, SAML IdP, and proprietary)
OneLogin	OneLogin Inc.	Thương mại và Miễn Phí	Yes	Cloud-based identity and access management with single sign-on (SSO) and active directory integration
Okta	Okta,Inc.	Thương mại		On-demand identity and access management service in the cloud
OpenAM	ForgeRock	Miễn phí	Yes, used in conjunction withOpenDJ and OpenIDM	Access management, entitlements and federation server platform

Tên sản phẩm	Nhà phát triển	Loại hình	Nền tảng	Mô tả
Persona	Mozilla	Miễn phí		
Pubcookie	University of Washington	Protocol		
SecureLogin	NetIQ	Thương mại		Enterprize Single-Sign-On
SAML	OASIS	Protocol		XML-based open standard protocol
Shibboleth	Shibboleth	Miễn phí		SAML-based open source access control
Ubuntu Single Sign On	Canonical Ltd.	Thương mại và miễn phí		OpenID-based SSO for Launchpad and Ubuntu services
ZXID	ZXID	Miễn phí	Yes	Reference Implementation of TAS3 security

CHƯƠNG IIPHẦN MỀM NGUỒN MỞ CENTRAL AUTHENTICATION SERVICE.

2.1. Giới thiệu về phần mềm nguồn mở (Opensource).[3]

Phần mềm nguồn mở là gì?

Open source software là những phần mềm được viết và cung cấp một cách tự do. Người dùng phần mềm mã nguồn mở không những được dùng phần mềm mà còn được tải mã nguồn của phần mềm, để tùy ý sửa đổi, cải tiến và mở rộng cho nhu cầu công việc của mình.

Một phần mềm áp dụng loại giấy phép mà cho phép bất cứ ai sử dụng dưới mọi hình thức, có thể là truy cập, chỉnh sửa, sao chép,…và phân phối các phiên bản khác nhau của mã nguồn phần mềm, được gọi là open-source software. Nhìn chung, thuật ngữ "Open source" được dùng để lôi cuốn các nhà kinh doanh, một điều thuận lợi chính là sự miễn phí và cho phép người dùng có quyền "sở hữu hệ thống".

Tiện ích mà opensource mang lại chính là quyền tự do sử dụng chương trình cho mọi mục đích, quyền tự do để nghiên cứu cấu trúc của chương trình, chỉnh sửa phù hợp với nhu cầu, truy cập vào mã nguồn, quyền tự do phân phối lại các phiên bản cho nhiều người, quyền tự do cải tiến chương trình và phát hành những bản cải tiến vì mục đích công cộng.

2.2. Dịch vụ chứng thực trung tâm (Central Authentication Service).[4]

2.2.1 Tổng quan về CAS.

CAS là 1 giao thức đăng nhập một lần (SSO) cho web được phát triển bởi đại học Yale. Mục đích của nó là cho phép người dùng truy cập nhiều ứng dụng trong khi chỉ cần cung cấp thông tin của họ (ví dụ như username và password) chỉ một lần. Nó cũng cho phép các ứng dụng web xác thực người sử dụng mà không cần tiếp cận với các thông tin bảo mật người dùng, chẳng hạn như mật khẩu.

CAS hỗ trợ nhiều thư viện phía client được viết bởi nhiều ngôn ngữ như PHP,.NET, JAVA,RUBY….

Giao thức CAS bao gồm ít nhất ba bên: một trình duyệt web của client, các ứng dụng web yêu cầu chứng thực, và các máy chủ CAS. Nó cũng có thể liên quan đến một dịch vụ back-end, chẳng hạn như một máy chủ cơ sở dữ liệu, nó không có giao diện HTTP riêng của mình nhưng giao tiếp với một ứng dụng web.

Khi client truy cập một ứng dụng mong muốn để xác thực với nó, ứng dụng chuyển hướng nó đến CAS. CAS xác nhận tính xác thực của client, thường là bằng cách kiểm tra tên người dùng và mật khẩu đối với một cơ sở dữ liệu (chẳng hạn như MYSQL/PGSQL). Nếu xác thực thành công, CAS trả client về ứng dụng trước đó thông qua 1 service ticket(ST). Ứng dụng này sau đó xác nhận ticket bằng cách liên

hệ CAS trên một kết nối an toàn và cung cấp dịch vụ nhận dạng riêng của mình và ticket.Nếu CAS sau đó cung cấp cho các ứng dụng đáng tin cậy thông tin về việc một người dùng cụ thể đã thành công.Ngoài ra, người dùng cũng có thể xác thực thông tin trực tiếp tại trang đăng nhập của CAS, nếu vượt qua sự xác thực của CAS thì người dùng có thể dùng bất cứ dịch vụ nào đã được đăng ký SSO. CAS cho phép chứng thực đa cấp thông qua địa chỉ proxy. Một hợp tác dịch vụ back-end, như một cơ sở dữ liệu hoặc máy chủ mail, có thể tham gia trong CAS, xác nhận tính xác thực của người dùng thông qua các thông tin nhận được từ các ứng dụng web. Do đó, một webmail và một máy chủ email trực tuyến đều có thể thực hiện CAS.

CAS còn cung cấp tính năng "Remember Me". Những người phát triển có thể cấu hình tính năng này trong nhiều file cấu hình khác nhau và khi người dùng chọn "Remember Me" trên khung đăng nhập thì thông tin đăng nhập sẽ được ghi nhớ với thời gian cấu hình mặc định là 3 tháng và khi người dùng mở trình duyệt thì CAS sẽ tự động chuyển hướng tới service URL mà người dùng muốn truy cập mà không hiển thị form đăng nhập.

2.2.2 Lịch sử hình thành.[5]

CAS được hình thành và phát triển bởi Shawn Bayern của Yale trường đại học công nghệ và kế hoạch. Sau đó nó được duy trì bởi Drew Mazurek ở Đại học Yale. CAS 1.0 thực hiện đơn-đăng nhập. CAS 2.0 giới thiệu xác thực ủy quyền multi-tier. Một số các bản phát hành CAS khác đã được phát triển với tính năng mới.

Trong tháng 12 năm 2004, CAS đã trở thành một dự án của Java Kiến trúc Special Interest Group, chịu trách nhiệm duy trì và phát triển của nó năm 2008. Trước đây gọi là "Đại học Yale CAS", CAS là bây giờ còn được gọi là "Jasig CAS".

Tháng 12 năm 2006, Andrew W. Mellon Quỹ giải Yale của nó đầu tiên hàng năm Mellon cho nghiên cứu khoa học công nghệ, trong số tiền $50.000, cho sự phát triển của Yale của CAS. Vào thời điểm đó giải CAS sử dụng tại "hàng trăm của trường đại học (trong số các đơn vị thụ hưởng)".

Hiện nay rất nhiều trường đại học nổi tiếng trên thế giới tin dùng vào hệ thống đăng nhập 1 lần SSO do đại học Yale cung cấp. Chúng ta có thể xem tại địa chỉ: http://www.jasig.org/cas/deployments

2.2.3 Các phiên bản của CAS.

- CAS 1.0
  - Được tạo bởi Yale University, khởi đầu từ năm 1999.
  - Là 1 SSO dễ sử dụng
- CAS 2.0
  - Cũng được tạo bởi Yale University
  - Giới thiệu thêm tính năng mới là Proxy Authentication.
- JA-SIG CAS 3.0
  - Trở thành JA-SIG project từ năm 2004
  - Mục đích là cho nó mềm dẻo hơn và tích hợp được với nhiều hệ thống hơn.

2.2.4 CAS Protocol.

CAS là một giao thức HTTP/HTTPS dựa trên giao thức mà đòi hỏi mỗi thành phần của nó có thể truy cập thông qua các URI cụ thể.

2.2.4.1./login.

- Vai trò.
  - Yêu cầu chứng thực.
  - Chấp nhận chứng thực.
- Tham số

Theo như HTTP yêu cầu các tham số sau đây có thể được thông qua với /login trong khi nó đang hành động như một người yêu cầu chứng thực. Các tham số đều là những trường hợp nhạy cảm, và tất cả đều phải được xử lý bởi /login.

- **Service[Tùy chọn]** - nhận dạng của các ứng dụng mà client đang cố gắng truy cập. Trong hầu hết các trường hợp, nó là URL của ứng dụng. Lưu ý rằng như một tham số yêu cầu HTTP, giá trị URL này phải là URL-encoded. Nếu một service không được chỉ định và 1 session SSO chưa tồn tại thì CAS nên yêu cầu chứng thực từ người sử dụng để bắt đầu một session SSO. Nếu một service không được chỉ định và session SSO đã

tồn tại, CAS sẽ hiển thị một tin nhắn thông báo cho client rằng nó đã được đăng nhập.

- **Renew [Tùy chọn]** - nếu tham số này được thiết lập, SSO sẽ được bỏ qua. Trong trường hợp này, CAS sẽ yêu cầu client trình thông tin đăng nhập hiện tại mà không quan tâm đến sự tồn tại của session SSO với CAS. Tham số này là không tồn tại song song với tham số "gateway". Service chuyển hướng đến các URI và form login /login để đăng URI /login. Không nên đặt cả "renew" và "gateway" trong 1 URL. Hành vi không xác định nếu cả hai được thiết lập. Khuyến nghị triển khai CAS bỏ qua các tham số "gateway" nếu tham số "renew" được thiết lập. Khuyến nghị khi các tham số renew được thiết lập thì giá trị của nó là "true".
- **Gateway [Tùy chọn]** – Nếu tham số này được thiết lập thì CAS sẽ không yêu cầu client chứng thực thông tin nữa. Nếu client đã đăng nhập từ trước đây với SSO session với CAS hay nếu SSO session được thiết lập thông qua không tương tới nhau(tức là xác thực tin tưởng) thì CAS có thể chuyển hướng client tới URL được chỉ định bởi tham số "service" và thêm vào 1 ST hợp lệ(CAS có thể thông báo cho client rằng đã có xác thực xảy ra trước đây.). Nếu client không có SSO session với CAS và xác thực không tương tác không thể thiết lập thì CAS phải chuyển hướng client đến URL được chỉ định bởi tham số "service" không có tham số "ticket" nào được thêm vào URL. Nếu tham số "service" không được chỉ định và tham số "gateway" được đặt thì các hành động của CAS là không khác định. Tham số này không cùng song hành trên 1 URL với tham số "renew". Hành động sẽ không xác định nếu cả 2 được set. Các tham số "gateway" nên có giá trị mặc định là "yes".

- Phản hồi
  - **Đăng nhập thành công**: chuyển hướng client đến URL được chỉ định bởi tham số "Service" một cách mà sẽ không làm cho thông tin đăng nhập của client được chuyển tiếp đến service. Chuyển hướng này phải dẫn đến client đưa ra một GET yêu cầu cho các service. Yêu cầu phải bao gồm một service ticket hợp lệ, thông qua như là tham số yêu cầu HTTP, "ticket". Xem phụ lục B để biết thêm thông tin. Nếu không xác định

"Service", CAS phải hiển thị một thư thông báo cho client rằng nó đã thành công bắt đầu single sign-on session.

- Đăng nhập thất bại: Trả lại /login như là một requestor ủy nhiệm. Đó là khuyến cáo trong trường hợp này máy chủ CAS hiển thị một thông báo lỗi được hiển thị cho người dùng mô tả lý do tại sao đăng nhập không thành công (ví dụ như sai mật khẩu, tài khoản bị khóa, vv), và nếu cần thiết, cung cấp một cơ hội cho người dùng thử đăng nhập lại.

Ví dụ về tham số trong /login

Đăng nhập đơn giản.

https://server/cas/login?service=http://www.service.com

Không nhắc tên người dùng và mật khẩu.

https://server/cas/login?service=http://www.service.com&gateway=true

Luôn nhắc tên người dùng và mật khẩu.

https://server/cas/login?service=http://www.service.com&renew=true

2.2.4.2. /logout

Phá hủy phiên làm việc của cơ chế SSO trên máy client. TGC sẽ bị phá hủy và yêu cầu tiếp theo vào /login sẽ không có được ST cho đến khi user thiết lập một SSO session mới.

- Tham số

Tham số "url" có thể được chỉ định đến /logout và nếu được chỉ định "url" sẽ được hiển thị trong trang logout cùng với thông báo đăng xuất.

2.2.4.3. /validate. CAS[1.0]

Kiểm tra tính hợp lệ của ST. CAS phải phản hồi 1 ticket validation thất bại khi có 1 proxy ticket được thông qua URI /validate.

- Tham số

Những tham số sau có thể chỉ định đến URI /validate.

- **Service [bắt buộc].**
- **Ticket [bắt buộc]** - service ticket được sinh ra bởi /login.

- **Renew [Tùy chọn]** - Nếu tham số này được thiết lập, ticket validation sẽ chỉ thành công nếu ST đã được phát hành từ bài trình bày của chứng chỉ chính của người dùng. Nó sẽ không thành công nếu ticket đã được phát hành từ một SSO session.

- Phản hồi

/validate sẽ trả lại 1 trong hai phản hồi sau.

Ticket validation thành công:

yes<LF>

username<LF>

Ticket validation thất bại:

no<LF>

<LF>

Ví dụ của /validate

Lỗ lực xác thực đơn giản:

https://server/cas/validate?service=http://www.service.com&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7

Chắc chắn rằng ST được ban hành các trình bày các thông tin chính.

https://server/cas/validate?service=http://www.service.com&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&renew=true

2.2.4.4. /serviceValidate [CAS 2.0]

/serviceValidate sẽ trả về phản hồi là một XML-fragment. Khi thành công phản hồi chứa username và proxy-granting tickets. Khi thất bại, phản hồi chứa 1 mã lỗi và 1 thông điệp tương ứng. Dưới đây là 1 số mã lỗi trả về nếu thất bại.

- **INVALID_REQUEST** – không tìm thấy tham số cần tìm tring request.
- **INVALID_TICKET** – Ticket cung cấp không hợp lệ hoặc ticket không đến từ login và "renew" được thiết lập trên validation.
- **INVALID_SERVICE** – Ticket được cung cấp hợp lệ nhưng dịch vụ được chỉ định không khớp với dịch vụ liên kết với ticket.

- **INTERNAL_ERROR** – Lỗi cục bộ trong khi kiểm tra tính hợp lệ của ticket.

- Phản hồi

/serviceValidate sẽ trả về 1 XML-formatted CAS được mô tả như trong XML schema. Dưới đây là ví dụ:

Xác thực Ticket thành công:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationSuccess>

<cas:user>username</cas:user>

<cas:proxyGrantingTicket>PGTIOU-84678-8a9d... </cas:proxyGrantingTicket>

</cas:authenticationSuccess>

</cas:serviceResponse>

2.2.4.5. /proxy callback.

Nếu một dịch vụ mong muốn ủy quyền chứng thực của client tới một service back-end, nó phải có được một proxy-granting ticket(PGT). Để có được PGT thì nó phải được xử lý thông qua một URL callback. URL này sẽ duy nhất và an toàn xác định dịch vụ back-end là proxying xác thực của client. Các dịch vụ back-end có thể sau đó quyết định có hay không để chấp nhận các chứng chỉ dựa trên các dịch vụ back-end xác đinhk URL callback.

Cơ chế làm việc của nó như sau:

Các dịch vụ yêu cầu 1 PGT cấp quy định trên ST hoặc PT xác nhận yêu cầu tham số HTTP "pgtUrl" tới /serviceValidate (or /proxyValidate). Đó là 1 callback URL của dịch vụ mà CAS sẽ kết nối để xác minh danh tích của dịch vụ. URL này phải có HTTPS và CAS phải xác minh cả 2 chứng chỉ SSL là hợp lệ và chính xác tên của dịch vụ. Nếu chứng chỉ không được xác nhận, không có PGT sẽ được cấp lại và đáp ứng dịch vị CAS không phải chứa 1 khối <proxyGrantingTicket>

Ticket validation thành công:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationSuccess>

<cas:user>username</cas:user>

<cas:proxyGrantingTicket>PGTIOU-84678-8a9d... </cas:proxyGrantingTicket>

</cas:authenticationSuccess>

</cas:serviceResponse>

Ticket validation thất bại:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationFailure code="INVALID_TICKET">

Ticket ST-1856339-aA5Yuvrxzpv8Tau1cYQ7 not recognized

</cas:authenticationFailure>

</cas:serviceResponse>

Tại thời điểm này, việc cấp phát 1 PGT phải dừng lại nhưng xác nhận ST vẫn tiếp tục, trả lại thành công hoặc thất bại như là thích hợp. Nếu chứng chỉ chứng nhận thành công, phát hành 1 PGT được xử lý như ở bước 2.

CAS sử dụng 1 HTTP GET request để vượt qua tham số HTTP "pgId" và "pgIou" tới pgtUrl.

Nếu HTTP GET trả lại 1 một mã trạng thái HTTP 200 (OK), CAS sẽ phải phản hồi tới /serviceValidate (or /proxyValidate) yêu cầu cho một phản hồi dịch vụ có chứa PGT IOU trong khối <cas:proxyGrantingTicket>. Nếu HTTP GET trả về bất kỳ mã trạng thái khác, ngoại trừ HTTP 3xx redirect, CAS phải phản hồi /serviceValidate (or /proxyValidate) yêu cầu cho 1 phản hồi dịch vụ mà không phải có một khối <cas:proxyGrantingTicket>. CAS có thể làm theo bấy kỳ HTTP redirects do pgtUrl. Tuy nhiên, xác định các callback url cung cấp trên xác nhận trong khối <proxy> phải cùng một URL mà ban đầ đã được thông qua vào /serviceValidate (or /proxyValidate) là than số "pgtUrl".

Dịch vụ sau khi nhận 1 PGTIOU do CAS phản hồi và cả 1 PGT, 1 PGT IOU từ proxy callback, sẽ sử dụng PGTIOU tương quan với PGT với các phản ứng xác nhận. Dịch vụ này sau đó sẽ sử dụng PGT cho việc có lại các PT như mô tả trong phần "Proxy Tickets".

Một PGT là 1 chuỗi ngẫu nhiên sử dụng bởi 1 dịch vụ để có được PT cho việc tiếp cận dịch vụ back-end thay mặt cho 1 client. PGT có thể được sử dụng bởi các dịch vụ để có được nhiều PT. PGTs không phải là 1 ticket thời gian sử dụng.

PGT phải hết hạn khi client có xác thực đang được các bản ghi ra uỷ nhiệm của CAS.

PGT nên bắt đầu với các ký tự "PGT-".

URL ví dụ của /serviceValidate

Lỗ lực xác thực đơn giản:

https://server/cas/serviceValidate?service=http://www.service.com&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7

Đảm bảo ST được đưa ra bởi các trình bày thông tin đăng nhập chính:

https://server/cas/serviceValidate?service=http://www.service.com&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&renew=true

Vượt qua một callbackURL cho proxying:

https://server/cas/serviceValidate?service=http://www.service.com&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&pgtUrl=https://my-server/myProxyCallback

2.2.4.6. /proxyValidate [CAS 2.0].

Làm việc giống như serviceValidate ngoại trừ nó làm cho proxy ticket có hiệu lực. Tham số và mã lỗi cũng tương tự. Khi thành công, phản hồi chứa PGT và danh sách các proxy cái mà việc xác thực được thực thi. Những proxy được viếng thăm gần nhất sẽ được liệt kê đầu tiên và ngược lại.

Ví dụ

Ticket validation thành công:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationSuccess>

<cas:user>username</cas:user>

<cas:proxyGrantingTicket>PGTIOU-84678-8a9d...</cas:proxyGrantingTicket>

<cas:proxies>

<cas:proxy>https://proxy2/pgtUrl</cas:proxy>

<cas:proxy>https://proxy1/pgtUrl</cas:proxy>

</cas:proxies>

</cas:authenticationSuccess>

</cas:serviceResponse>

Ticket validation thất bại:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationFailure code="INVALID_TICKET">

ticket PT-1856376-1HMgO86Z2ZKeByc5XdYD not recognized

</cas:authenticationFailure>

</cas:serviceResponse>

URL ví dụ của /proxyValidate

Tương tự như /serviceValidate

2.2.4.7. /proxy [CAS 2.0]

Cung cấp PT để các dịch vụ đã có PGT và sẽ được xác thực proxy với các dịch vụ back-end.

- Tham số

2 tham số bắt buộc phải có là:

- **pgt [Bắt buộc]** - proxy-granting ticket đạt được bởi service trải qua service ticket hoặc proxy ticket validation.
- **targetService [Bắt buộc]** - định danh dịch vụ của dịch vụ back-end. Lưu ý rằng, không phải tất cả các service back-end là dịch vụ web để nhận dạng dịch vụ này sẽ không phải luôn luôn là một URL. Tuy nhiên, định danh dịch vụ quy định ở đây phải phù hợp với tham số "service" quy định cho / proxyValidate dựa trên xác nhận hợp lệ của proxy ticket.

- Phản hồi

/proxy sẽ trả lại 1 XML-formatted CAS được mô tả như trong XML chema trong phần Phụ lục A. Bên dưới là 1 ví dụ của phản hồi:

Yêu cầu thành công:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:proxySuccess>

<cas:proxyTicket>PT-1856392-b98xZrQN4p90ASrw96c8</cas:proxyTicket>

</cas:proxySuccess>

</cas:serviceResponse>

Yêu cầu thất bại:

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:proxyFailure code="INVALID_REQUEST">

'pgt' and 'targetService' parameters are both required

</cas:proxyFailure>

</cas:serviceResponse>

- Mã lỗi

Các giá trị sau đây có thể được sử dụng như là thuộc tính "code" của các phản ứng thất bại. Sau đây là các thiết lập tối thiểu của mã lỗi rằng tất cả các máy chủ CAS phải thực hiện.

- **INVALID_REQUEST** - không phải tất cả các thông số yêu cầu cần thiết đã có mặt
- **BAD_PGT** - các PGT cung cấp không hợp lệ
- **INTERNAL_ERROR** - một lỗi nội bộ xảy ra trong quá trình ticket validation.

Đối với tất cả các mã lỗi, khuyến nghị rằng, CAS cung cấp tin chi tiết hơn trong phần body của khối <cas:authenticationFailure> của phản hồi XML.

URL example of /proxy

Yêu cầu proxy đơn giản:

https://server/cas/proxy?targetService=http://www.service.com&pgt=PGT-490649-W81Y9Sa2vTM7hda7xNTkezTbVge4CUsybAr

2.2.5. Tổng kết.

Bảng 2.1: Tổng hợp các URI.

URI	**Mô tả**
/login	Nó phản ứng với thông tin bằng cách hành động như một người chấp nhận chứng chỉ và nếu không hoạt động như

	một người yêu cầu chứng chỉ. Nếu client đã thiết lập phiên làm việc SSO (single sign-on) với CAS thì web browser sẽ gửi đến CAS 1 Cookies an toàn nó bao gồm 1 chuỗi xác định 1 TGT(Ticket granting ticket). Cookie này được gọi là TGC(ticket-granting cookie). Nếu khóa TGC hợp lệ cho TGT, CAS có quyền cấp một ST(service ticket) cung cấp tất cả các điều kiện khác nhau trong đặc điểm kỹ thuật nó đã gặp.
/logout	Phá hủy phiên làm việc của cơ chế SSO trên máy client. TGC sẽ bị phá hủy. và yêu cầu tiếp theo vào /login nhập sẽ không có được ST cho đến khi user thiết lập một SSO
/validate	Kiểm tra tính hợp lệ của service ticket. /validate là 1 phần của giao thức CAS 1.0 và do đó nó không xử lý xác thực proxy.
/serviceValidate	Kiểm tra tính hợp lệ của một ST và trả về một đoạn XML(XML-fragment)
/proxyValidate	Thực hiện các nhiệm vụ tương tự như /serviceValidate và bổ sung xác nhận PT(proxy ticket).
/proxy	Cung cấp PT để các dịch vụ đã có PGT và sẽ được xác thực proxy với các dịch vụ back-end.
/samlValidate	

/services/add.html	Một chức năng quản trị. Bổ sung thêm dịch vụ vào danh sách dịch vụ đăng ký.
/services/edit.html	Một chức năng quản trị. Sửa dịch vụ đã đăng ký.
/services/manage.html	Cung cấp một giao diện để quản lý các dịch vụ đăng ký (thêm / sửa / xóa các dịch vụ)
/services/logout.html	Thoát khỏi trang quản trị
/services/loggedOut.html	Thoát khỏi trang quản trị từ trang dịch vụ
/services/deleteRegisteredService.html	Xóa các tham số dịch vụ dựa vào "ID"
/openid/*	Yêu cầu map cho usernames đến một trang hiển thị Login URL cho nhà cung cấp định OpenID

2.2.6. CAS Entities.

Ticket-granting ticket (TGT):

TGT sẽ được tạo ra khi /login url vượt qua được được dịch vụ CAS và các thông tin cung cấp sẽ được chứng thực thành công. 1 TGT là 1 truy cập chính vào lớp dịch vụ của CAS. Nếu không có TGT thì user của CAS sẽ không làm được bất cứ điều gì. TGT là 1 chuỗi ngẫu nhiên với tiền tố là "TGT-". TGT sẽ được thêm vào 1 HTTP Cookies trên sự thành lập của của cơ chế SSO và bất cứ khi nào user truy cấp vào các ứng dụng khác nhau thì cookies này sẽ gọi cơ chế auto-login cho user đó.

Ticket Granting Cookie (TGC):

TGC là 1 cookies của HTTP cookies đặt bởi CAS trên sự khởi tạo phiên làm việc của cơ chế SSO. Cái Cookies này duy trì trạng thái đăng nhập cho client và khi client điều hướng tới 1 ứng dụng khác thì cookies sẽ kiểm tra auto-login cho

user này. TGC sẽ bị phá hủy khi client đóng trình duyệt và nó cũng bị phá hủy khi client click vào /logout. Giá trị của TGC nên bắt đầu với "TGC-".

Service Ticket (ST):

ST sẽ được tạo khi CAS url bao gồm tham số dịch vụ và các thông tin đã thông qua được xác thực thành công.

Ví dụ: https://server/cas/login?service=http://www.service.com

Các dịch vụ mà bạn thông qua url phải là một dịch vụ đăng ký thông qua các dịch vụ quản lý của CAS nếu không một dịch vụ không được xác thực sẽ bị ném ra.ST là 1 chuỗi ngẫu nhiên được sử dụng bởi client như 1 thông tin được truy cấp vào 1 dịch vụ. ST phải bắt đầu với "ST-".

Ví dụ: ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7

Khi tạo ST, service identifier (thường là service url) không phải là một phần của ST.

Proxy Ticket (PT):

Mô tả tóm tắt về proxy: 1 proxy hoạt động như 1 máy chủ, nhưng khi có yêu cầu từ client, hoạt động chính của nó như là 1 client đến các máy chủ thực. (Nó đại diện cho client giao tiếp với máy chủ.). 1 HTTP proxy không chuyển tiếp các yêu cầu gửi thông qua nó. Thay vào đó, việc đầu tiên nó kiểm tra nếu đã có các trang web yêu cầu trong bộ nhớ cache. Nếu như vậy, sau đó nó sẽ trang về trang đó mà không gửi thêm yêu cầu khác đến máy chủ đích. Bởi vì các proxy hoàn toàn chấn dứt các kênh giao tiếp, chúng được coi là 1 công nghệ firewall an toàn hơn các bộ lọc gói tin, vì chúng là tăng đáng kể sự cô lập giữa các mạng.

Trong CAS, proxy là 1 dịch vụ muốn truy cập vào các dịch vụ khác thay mặt cho 1 user đặc biệt. PT được tạo ra từ CAS trên 1 trình bày của dịch vụ hợp lệ TGT và 1 dịch vụ định danh (các giá trị của tham số "service" của /proxy url) cho dịch vụ back-end mà nó được kết nối.

PT là một chuỗi ngẫu nhiên mà một dịch vụ sử dụng như một chứng chỉ để có được quyền truy cập vào một dịch vụ back-end thay mặt cho một client.

PT chỉ có giá trị định danh dịch vụ quy định để cho/proxy url khi chúng được tạo ra. PT nên bắt đầu với các ký tự "PT-".

Proxy-granting Ticket (PGT):

PGT thu được từ CAS khi xác nhận của 1 ST hoặc 1 PT. Nếu một dịch vụ mong muốn ủy quyền chứng thực cho client tới 1 dịch vụ back-end, nó phải có được 1 PGT. Sự có được TGT này được xử lý thông qua một proxy callback URL. Cái URL này độc đáo và an toàn sẽ xác định các dịch vụ trong back-end sau là proxy chứng thực của client. Dịch vụ back-end sau đó có thể quyết định có hay không chấp nhận các thông tin dựa vào việc xác định gọi lại các URL.

Proxy-Granting Ticket IOU (PGTIOU):

1 PGT IOU là 1 chuỗi ngẫu nhiên với tiền tố là "PGTIOU-"cái mà được đặt trong phản ứng được cung cấp bởi /serviceValidate và /proxyValidate sử dụng để liên kế một ST hoặc xắc nhận PT với 1 PGT cụ thể. Mô tả đầy đủ của quá trình này khả dụng tại phiên làm việc của PGT.

Quả trình được mô tả đơn giản và đầy đủ được đưa ra trong phiên làm việc PGT. 1 yêu cầu được gửi cho PGT thông qua /serviceValidate hoặc /proxyValidate URI. CAS server không thể cung cấp cho PGT phản ứng trở trong phản ứng của nó, bởi vì nó không tin chắc nhận dạng người yêu cầu. Nếu nhận dạng người yêu cầu là nhận dạng chính xác thì CAS nói "IOU (I Owe You) PGT" và gửi PGTIOU. Người yêu cầu sau khi nhận được 1 PGTIOU trong phản hồi CAS, cả 1 PGT và 1 PGTIOU từ proxy callback được đưa ra như giá trị tham số pgturl khi yêu cầu được gửi, sẽ sử dụng PGTIOU để tương quan các PGT với các phản ứng xác nhận. Người yêu cầu sau đó sẽ sử dụng PGT cho việc có được các PT, nếu người yêu cầu nhận diện chính xác.

Ticket granting ticket IOU (TGTIOU):

Trên 1 ticket validation, 1 dịch vụ của thể yêu cầu 1 PT. Trong CAS 2, con đường để chúng ta xác thực là đúng là yêu cầu dịch vụ gửi đến PGT, PGTIOU cặp đến 1 proxy callback URL đuy định như 1 tham số yêu cầu. Proxy callback URL này phải trên 1 kênh an toàn. Chúng ta xác mình chứng chỉ của nó. Khả năng nhận callback này xác nhận. Sau đó chúng ta trở lại trong xác nhận ticket phản ứng TGTIOU. Từ phản ứng, các dịch vụ mở rộng từ TGTIOU và sử dụng nó để tra cứu TGT từ nơi nó được lưu trữ.

Login Ticket (LT):

Một LT là 1 chuỗi được tạo ra bởi /login như một người yêu cầu chứng chỉ và vượt qua /login như là người chấp nhận chứng chỉ cho username/password. Mục

đích của nó là ngăn chặn sự phát lại các thông tin do lỗi trong trình duyệt, LT cấp bởi /login phải là duy nhất và nên bắt đầu các ký tự "LT-".

Tất các các CAS tickets và giá trị của GTC phải bao gồm dữ liệu ngẫu nhiên an toàn để không là 1 ticket có thể đoán được trong thời gian hợp lý thông qua các cuộc tấn công brute-force [http://vi.wikipedia.org/wiki/Brute_force] và cũng phải chứa các ký tự từ tập hợp {A-Z, a-z, 0-9, and ký tự đặc biệt ?-'} Ticket-granting ticket, service tickets, proxy tickets and login tickets chỉ phải có giá trị trong 1 lỗ lực xác thực. Có hay không xác thực thành công, CAS sau đó phải mất hiệu lực những tickets này gây ra tất cả những nỗ lực xác thực trong tương lai với điều đó thể hiện của vé đặc biệt thất bại. CAS sẽ hết hạn hiệu lực vé dịch vụ trong một thời gian hợp lý (tối đa 5 phút) sau khi được ban hành. Nếu một dịch vụ trình bày để xác nhận service ticket hết hạn, CAS phải đáp ứng với một phản ứng không xác nhận.

2.2.7. Nguyên tắc hoạt động

2.2.7.1.Chứng thực người dùng với CAS server.

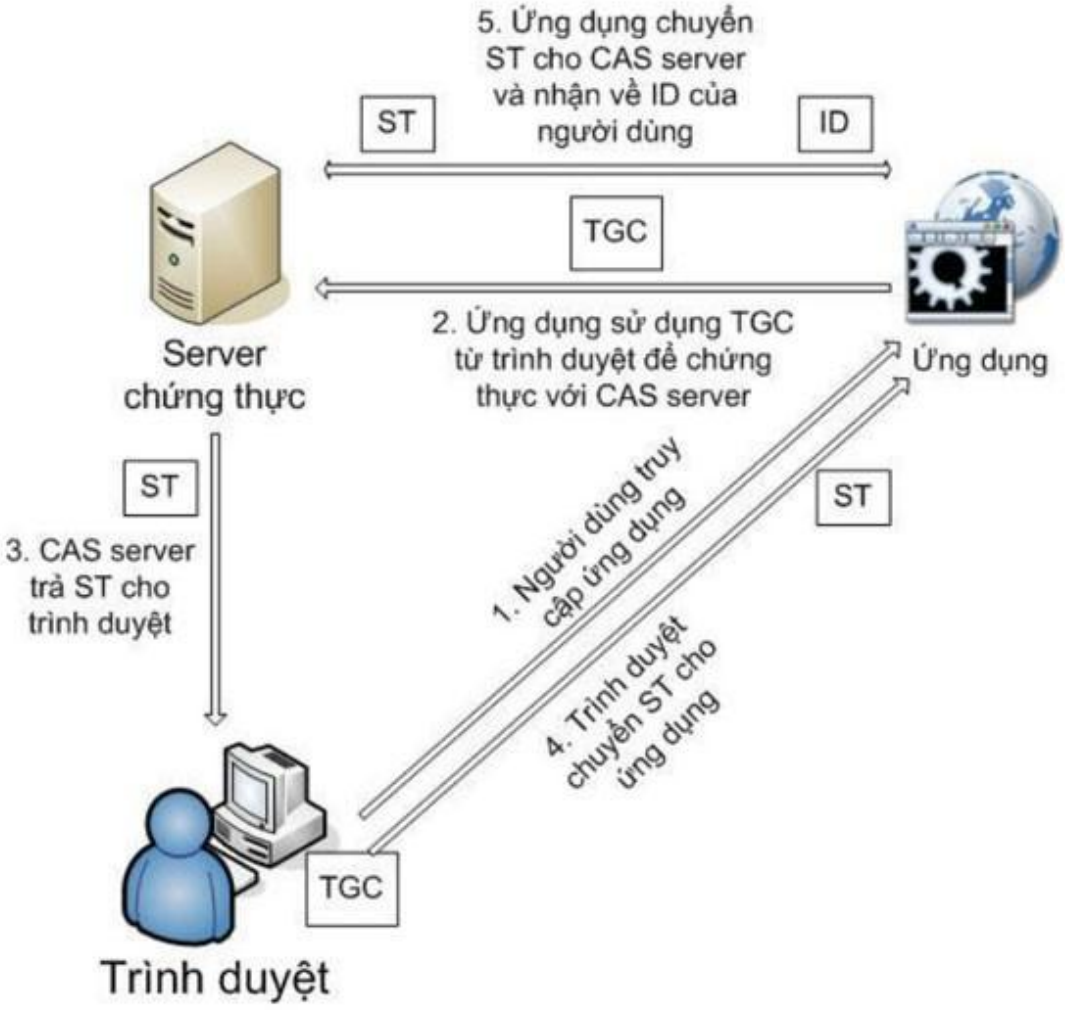
Người dùng nhập username và password vào khung đăng nhập. các thông tin được truyền cho CAS server thông qua giao thức HTTPS hoặc HTTP (tùy theo cách người dùng đặt)

Xác thực thành công, TGC được sinh ra và thêm vào trình duyệt dưới hình thức cookie. TGC này sẽ được sử dụng để SSO với tất cả các ứng dụng.

Truy cập ứng dụng.

- Người dùng truy cập vào ứng dụng khi đã chứng thực với CAS server.
  - Người dùng truy xuất ứng dụng thông qua trình duyệt,
  - Ứng dụng lấy TGC từ trình duyệt và chuyển nó cho CAS server thông qua giao thức HTTPS/HTTP.
  - Nếu TGC này là hợp lệ, CAS server trả về 1 ST cho trình duyệt, trình duyệt truyền ST vừa nhận cho ứng dụng.
  - Ứng dụng sử dụng ST nhận được từ trình duyệt và sau đó chuyển nó cho CAS
  - CAS sẽ trả về ID của người dùng cho ứng dụng, mục đích là để thông báo với ứng dụng người dùng này đã được chứng thực bởi CAS

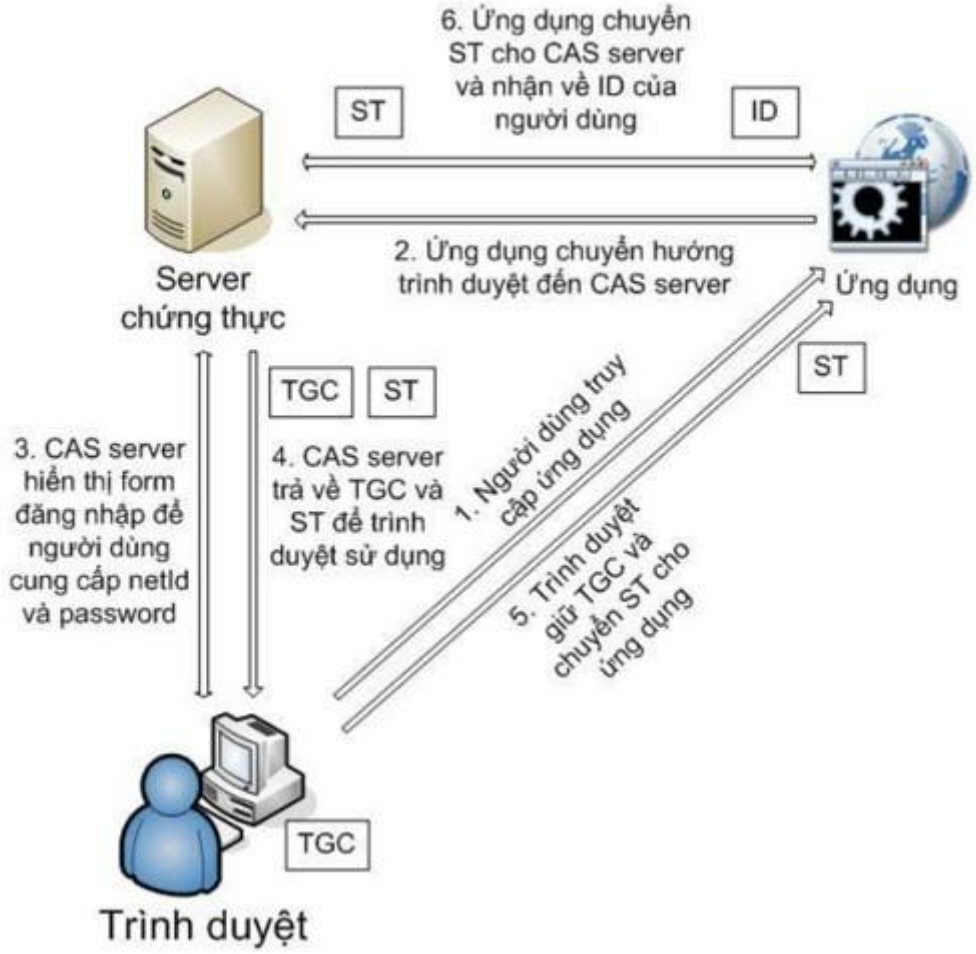
- Ứng dụng đăng nhập cho người dùng và bắt đầu phục vụ người dùng.



Hình 2.1: Người dùng truy cập vào ứng dụng khi đã chứng thực với CAS.

- Người dùng truy cập vào ứng khi khi chưa chứng thực với CAS.
  - Người dùng truy cập vào ứng dụng, vì chưa nhận được TGC nên ứng dụng sẽ chuyển hướng người dùng đến CAS.
  - Người dùng cung cấp username/password thông qua khung đăng nhập để CAS xác thực. Thông tin được chuyển đi bởi giao thức HTTPS hoặc HTTP
  - Xác thực thành công, CAS sẽ chuyển cho trình duyệt đồng thời TGC và ST.

- Trình duyệt sẽ giữ lại TGC để phục vụ cho việc truy cập vào ứng dụng khác và truyền cho ứng dụng ST.
- Ứng dụng chuyển ST cho CAS và nhận về ID của người dùng.
- Ứng dụng đăng nhập và bắt đầu phục vụ



Hình 2.2: Người dùng truy cập vào ứng dụng khi chưa chứng thực với CAS server.

Dưới đây là phần mô tả chi tiết quả trình hoạt động xác thực của CAS.

Dịch vụ chứng thực trung tâm (CAS) được thiết kế như 1 ứng dụng web độc lập. Nó hiện đang được thực hiện như 1 số Java servlets và chạy thông qua máy chủ HTTP/HTTPS. Nó được truy cập thông qua 3 URL mô tả dưới đây. URL login, URL validation, và các tùy chọn URL logout.

Để sử dụng dịch vụ chứng thực trung tâm (CAS), 1 ứng dụng chuyển hướng tới người dùng của nó, hoặc chỉ đơn giản là tạo ra 1 siêu liên kết (hyperlink) đến

URL login. Ví dụ Yale's CAS login URL là https://domain.com/cas/login. Người dùng cũng có thể truy cập vào URL này nếu họ muốn xác thực trước cái sessions của họ.

URL login xử lý thực tế và xác thực chính. Có nghĩ là nó nhắc nhở người dùng cung cấp 1 username và 1 password và xác thực nó với 1 nhà cung cấp chứng thực. Đặc biệt những người triển khai CAS sẽ cắm chung hoặc tùy chỉnh PasswordHandlers để xác thực tên người dùng và mật khẩu với bất kỳ cơ chế xác thực thích hợp.

Để cho phép khả năng tái xác thực sau đó, CAS cũng cố gắng gửi 1 cookies trong bộ nhớ(1 trong đó sẽ bị hết hạn khi đóng trình duyệt) lại cho trình duyệt. Cookies này cái mà chúng ta gọi là Ticket Granting Cookies xác định người dùng khi đã đăng nhập thành công. Cần lưu ý tằng cookies này là bắt buộc trong cơ chế xác thực CAS. Với nó, người dùng đạt được sự xuất hiện của đăng nhập 1 lần (SSO) cho nhiều ứng dụng web. Đó là khi người dùng nhập vào username và password của mình chỉ 1 lần nhưng có quyền truy cập vào tất cả dịch vụ nào sử dụng CAS. Nếu không có các tập tin cookie, người dùng sẽ cần phải nhập username và password của mình mỗi khi ứng dụng chuyển hướng nó đến CAS (Người dùng có thể chỉ đạo CAS phá hủy các tập tin cookie này bằng cách gọi đến URL logout.

Ví dụ https://domain.com/cas/logout).

Ngoài việc xử lý xác thực chính, CAS cũng lưu ý cách dịch vụ mà người sử dụng đã được chuyển hướng hoặc liên kết từ đó. Nó có thể làm điều này bởi vì các ứng dụng chuyển hướng hoặc liên kết một người dùng đến URL login được yêu cầu cũng phải vượt qua dịch vụ định danh CAS. Nếu xác thực thành công, CAS tạo ra 1 số dài và ngẫu nhiên mà chúng ta gọi là 1 ticket. Sau đó liên kết ticket này với người dùng xác thực thành công và các dịch vụ mà người sử dụng đã cố gắng xác thực. Có nghĩa là, nếu người sử dụng được thông qua từ dịch vụ S, CAS tạo ra T ticket cho phép người dùng truy cập vào dịch vụ S. Ticket này được thiết kế như 1 chứng chỉ chỉ sử dụng 1 lần. Nó hữu ích cho người dùng, chỉ cho dịch vụ S và chỉ sử dụng 1 lần. Nó hết hạn ngay sau khi nó được sử dụng.

Sau khi xác thực hoàn tất, CAS chuyển hướng trình duyệt của người dùng trở lại ứng dụng mà nơi người dùng truy cập vào. Nó biết cái URL để chuyển hướng người dùng đến vì các thảo luận serviceID ở trên cũng có chức năng như một callback URL. Đó là các định danh mà một ứng dụng sử dụng phải đại diện cho 1

URL mà một phần hoặc ít nhất kết hợp với ứng dụng này(Có nghĩa là mỗi 1 ứng dụng có 1 URL riêng). CAS chuyển hướng trình duyệt của người dùng chuyển hướng trở lại các URL này và thêm cái ticket đã được thảo luận ở trên (Service Ticket) như 1 tham số yêu cầu.

Để làm sáng tỏ điều này thì có 1 ví dụ như sau : Giả sử tôi muốn xác thực người dùng trước khi họ truy cập vào http://localhost/en, khi người dùng đăng nhập để sử dụng http://localhost/en, nó sẽ được chuyển hướng sang

https://localhost:8082/login?service= http://localhost/en/processing.php

Giả sử processing.php là một phần ứng dụng webPHP. Trang web này được thiết kế để mong đợi 1 chuỗi ticket để được thông qua với nó như 1 tham số yêu cầu đặt tên ticket.Trang PHP này chỉ cần xác nhận ticket khi nhận được nó bằng cách thông qua URL validation với tham số ticket.

Trang PHP cần sắp xếp yêu cầu đến URL này và đọc dữ liệu URL đó. Khi xây dựng các yêu cầu này, các trang PHP cũng cần phải vượt qua các serviceID đã được sử dụng trước đây khi chuyển hướng người dùng đến URL login. Để làm điều này, nó sử dụng các tham số yêu cầu đặt tên dịch vụ.

Khi CAS nhận được 1 ticket thông qua URL validation, nó sẽ kiểm tra CSDL nội bộ của mình để xác định xem nó đã tồn tài chưa hay chỉ vừa mới nhận được. Nếu nó đã làm và các dịch vụ liên quan đến ticket khớp với các dịch vụ đã được thông qua bởi các ứng dụng cái mà yêu cầu xác thực. Nó sẽ trả lại các username liên quan với ticket tới các ứng dụng yêu cầu. Nếu không nó từ chối xác nhận yêu cầu.

Giao thức mà URL validation sử dụng trả lại dữ liệu cho các ứng dụng yêu cầu là đơn giản. CAS sẽ phản ứng với 2 dòng (in a text/plain HTTP response), dòng đầu tiên là “yes” hay “no” tương ứng với ticket là ứng dụng được trình bày hợp lệ hay không? Nếu ticket là hợp lệ, dòng thứ 2 chứa các tên đăng nhập của chủ sở hữu ticket – có nghĩ là việc xác định người sử dụng đã xác thực thành công. Nếu ticket không hợp lệ, dòng thứ 2 là rỗng.

Dưới đây là 1 ví dụ

/validate sẽ trả lại 1 trong 2 câu trả lời sau:

Xác thực ticket thành công:

yes<LF>

username<LF>

Xác thực ticket thất bại:

no<LF>

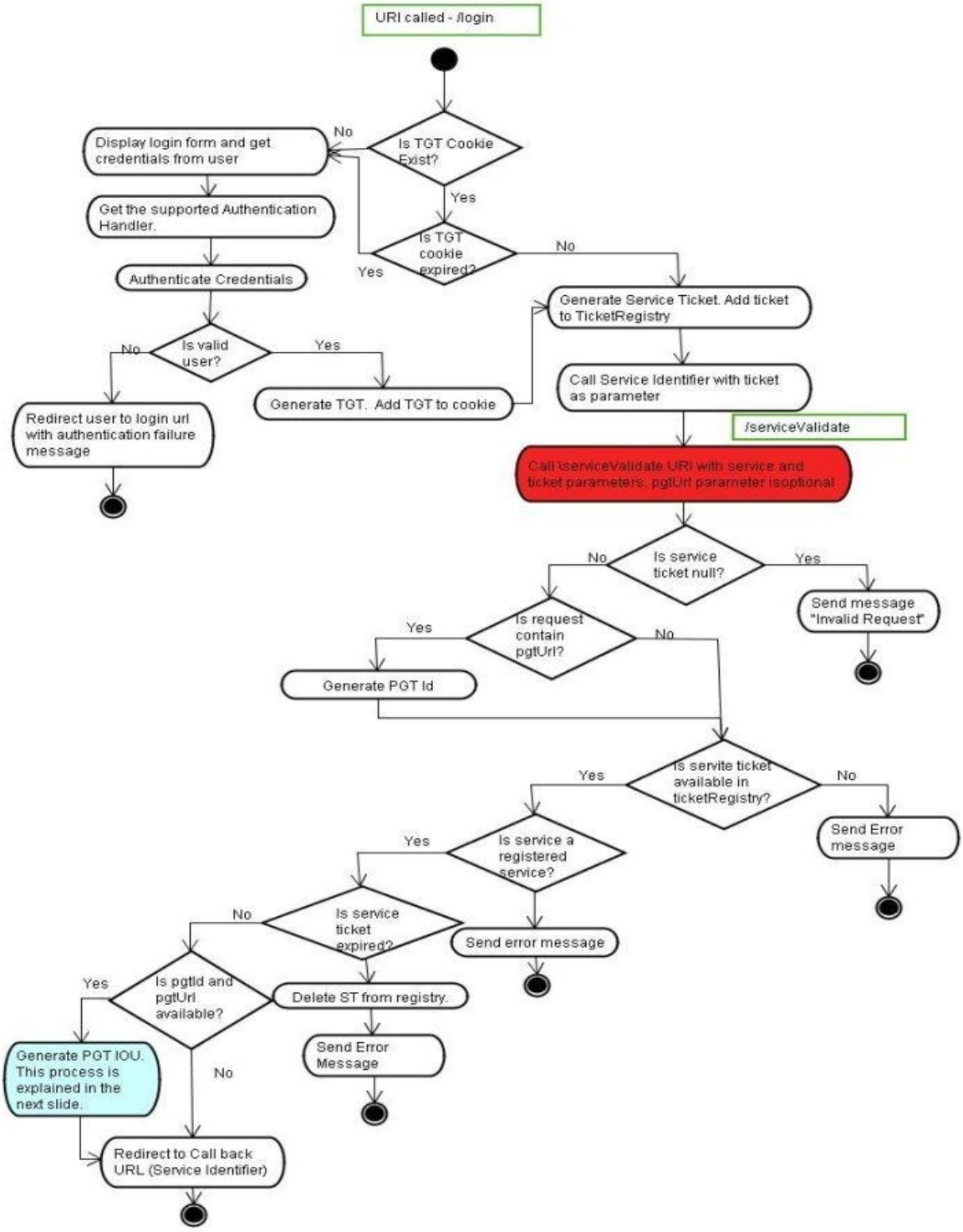
<LF>

Nếu ticket là hợp lệ, CAS ngay lập tức loại bỏ nó để không sử dụng 1 lần nữa.Khi chu trình hoàn thành, một ứng dụng web đã có thể xác minh danh tính của người dùng mà không bao giờ có quyền truy cập vào mật khẩu của người dùng đó. Hơn nữa, trong trường hợp của người dùng chấp nhận cookies, thì có thể tái xác định người sử dụng CAS để người dùng không cần phải nhập username và password của mình trong tương lai. (Hiện nay, trong bộ nhớ " ticket-granting cookies " vẫn hoạt động trong tám giờ.).

Hiện nay, ngoài username mà khi xác thực thành công CAS trả lại cho client thì hệ thống CAS còn được tùy biến trả lại cho client nhiều thông tin khác nữa. Nó được gọi là Extra user attributes. Phần này sẽ được thể hiện rõ trong phần thực nghiệm hệ thống để thấy rõ điều đó.

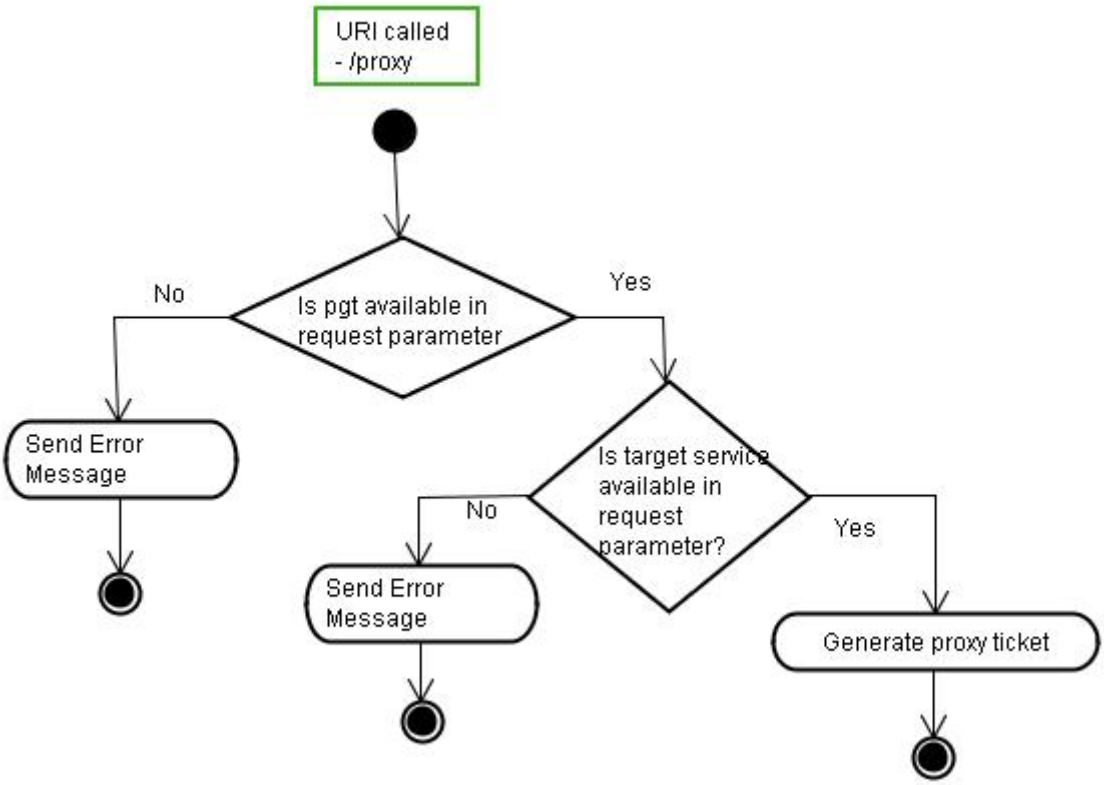
2.2.8. Kiến trúc tổng quan CAS.

2.2.8.1./login flow



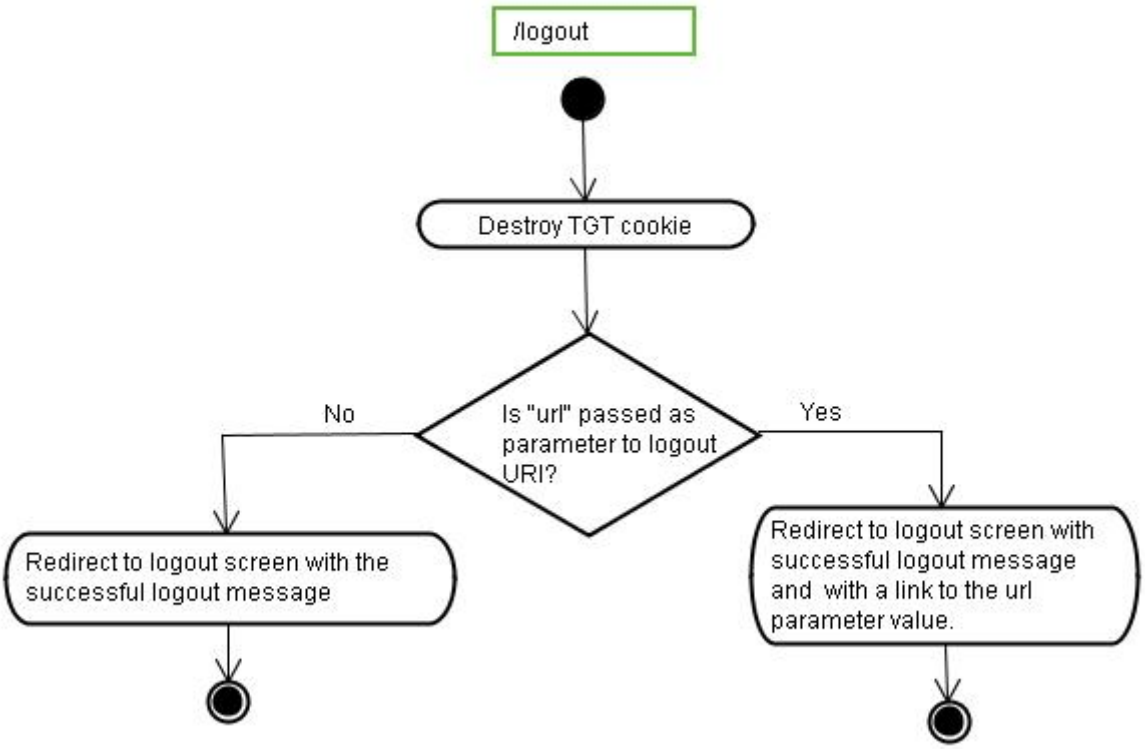
Hình 2.3: Login flow

2.2.8.2./proxy flow



Hình 2.4: Proxy flow.

2.2.8.3. /logout flow



Hình 2.5: logout flow.

2.3. Ruby CAS.[6]

RubyCAS-server là một thực thi đầy đủphía server của JA-SIG's CAS protocol, nó cung cấp giải pháp cross-domainSSO cho các ứng dụng web.

Khái quát

RubyCAS-Server đưa cho bạn:

- Một trang đăng nhập độc lập nơi mà người dùng có thể nhập thông tin của họ. (ví dụ username và password).
- Một cơ chế xác thực thông tin người dùng dựa vào nhiều backends khác nhau (1 bảng trong SQL database, ActiveDirectory/LDAP, Google accounts, vv.).
- Một back-end xác nhận các client applications nơi CAS cho phép kết nối để kiểm tra xem người dùng hiện hành được xác thực (nếu người dùng đã được chứng thực với máy chủ CAS, sau đó họ được phép tiếp tục, nếu không họ được chuyển hướng tới trang đăng nhập CAS server của để xác thực).

- Có khả năng tương thích mở với rất nhiều nền tảng (PHP framwork, various Java frameworks,.NET, Zope, vv).
- Đa ngôn ngữ bản địa hóa, RubyCAS-servers sẽ tự động phát hiện ngôn ngữ ưa thích của người dùng và trình bày giao diện thích hợp.

RubyCAS-server được thực hiện bằng cách sử dụng Sinatra microframework, và được thiết kế cho dễ dàng triển khai hoặc như một máy chủ độc lập (qua WEBrick hoặc Mongrel) hoặc dưới Apache (thông qua Rack). Nó hoàn toàn thực hiện các giao thức CAS 2.0 cùng với một số tiện ích mở rộng không chính thức hiện nay trong ứng dụng khách tham khảo cho JA-SIG 3.x phiên bản.

2.4. CAS Client.

2.4.1. Giới thiệu ngôn ngữxây dựng website phía client.

A. PHP là gì?

PHP (viết tắt hồi quy "PHP: Hypertext Preprocessor") là một ngôn ngữ lập trình kịch bản hay một loại mã lệnh chủ yếu được dùng để phát triển các ứng dụng viết cho máy chủ, mã nguồn mở, dùng cho mục đích tổng quát. Nó rất thích hợp với web và có thể dễ dàng nhúng vào trangHTML. Do được tối ưu hóa cho các ứng dụng web, tốc độ nhanh, nhỏ gọn, cú pháp giống C và Java, dễ học và thời gian xây dựng sản phẩm tương đối ngắn hơn so với các ngôn ngữ khác nên PHP đã nhanh chóng trở thành một ngôn ngữ lập trình web phổ biến nhất thế giới.

Ngôn ngữ, các thư viện, tài liệu gốc của PHP được xây dựng bởi cộng đồng và có sự đóng góp rất lớn của Zend Inc., công ty do các nhà phát triển cốt lõi của PHP lập nên nhằm tạo ra một môi trường chuyên nghiệp để đưa PHP phát triển ở quy mô doanh nghiệp.

2.5. Thư viện phpCAS.[7]

2.5.1. phpCAS requirements.

- Webserver
  - Mọi webserver như Apache, IIS hay những cái khác đều hoạt động.
  - CURL (7.5+)
  - Thư viện CRUL phải được bật trong hệ thống và phải được biên soạn với sự hỗ trợ SSL.

cURL là một hàm hay của PHP. Hàm này giúp ta lấy, chiết tách hay đọc nội dung một trang web khác ngay trên Server của chúng ta. Một thuận lợi lớn nhất mà hàm curl này mang lại đó là tốc độ, nhanh hơn rất nhiều so với hàm open file gần gấp 3 lần. cURL được ví như một công cụ giao tiếp đa giao thức, giúp ta xem hoặc tải một địa chỉ.

- PHP >= 5.0 (PHP >= 4.2.2 for 1.1.x)

phpCAS users phải có PHP compiled với các tùy chọn sau:

- --with-curl: Hỗ trợ CURL, cần để truy cập vào các proxy.
- --with-openssl: hỗ trợ SSLt, cần cho fopen('https://...'), để kiểm tra tính hợp lệ của CAS ticket;
- --with-dom: hỗ trợ DOM, để đọcXML responses of the CAS server (PHP4);
- --with-zlib: hỗ trợ Zlib, cái này cần bởi DOM.

Khi nó được sử dụng trên Horde FrameWork:

- --with-gettext: Hỗ trợ gettext.

Khi nó được sử dụng trên Horde IMP:

- --with-imap: hỗ trợ IMAP và POP, cần khi sử dụng IMP;
- --with-kerberos: hỗ trợ Kerberos, cần bởi IMAP.

Khi lưu trữ thông tin người dùng Horde đến cơ sở dữ liệu MySQL:

- --with-mysql: hỗ trợ MySQL.

Ghi chú:

- PHP> = 4.3.0 là cần thiết để có được thông tin đăng nhập đầy đủ (nhờ debug_backtrace ()).
- Trên một số hệ thống (Fedora Core 2 ví dụ), gói php_domxml là bắt buộc.
- SSL

Nếu bạn có kế hoạch viết một proxy CAS, bạn sẽ cần phải đảm bảo máy chủ Apache của bạn với OpenSSL. HTTPS cấu hình là cần thiết để sử dụng CAS proxy

(URL gọi lại cho máy chủ CAS để truyền tải các PGTIou phải được bảo vệ). Để đạt được điều này, chỉnh sửa file httpd.conf và thêm dòng như:

SSLCertificateFile /etc/x509/cert.server.pem

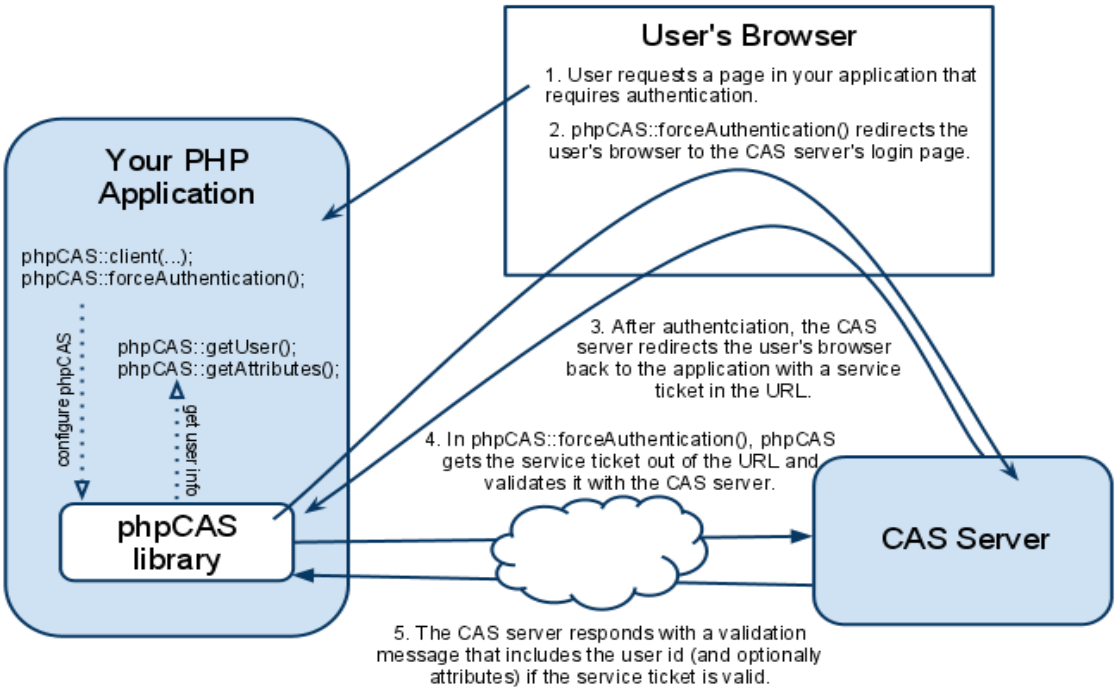
SSLCertificateKeyFile /etc/x509/key.server.pem

SSLCertificateChainFile /etc/x509/cachain.pem

SSLCACertificateFile /etc/x509/cacert.pem

2.5.2 phpCAS examples.

Thư viện phpCAS cung cấp một API đơn giản để xác thực người sử dụng với CAS server. phpCAS được cấu hình bằng cách sử dụng phương pháp API tĩnh như phpCAS :: client () và phpCAS :: setCasServerCACert (). Sau khi phpCAS đã được cấu hình, một cuộc gọi đến phpCAS :: forceAuthentication () thực hiện quá trình đăng nhập người dùng hiện hành chưa được xác thực, chuyển hướng đến trang đăng nhập của CAS server. Sau khi phpCAS :: forceAuthentication () đã được gọi, id người dùng hiện hành có thể truy cập thông qua phpCAS :: getUser ().



Hình 2.6: Nguyên tắc hoạt động phpCAS.

2.5.3. phpCAS logout.

Đăng xuất từ phpCAS được thực hiện bằng cách gọi một trong những phương thức phpCAS sau: phpCAS::logoutXxx(). Khi gọi 1 trong những phương thức đó thì sẽ có các hành động cụ thể như:

- Phá hủy PHP session hiện tại
- Chuyển hướng trình duyệt đến CAS server
- Phá hủy CAS session

Hành vi của CAS server phụ thuộc vào:

- Phương thức logout được gọi
- Cách cấu hình

phpCAS::logout()

Sau khi logout, CAS sẽ show trang logout.

phpCAS::logoutWithRedirectService($service)

Sau khi logout, CAS server chuyển hướng trình duyệt tới cái URL được đưa ra.

phpCAS::logoutWithUrl($url)

- Yêu cầu phiên bản CAS servers > 3.3.5.

Sau khi logout, CAS server show 1 trang với cái link URL được đưa vào.

phpCAS::logoutWithRedirectServiceAndUrl($service, $url)

- Yêu cầu phiên bản CAS servers > 3.3.5.

Nếu chuyển hướng được kích hoạt. CAS server chuyển hướng trình duyệt đến URL được cung cấp ($service) và các tham số $url được bỏ qua.

Nếu không, CAS server cho thấy một trang với một liên kết đến các URL được cung cấp.

phpCAS::logout($params)

Service và các tham số url có thể cũng vượt qua như trong mảng:

Bảng 2.2: Danh sách tham số phpCAS.

all with an array	shortcut
logout(array())	logout()
logout(array('service'=>'www.myservicesite.com'))	logoutWithRedirectService('www.myservicesite.com')
logout(array('url'=>'www.myurlsite.com'))	logoutWithUrl('www.myurlsite.com')
logout(array('service'=>'www.myservicesite.com', 'url'=>'www.myurlsite.com'))	logoutWithRedirectServiceAndUrl('www.myservicesite.com', 'www.myurlsite.com')

2.5.4. phpCAS troubleshooting.

Để phát hiện được lỗi, vui lòng kích hoạt phpCAS debug log.

phpCAS::setDebug($filename);

Logfile này mặc định là phpCAS.log hoặc là có trong / tmp (Linux / Unix) hoặc trong windows của bạn thư mục temp. Bạn luôn luôn có thể chỉ định một tập tin như $ filename. Ngoài ra kiểm tra các bản ghi máy chủ web cho bất kỳ lỗi nào.

Không có Proxy-granting ticket IOU (PGTIOU) được truyền đi khi đang kiểm tra tính hợp lệ của 1 ST hoặc 1 PT

Có lẽ là máy chủ CAS không tin tưởng ứng dụng. Ứng dụng phpCAS cần phải được truy cập thông qua https và giấy chứng nhận phải được tin cậy bởi các máy chủ CAS. (Thêm một keystore có chứa các chứng chỉ của máy chủ ứng dụng của bạn và các chuỗi xác nhận vào máy chủ CA của bạn)

Nếu nhận được tin nhắn thông báo, cảnh báo nói rằng tiêu đề đã được gửi đi, và authentication fails.

Thêm dòng bên dưới vào trước phương thức phpCAS được gọi

error_reporting(E_ALL & ~E_NOTICE);

Và thêm dòng bên dưới vào trong file php.ini:

error_reporting=E_ALL & ~E_NOTICE)

2.6. Vấn đề về bảo mật cho SSO.

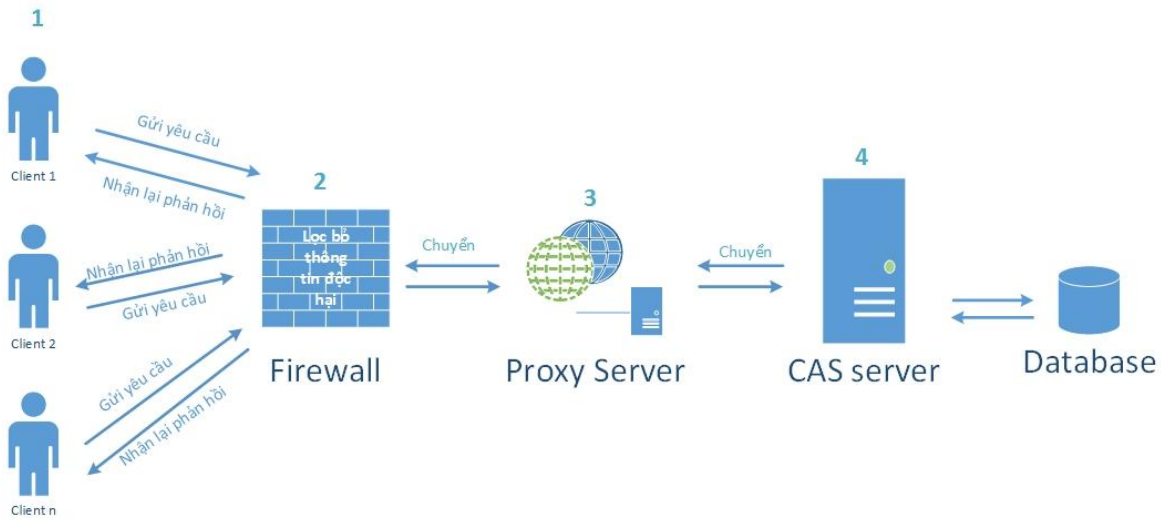
Hiện nay vấn đề bảo mật đang là vấn đề được quan tâm hàng đầu khi triển khai 1 hệ thống nào đó, nó là mấu chốt của thành công. Khi triển CAS cũng không ngoại trừ việc đó. Dưới đây là 1 vài thông tin em tìm hiểu được trong quá trình nghiên cứu và triển khai CAS.

- Về phía CAS-server.

Như đã nói ở chương I, để triển khai hệ thống SSO không đơn giản chỉ cài đặt, cấu hình và tích hợp mà nó còn rất nhiều vấn đề cần lưu tâm. Bản thân CAS chỉ là 1 ứng dụng, nó mang tính phục vụ hơn để có thể đạt hiệu suất tốt nhất phục vụ cho các yêu cầu từ client. Bình thường bản thân CAS cũng có tích hợp thêm chức năng bảo mật cho chính bản thân nó nhưng nếu để triển khai hệ thống lớn nó sẽ không được dùng vì khi dùng nó sẽ bị giảm hiệu suất phục vụ, tiêu hao nhiều tài nguyên hệ thống.

Vậy đặt ra câu hỏi là thế vậy nó sẽ bảo vệ như thế nào trước các cuộc tất công như DOS, DDOS, FLOOD….?

Trả lời: CAS được triển khai lớp trong cùng của kiến trúc IT của 1 tổ chức, nó sẽ được bao bọc, bảo vệ bởi các tường lửa (firewall),các máy chủ ủy quyền (proxy), …Mọi việc ngăn chặn các tấn công từ bên ngoài sẽ được hệ thống bảo vệ chặn lại và xử lý trước khi đến được với phần CAS server. Thế nên khi quyết định triển khai SSO thì cần phải được tính toán kỹ lưỡng về mặt chi phí, vấn đề bảo mật…



Hình 2.7: Sơ đồ vị trí CAS trong hệ thống mạng.

- Về phía CAS-client.

Cơ chế bảo mật cho CAS cũng phải được chú trọng ngay từ phía client. Bạn đừng nghĩ là chỉ cần CAS server được bảo vệ tốt có nghĩa là không thể bị tấn công. Client là nơi tiếp nhận yêu cầu đầu tiên, nó cũng là nơi mà người dùng trực tiếp làm việc và chuyển yêu cầu đến CAS Server. Bạn thử hình dung nếu mà CAS client bị sập thì CAS server cũng không còn ý nghĩa gì nữa. Một lần nữa xin nhắc lại về vấn đề để triển khai SSO là 1 vấn đề cần phải được đánh giá kỹ càng trước khi triển khai.

CHƯƠNG IIITHỰC NGHIỆM.

3.1. Cài đặt hệ thống.

3.1.1. Điều kiện cần thiết.

A. Yêu cầu phần cứng tối thiểu.

- Intel® Xeon® Quad Core Processor E3-1220 (8M Cache, 3.10 GHz)
- 2GB PC3-10600E UDIMMs DDR3
- Hard Drive: 140Gb.
- ServeRAID C100 for IBM System x® supports RAID-0; -1
- Power Supply 350 W fixed
- IBM Prefer KYB USB US ENG 103P & IBM 3 Button Optical Mouse USB

B. Yêu cầu phần mềm.

- RubyInstaller(Version 1.9.3-p448e)Development Kit(Version 32-4.5.2-20111229-1559) cho Ruby được cài đặt.
- Git (version 1.8.4-preview20130916) được cài đặt.
- Bundle được cài đặt
- pgAdmin III (version 1.18.1) được cài đặt.

3.1.2. Giới thiệu.

3.1.2.1. RubyInstaller.

RubyInstaller dự án cung cấp một dựa trên Windows installer khép kín có chứa một môi trường thực hiện ngôn ngữ Ruby, thiết lập một đường cơ sở của yêu cầu RubyGems và tiện ích mở rộng.

3.1.2.2. Development Kit.

Ruby Development Kit là bộ công cụ phát triển Ruby. Nó bao gồm nhiều chương trình tiện ích như trình biên dịch ruby(ruby compiler), chương trình gỡ lỗi, trình phát sinh tài liệu, đóng gói dữ liệu v.v...

3.1.2.3. Git.

Giả lập môi trường linux trên windows.

3.1.2.4. Bundle.

Bundle có chức năng quản lý các version, nó sẽ tải các thư viện cần thiết đã được khai báo trong file config.yml về.

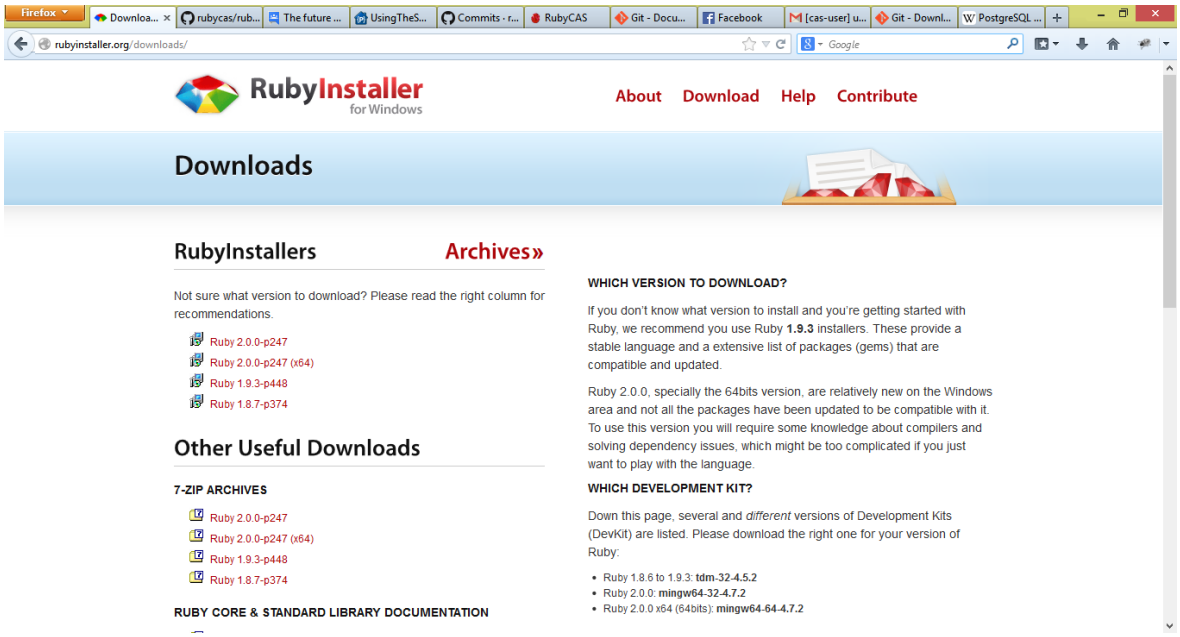
3.1.2.4. pgAdmin III.

Cung cấp Postgresql tool (version 9.31).

3.1.3. Cài đặt CAS-server.

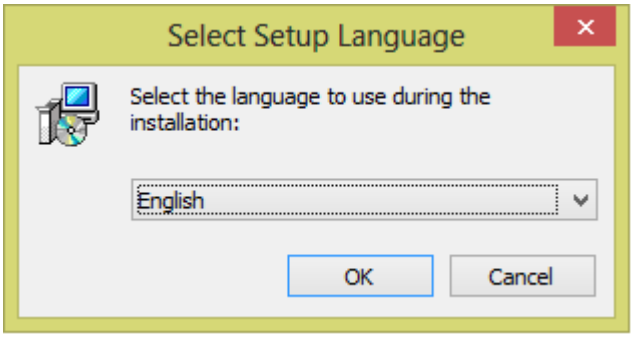
Tải RubyInstaller và Development Kittại:

http://rubyinstaller.org/downloads/



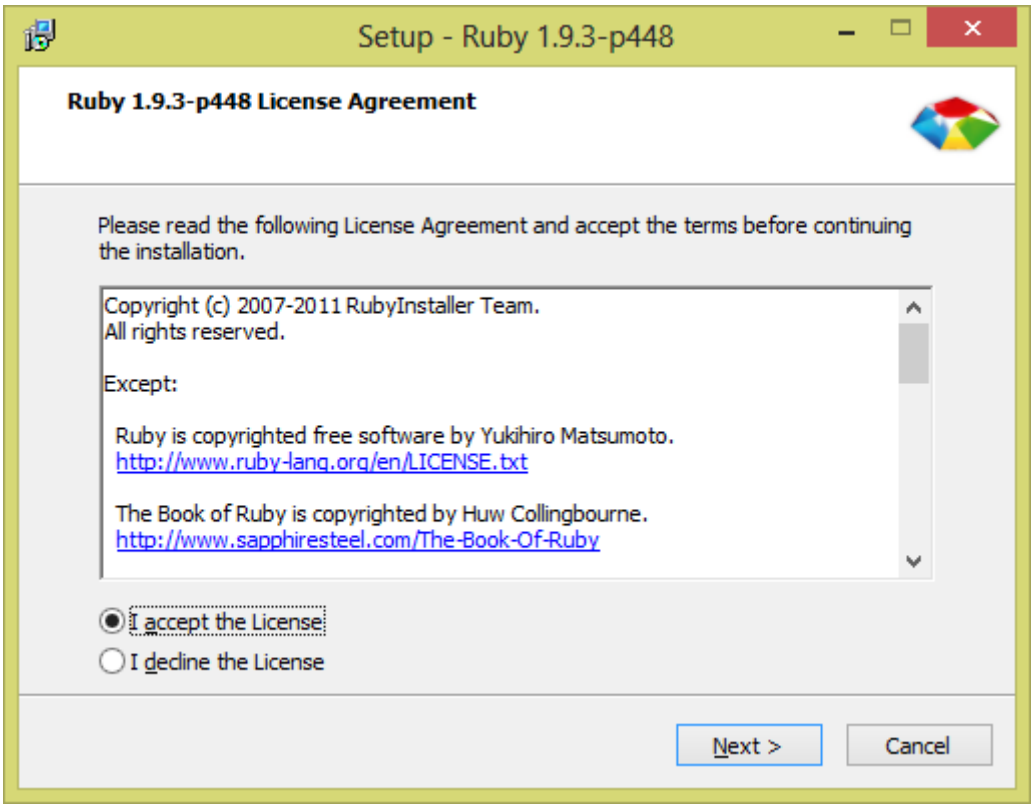
Hình 3.1: Tải RubyInstaller

Cài đặt RubyInstallertheo các bước dưới đây.



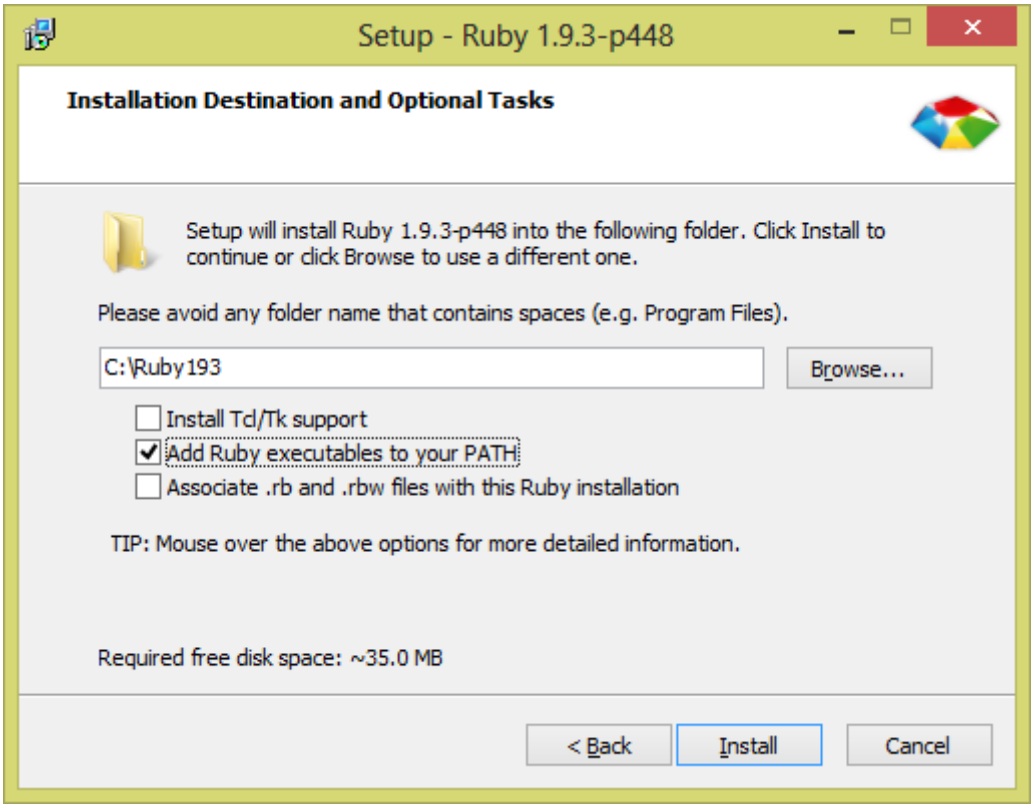
Hình 3.2: Cài đặt RubyInstaller bước1.

Bước 1: Chọn ngôn ngữ và nhấn ok.



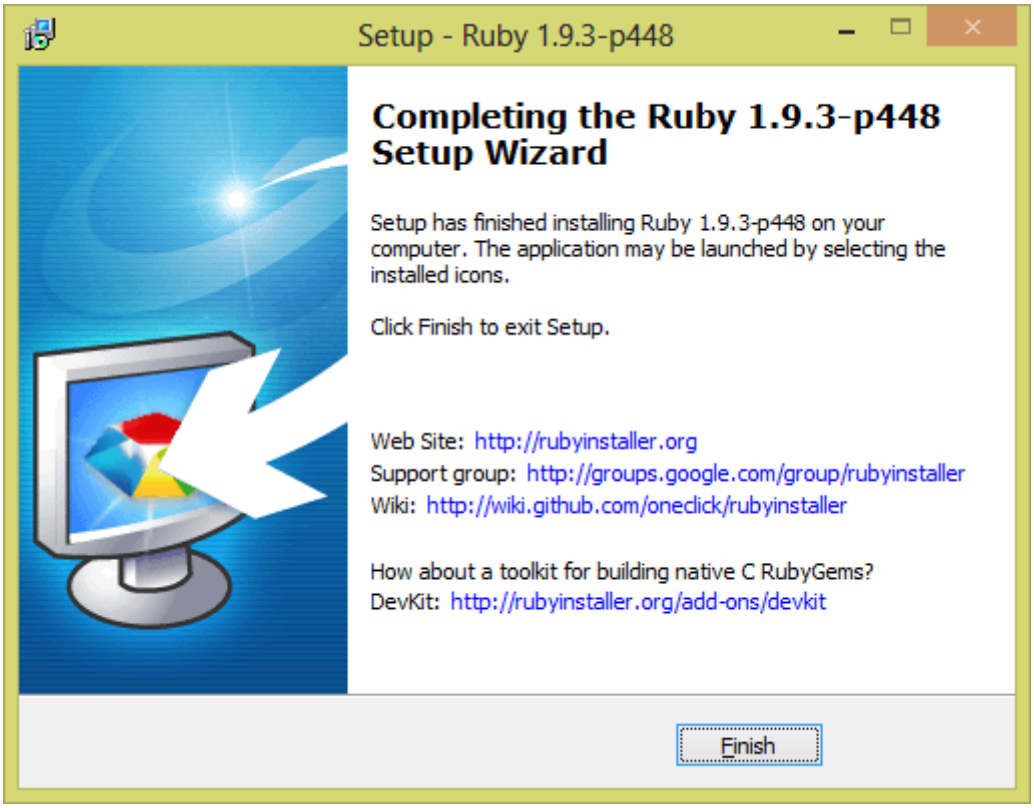
Hình 3.3: Cài đặt RubyInstaller bước2.

Bước 2: Chọn “I accept the License” và nhấn next.



Hình 3.4: Cài đặt RubyInstaller bước 3.

Bước 3: Chọn đường đường dẫn thư mục cài đặt Ruby và click chọn "Add Ruby excutables to your PATH". Nhấn Install.

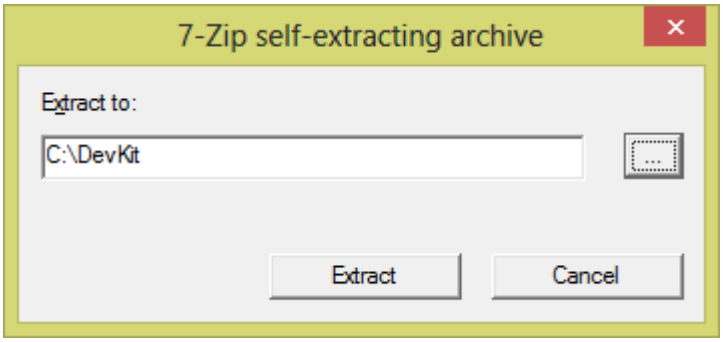


Hình 3.5: Cài đặt RubyInstaller bước4.

Bước 4: Nhấn Finish.

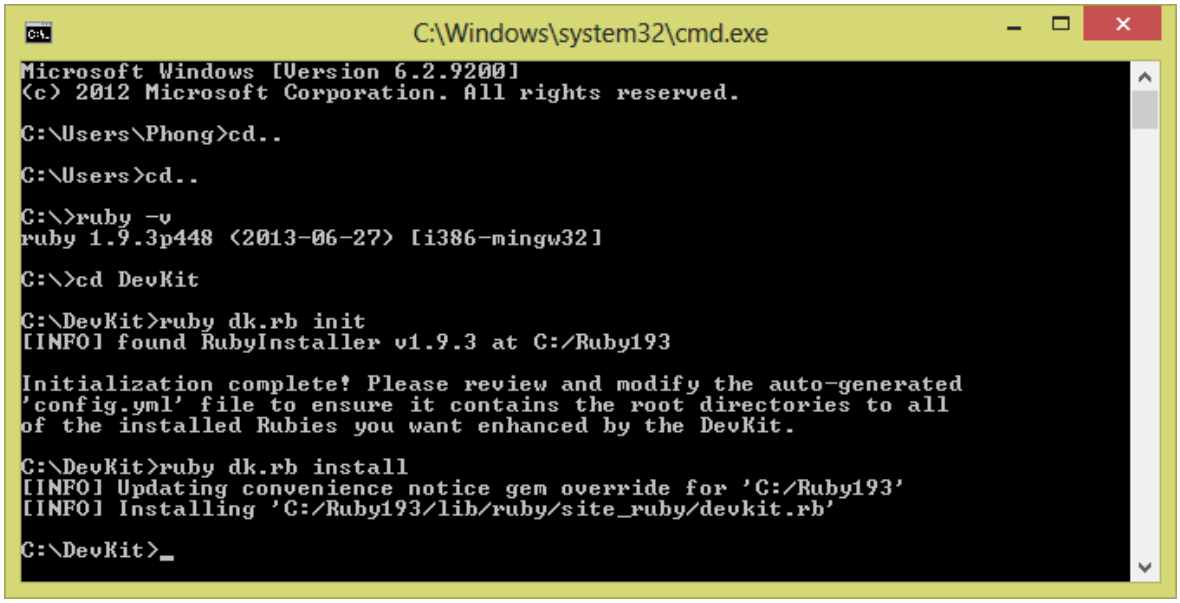
Giờ chuyển sang phần cài đặt Development Kit.

Mở Development Kit đã tải về, khi đó xuất hiện cửa sổ, ta chọn thư mục giải nén toàn bộ nội dung của Development Kit và nhấn Extract.



Hình 3.6: Giải nén Development Kit

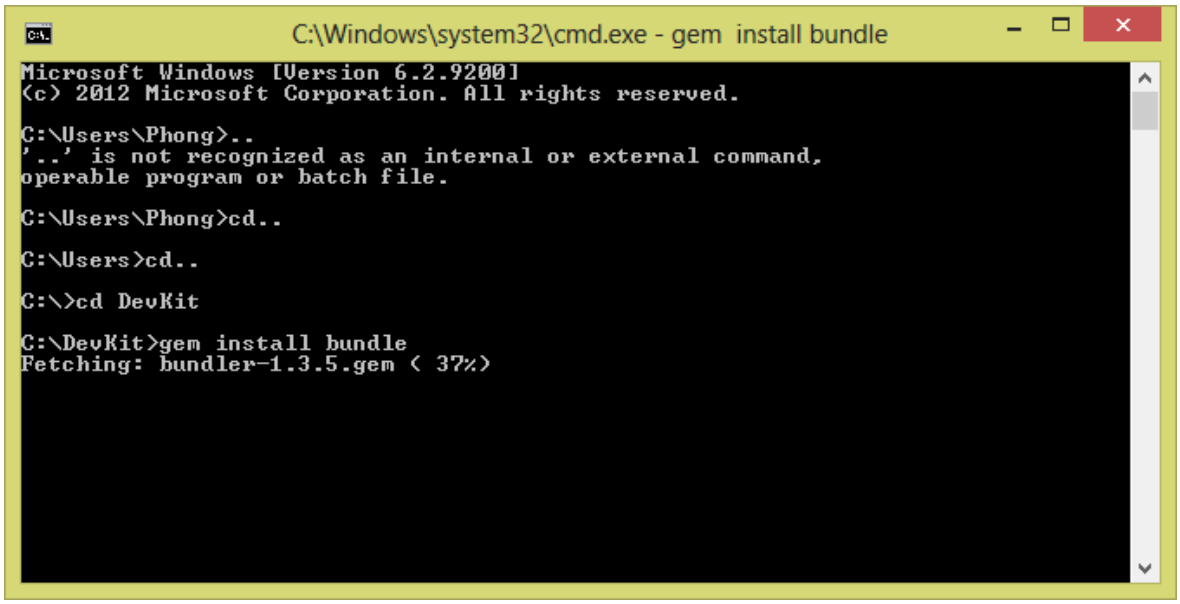
Bước 5: Mở "cmd" và làm theo hình bên dưới.



Hình 3.7: Cài đặt RubyInstaller bước 5.

Bước 6: Cài đặt "bundle". Hiện tại thì trong cửa sổ "cmd.exe" thì vì trí đang ở "C:\DevKit" thì tại đây ta gõ lệnh:

Gem install bundle

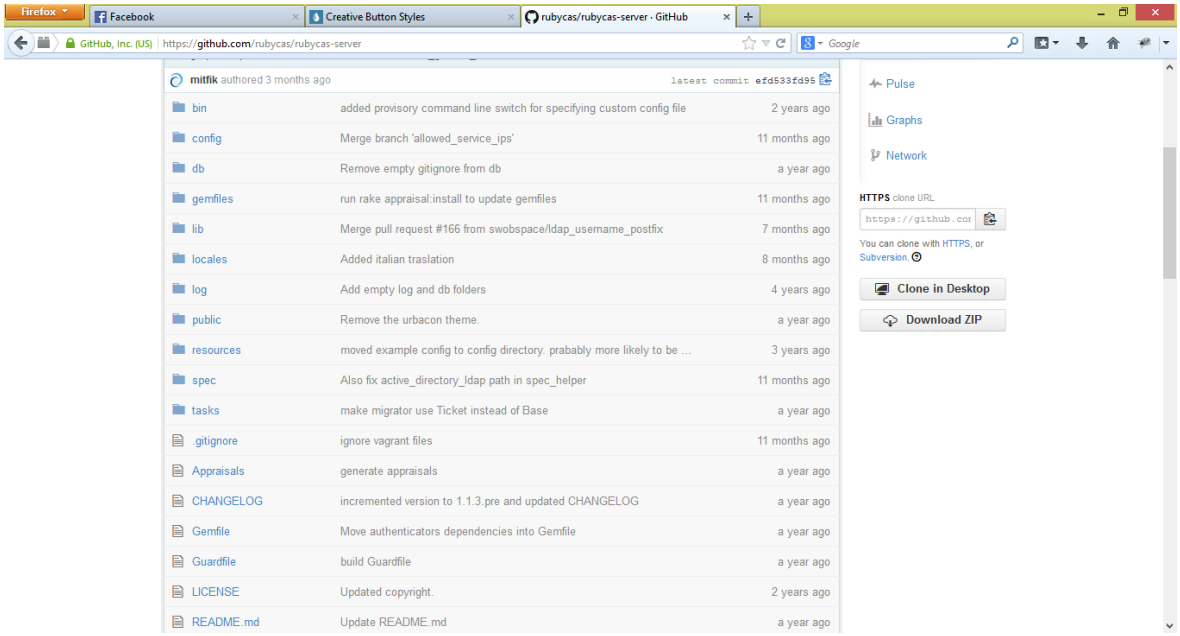


Hình 3.8: Cài dặt Bunlde.

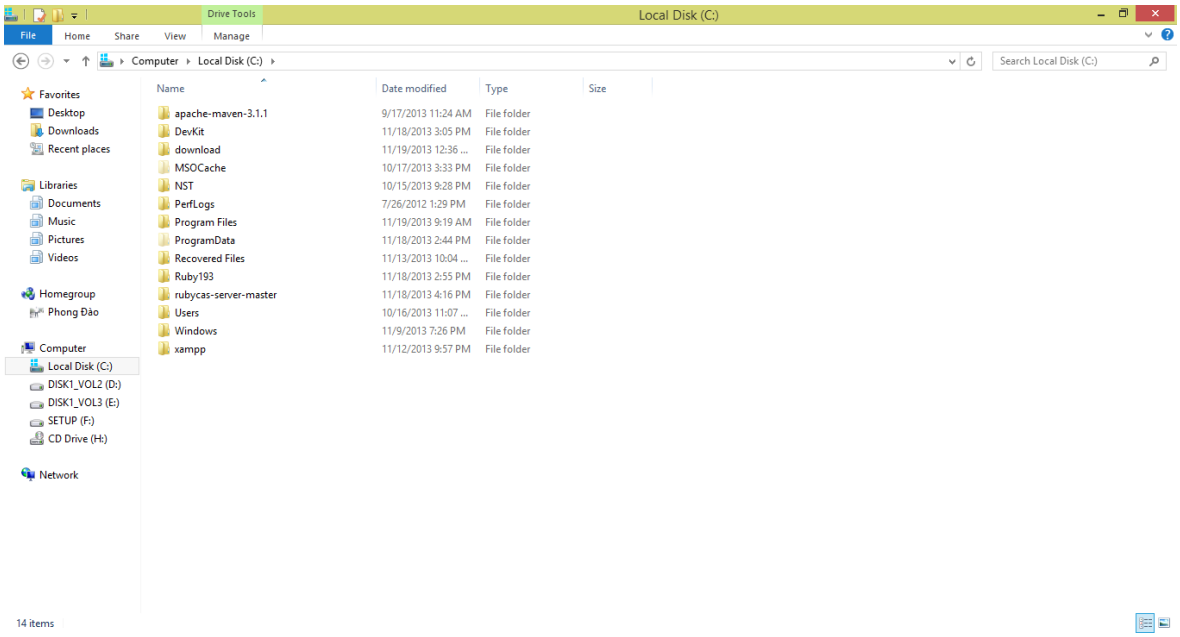
Chờ quá trình cài đặt hoàn tất và ta chuyển sang bước tiếp theo.

Bước 7: Tải bộ mã nguồn và giải nén bất kỳ chỗ nào tùy theo ý thích. Ở đây tôi giải nén ở trong phân vùng C.

Download RubyCAS tại: https://github.com/rubycas/rubycas-server



Hình 3.9: Tải mã nguồn RubyCAS.



Hình 3.10: Triển khai RubyCAS bước1.

Bước 8: Sao chép file "config.example.yml" trong config và dán ra thư mục bên ngoài ngang hàng với index. Sửa tên thành "config.yml", mở file "config" với notepad++.

Tìm đến dòng 31, 32, 33:

server: webrick

port: 443

ssl_cert: /path/to/your/ssl.pem

Sửa thành:

server: webrick

port: 8082

#: /path/to/your/ssl.pem

Việc sửa như vậy giúp tắt SSL, tùy vào mục đích sử dụng mà bạn cân nhắc tắt hay không. Ở đây tôi tắt đi cho dễ xử lý.

Tìm đến dòng 101:

database:

adapter: mysql

database: casserver

username: root

password:

host: localhost

reconnect: true

Sửa thành thông tin kết nối CSDL để RubyCAS lấy thông tin xác thực, nếu bạn đọc hết hướng dẫn ở trong file thì bạn sẽ thấy có rất nhiều kiểu để cho chúng ta chọn.Ở đây tôi dùng postgresql nên tôi sẽ sửa thành:

database:

adapter: postgresql

database: cas

host: 127.0.0.1

port: 5432

username: cas

password: 123456

reconnect: true

Tìm đến dòng 202 và thêm đoạn này vào sau:

authenticator:

class: CASServer::Authenticators::SQLEncrypted

database:

adapter: postgresql

database: cas

host: 127.0.0.1

port: 5432

username: cas

password: 123456

user_table: users

username_column: username

password_column: password

extra_attributes: username,permission,full_name,actived

encrypt_function: 'require "digest/md5"; user.password == Digest::MD5.hexdigest("#{@password}")'

Trong đó:

adapter: postgresql

database: tên cơ sở dữ liệu.

host: Địa chỉ cơ sở dữ liệu.

port: cổng kết nối.

username: tên người dùng được phép truy cập vào cơ sở dữ liệu.

password: mật khẩu để truy cập vào cơ sở dữ liệu.

user_table: bảng chứa thông tin người dùng.

username_column: tên của cột chứa username

password_column: tên cột chứa password.

extra_attributes: lấy thêm các thuộc tính khác trong bảng user ngoài username đã được trả ra. Như ở trên ngoài username thì tôi còn lấy được permission,full_name,actived.

encrypt_function: hàm mã hóa mật khẩu.

Tìm đến dòng 467 và thay:

log:

file: /var/log/casserver.log

level: INFO

thành:

log:

file: log/casserver.log

level: INFO

Lý do của việc thay thế này là do ban đầu hệ thống hỗ trợ trên linux nên ta phải sửa thành về windows thì nó mới hoạt động. Lưu và đóng file đó lại.

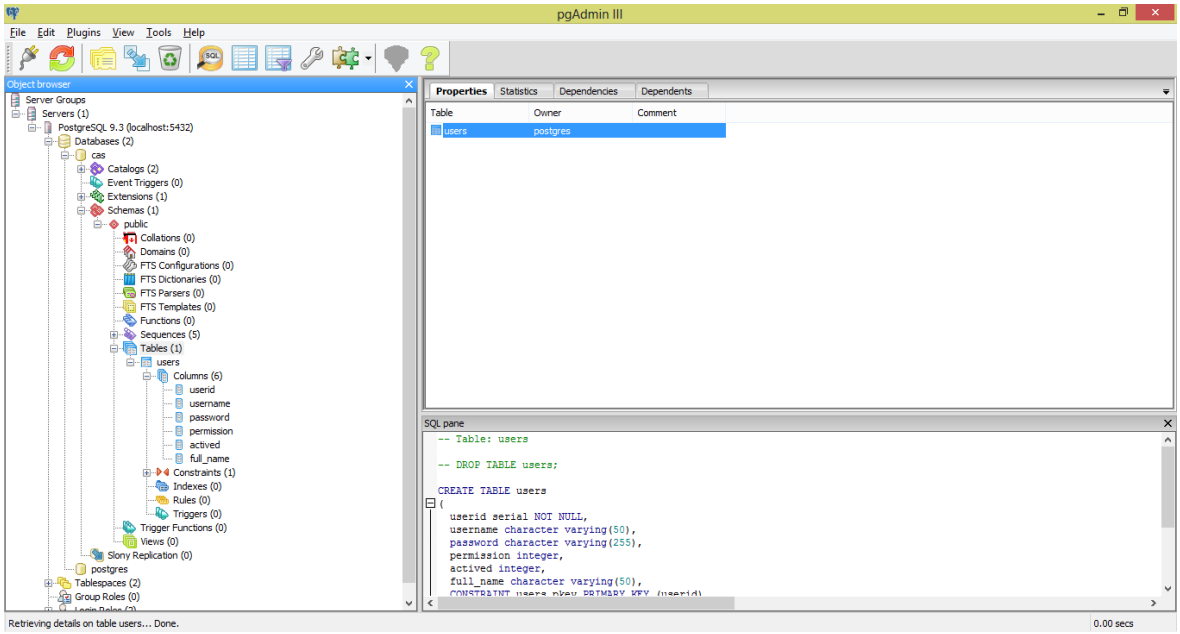
Bước 8: mở file "Gemfile" và thêm vào dòng cuối cùng đoạn sau:

 # gem for postgresql

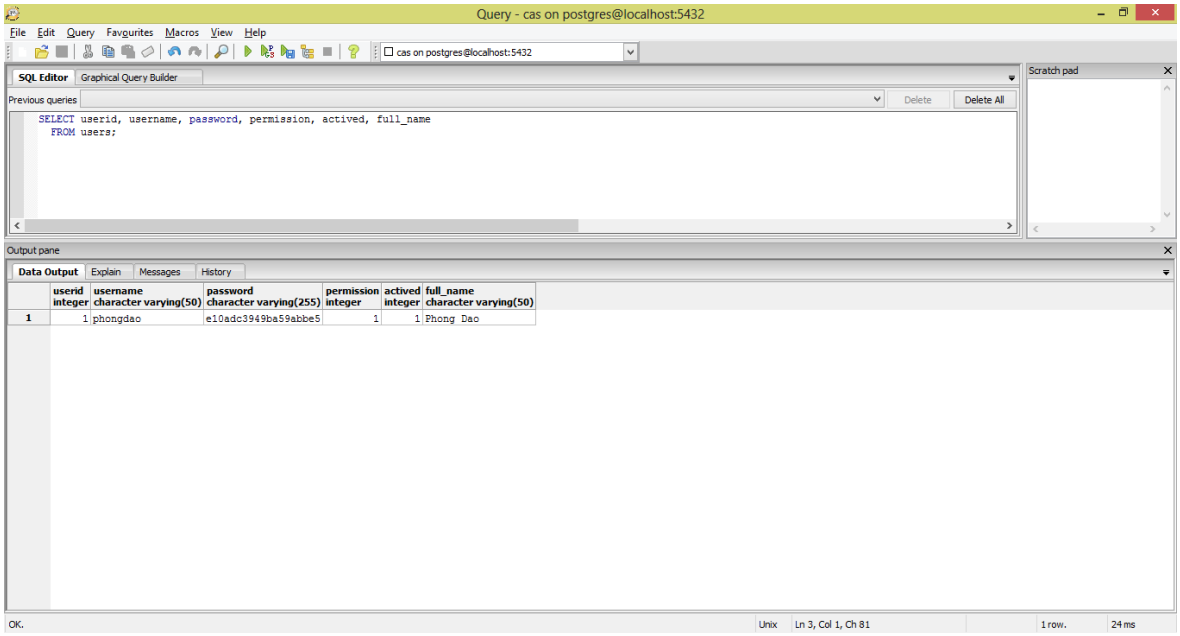
gem "activerecord-postgresql-adapter"

Đoạn này có ý nghĩa là thêm driver để RubyCAS có thể kết nối được với Postgresql. Với những cái khác thì có thể tìm tại đây: http://rubygems.org/gems. Lưu và đóng lại.

Bước 9: Mở pgAdmin III và tạo cơ dữ liệu sao cho nó giống với những gì bạn đã cấu hình trong file config.yml. Sau đó bạn thêm 1 bản ghi vào trong CSDL.

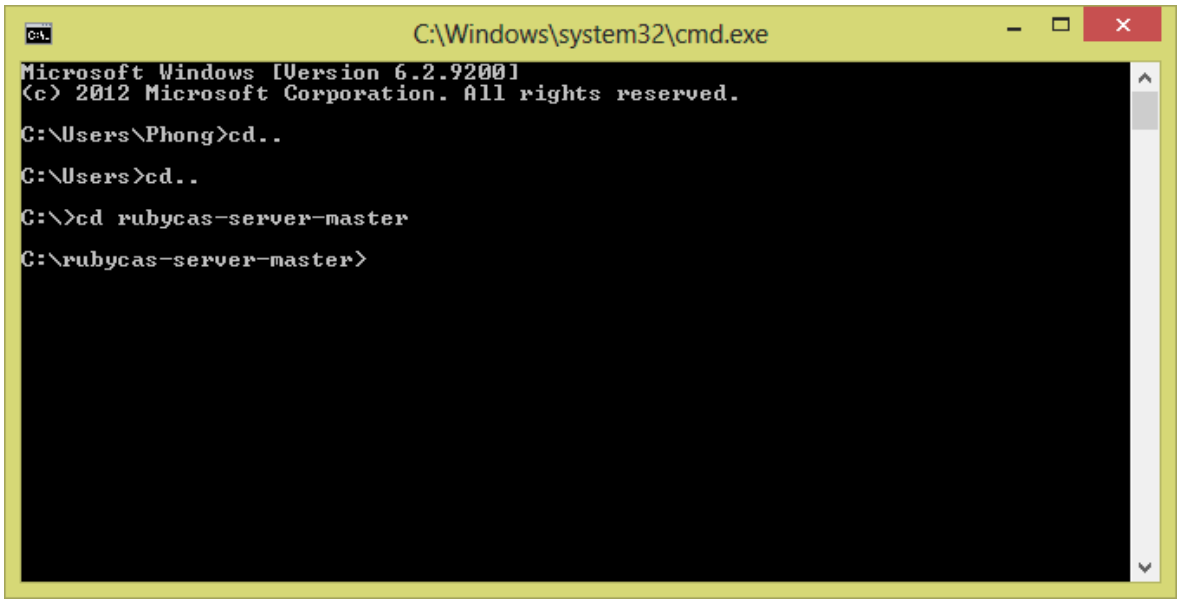


Hình 3.11: Tạo CSDL người dùng cho RubyCAS xác thực.



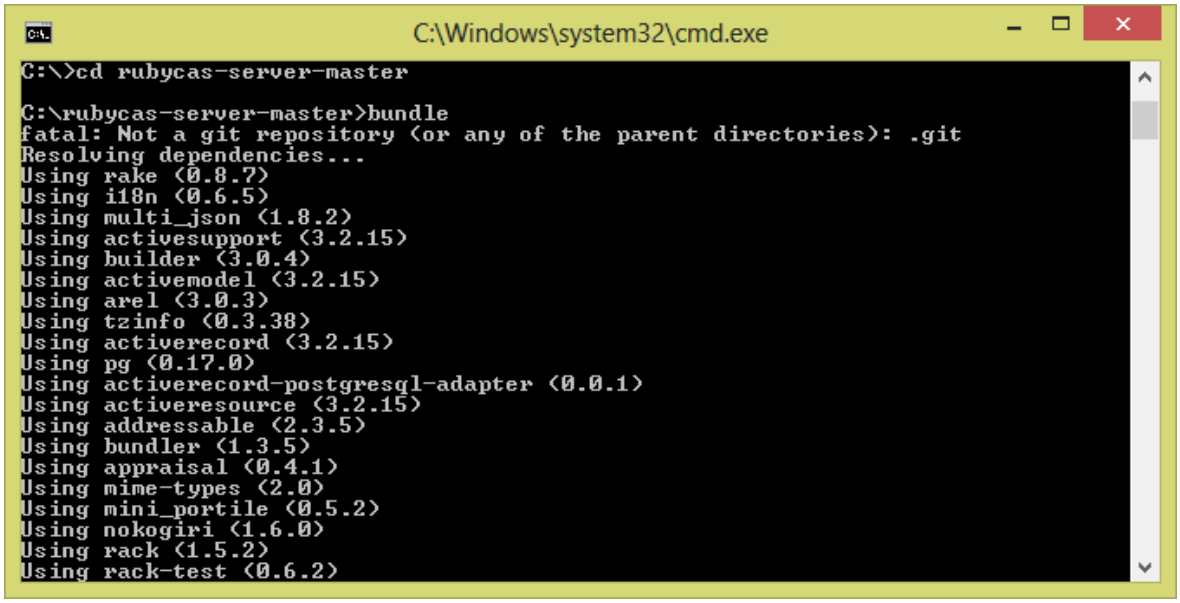
Hình 3.12: Tạo CSDL người dùng cho RubyCAS xác thực 2.

Bước 10: Mở 1 cửa sổ "cmd.exe" mới và cd tới thư mục rubycas-server đã được giải nén từ trước.



Hình 3.13: Triển khai RubyCAS bước 2.

Bước 11: Chạy lệnh "bundle" để tải các thư viên đã được khai báo trong file "rubycas-server.gemspec".

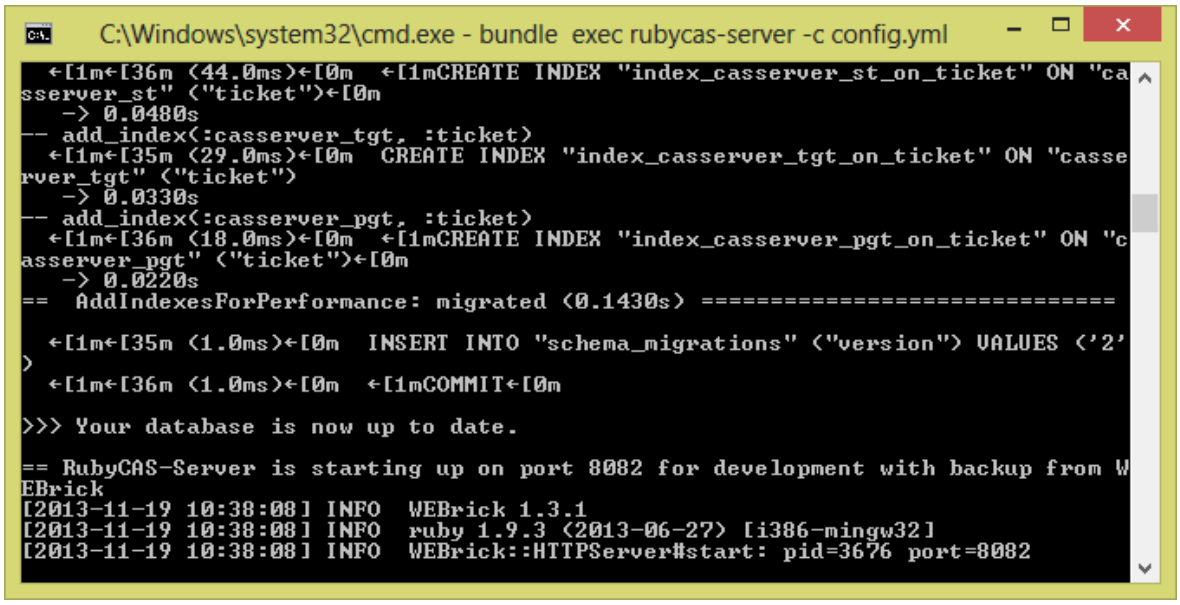


Hình 3.14: Triển khai RubyCAS bước 3.

Bước 12: Sau khi tải các thư viện cần thiết, ta chạy lệnh:

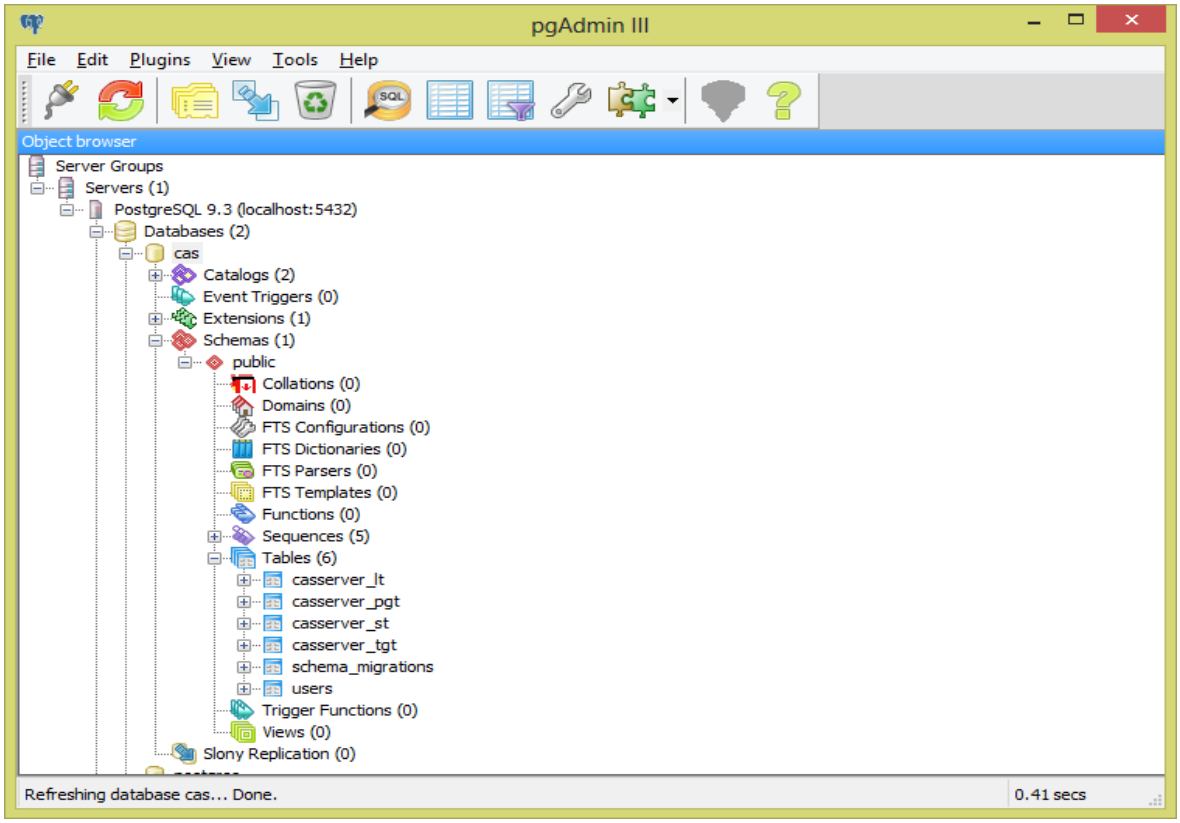
bundle exec rubycas-server -c config.yml

việc chạy lệnh này để hoàn tất quá trình cài đặt RubyCAS-server.



Hình 3.15: Triển khai RubyCAS bước 4.

Bước 13: Trở lại với pgAdmin III, làm tươi CSDL xem sự thay đổi.



Hình 3.16: Triển khai RubyCAS bước 5.

Vậy là xuất hiện thêm 5 bảng nữa, việc này đồng nghĩa với việc RubyCAS-server đã kết nối thành công tới CSDL.

Bảng 3.1: Thông tin table casserver_lt.

Tên Columns	Kiểu dữ liệu	Mô tả
Id	Serial	Là khóa chính của bảng.
Ticket	Character varying (255)	Lưu trữ các LT được CAS tạo ra.
Create_on	Timestamp without time zone	Thời gian tạo LT.
Consumed	Timestamp without time zone	Thời gian sử dụng.
Client_hostname	Character varying (255)	Tên hostname của client.

Bảng 3.2: Thông tin table casserver_pgt.

Tên Columns	Kiểu dữ liệu	Mô tả
Id	Serial	Là khóa chính của bảng.
Ticket	Character varying (255)	Lưu trữ các PGT được CAS tạo ra.
Create_on	Timestamp without time zone	Thời gian tạo PGT.
Client_hostname	Character varying (255)	Tên hostname của client.
Iou	Character varying (255)	Chứa IOU của PGT.
Service_ticket_id	Interger	Chứa ST ID từ table casserver_st

Bảng 3.3: Thông tin table casserver_st.

Tên Columns	Kiểu dữ liệu	Mô tả
Id	Serial	Là khóa chính của bảng.
Ticket	Character varying (255)	Lưu trữ các ST được CAS tạo ra.
Service	Text	Chứa service yêu cầu xác thực.
Create_on	Timestamp without time zone	Thời gian tạo ST.
Consumed	Timestamp without time zone	Thời gian sử dụng.
Client_hostname	Character varying (255)	Tên hostname của client.
Username	Character varying (255)	Chứa thông tin username.

Type	Character varying (255)	
Granted_by_pgt_id	Integer	Chứa pgt_id đã được chấp nhận.
Granted_by_tgt_id	Integer	Chứa tgt_id đã được chấp nhận.

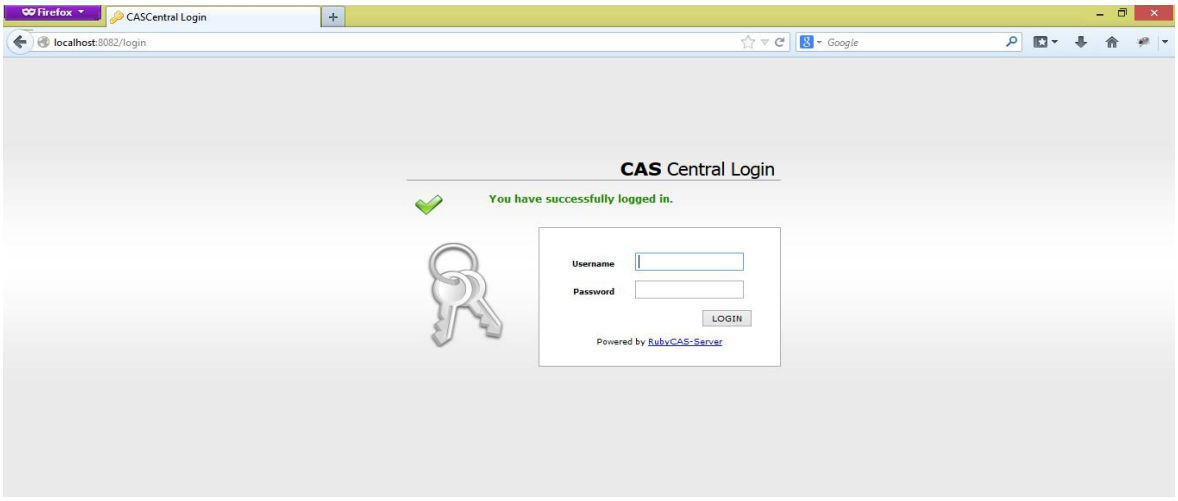
Bảng 3.4: Thông tin table casserver_tgt.

Tên Columns	Kiểu dữ liệu	Mô tả
Id	Serial	Là khóa chính của bảng.
Ticket	Character varying (255)	Lưu trữ các TGT được CAS tạo ra.
Create_on	Timestamp without time zone	Thời gian tạo TGT.
Client_hostname	Character varying (255)	Tên hostname của client.
Username	Character varying (255)	Chứa thông tin username.
Extra_attributes	Text	Chứa các Extra_attributes.

Bước 14: Để kiểm tra chắc chắn rằng CAS đã hoạt động, hãy mở đường dẫn sau: http://localhost:8082/login đăng nhập với thông tin đã thêm vào csdl trước đó là:

Username: phongdao

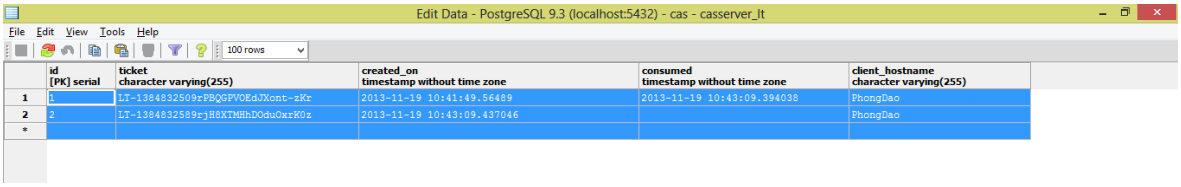
Password: 123456



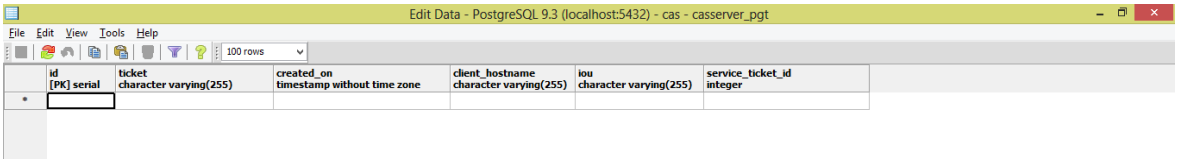
Hình 3.17: Kiểm thử quá trình cài đặt RubyCAS.

Nếu nhận được thông báo như hình trên thì việc cài đặt CAS-server đã thành công.

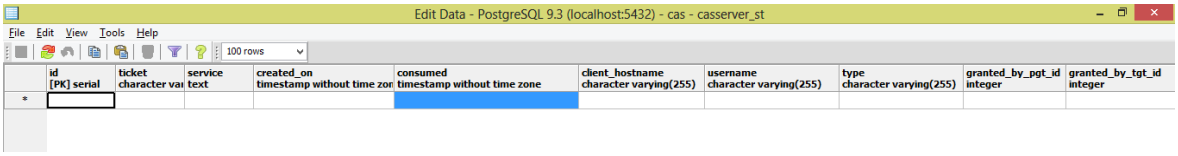
Bước 15: Trở lại với CSDL và xem có cập nhật khi ta đã đăng nhập lần đầu tiên không.



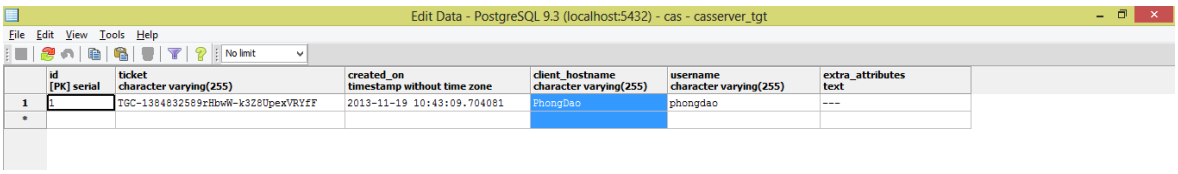
Hình 3.18: Cấu trúc bảng casserver_lt



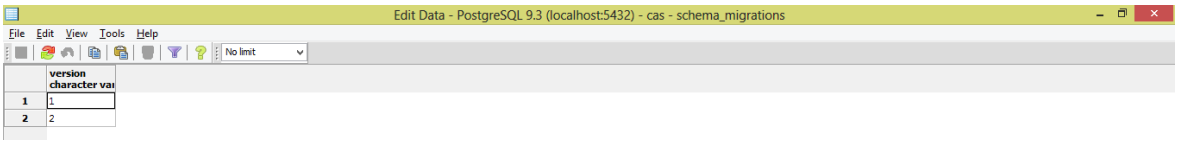
Hình 3.19: Cấu trúc bảng casserver_pgt



Hình 3.20: Cấu trúc bảng casserver_st.



Hình 3.21: Cấu trúc bảng casserver_tgt

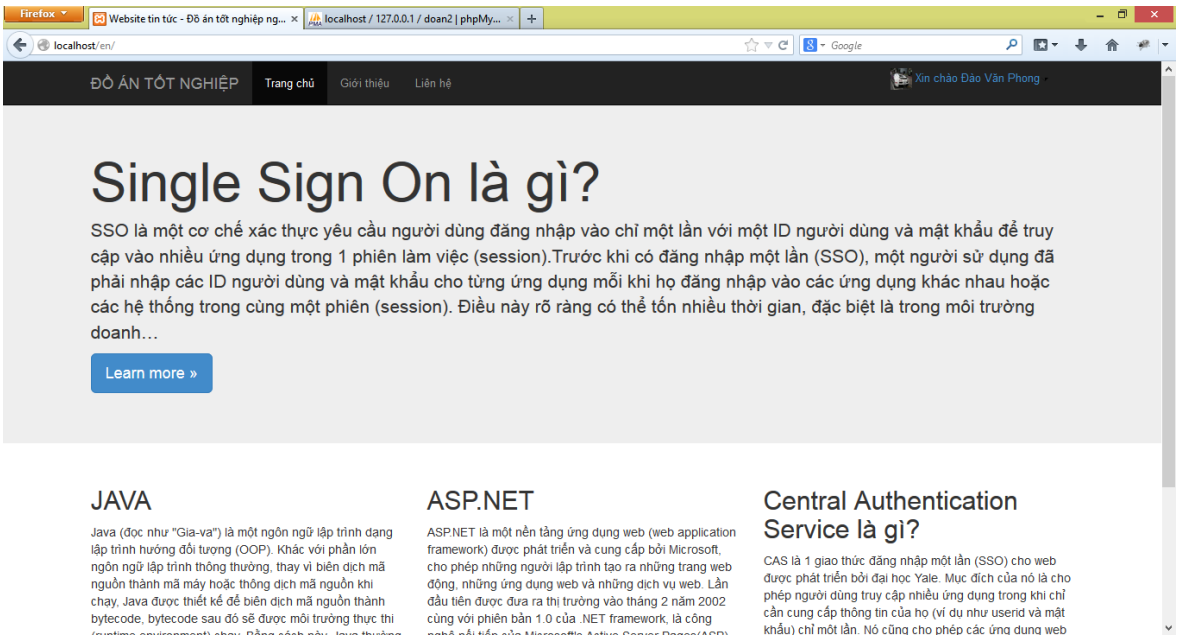


Hình 3.22: Cấu trúc bảng schema_migrations.

3.1.4. Tích hợp CAS client vào hệ thống.

3.1.4.1. Giới thiệu 2 website dùng để tích hợp SSO.

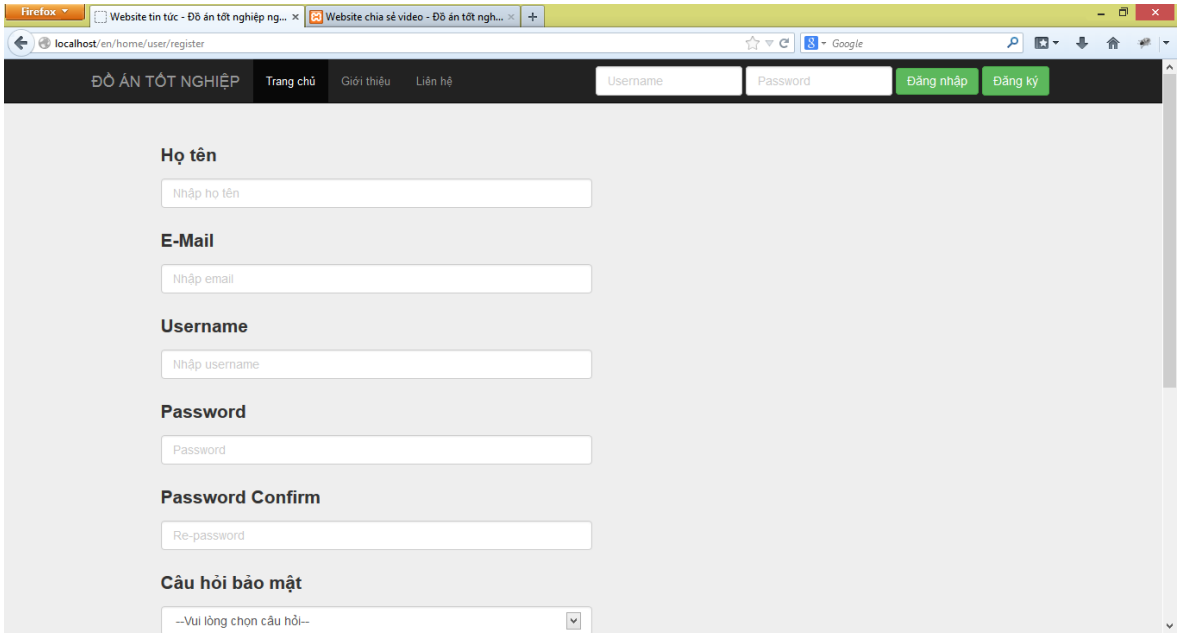
A. Website 1



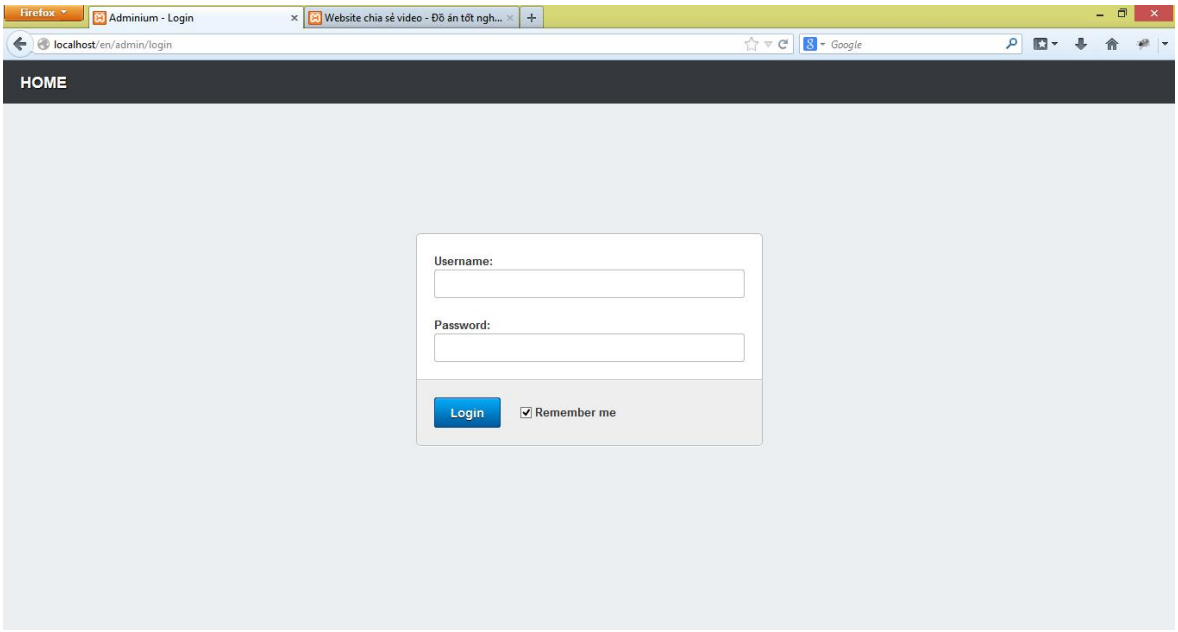
Hình 3.23: Trang chủ website 1.

Website 1 là website tin tức đơn giản nhưng đủ tiêu chuẩn để làm website tích hợp cơ chế đăng nhập 1 lần. Chức năng chính của website 1 bao gồm:

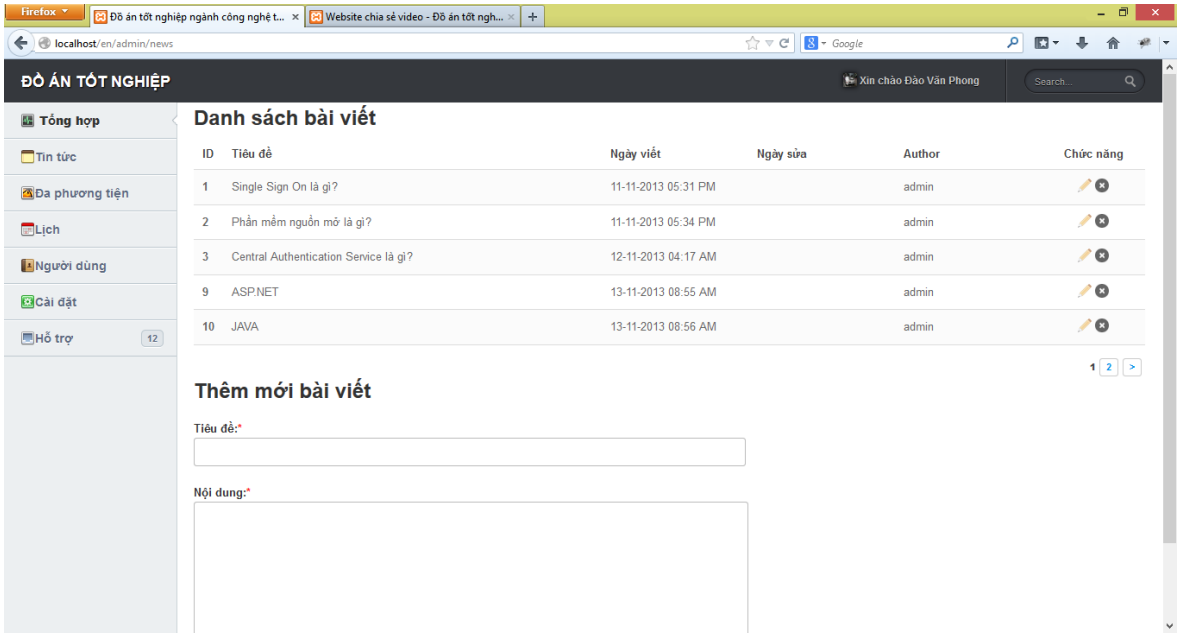
- Đăng ký người dùng
- Kích hoạt tài khoản thông qua email
- Đăng nhập
- Viết bài
- Cập nhật hồ sơ người dùng.



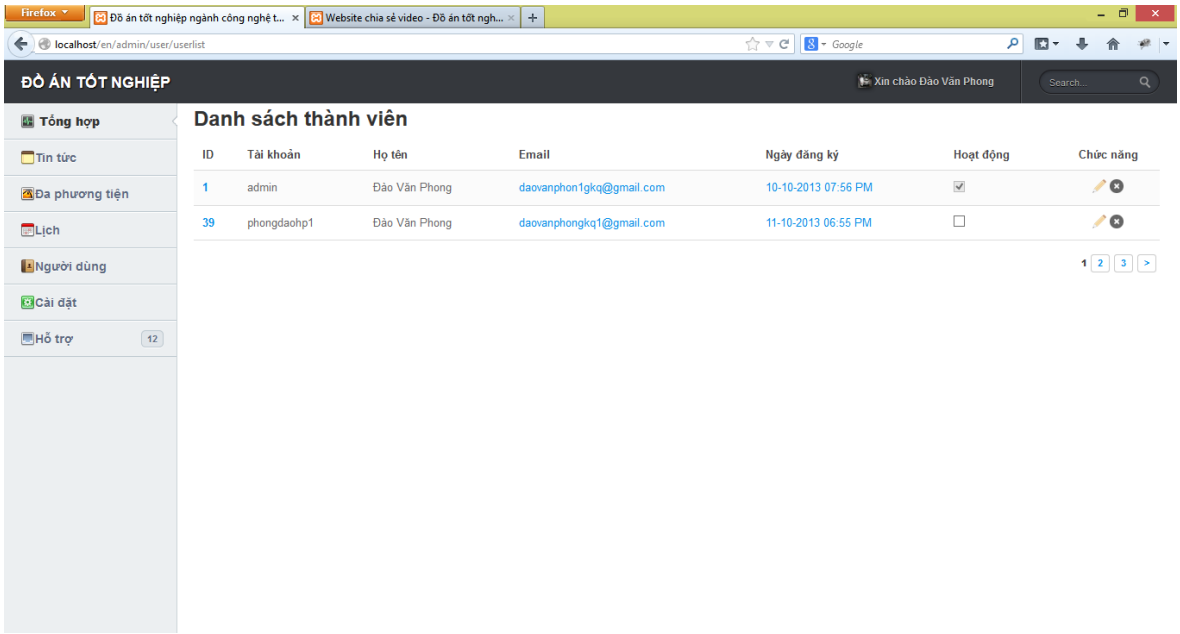
Hình 3.24: Trang đăng ký người dùng website 1.



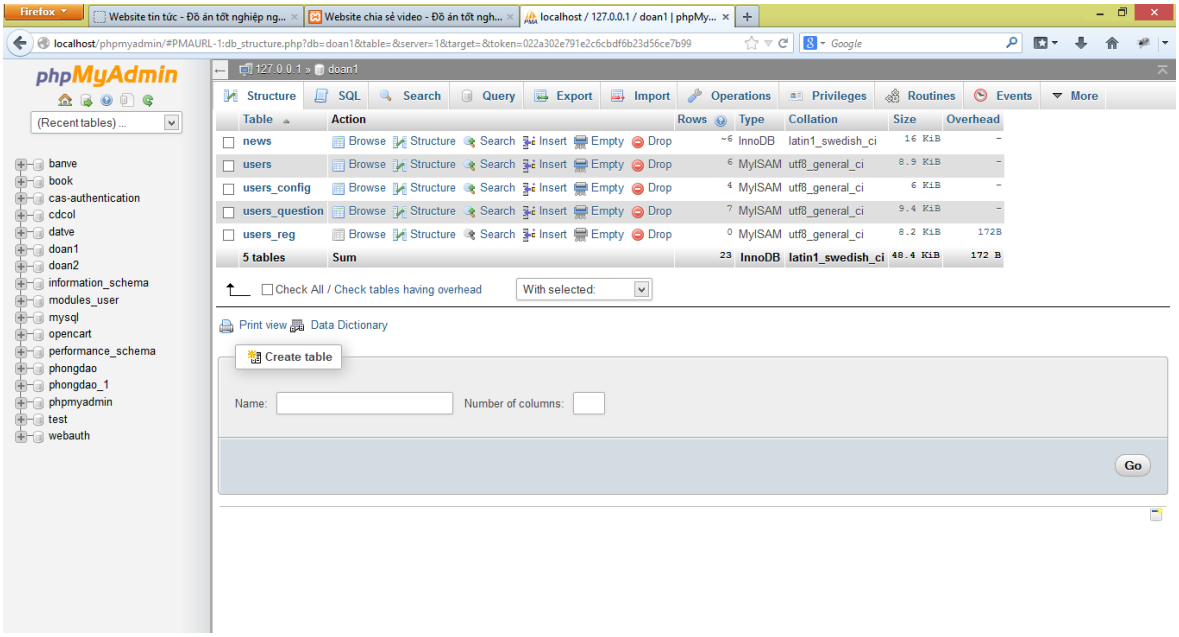
Hình 3.25: Trang đăng nhập hệ thống website 1.



Hình 3.26: Thêm mới bài viết.



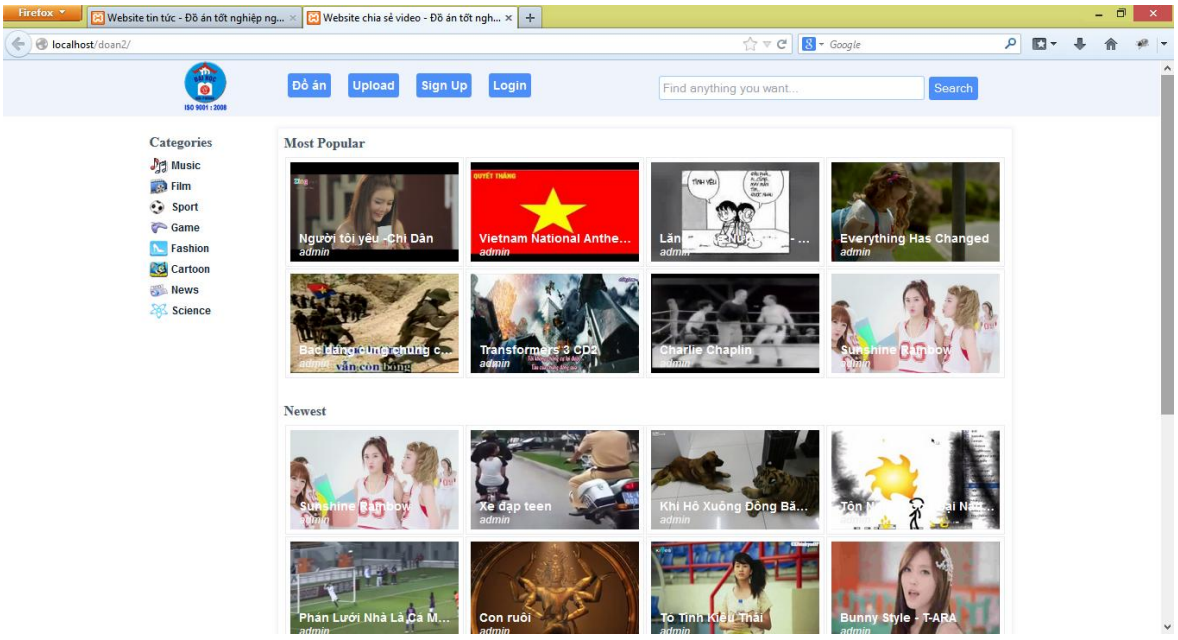
Hình 3.27: Danh sách người dùng.



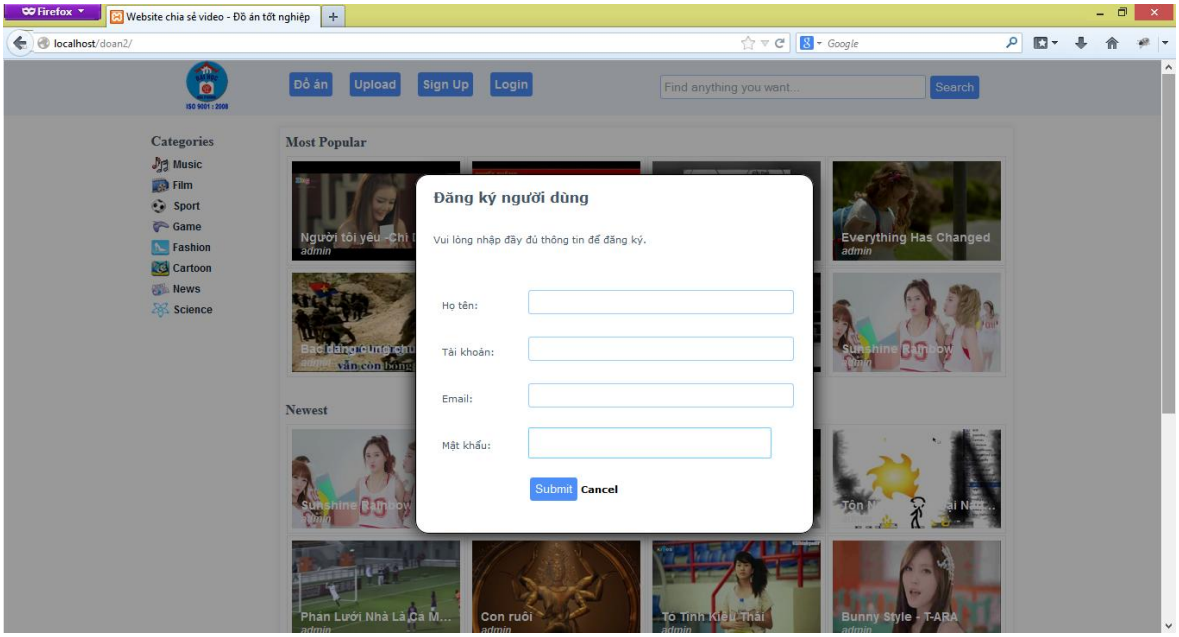
Hình 3.28: Cấu trúc CSDL website 1.

B. Website 2.

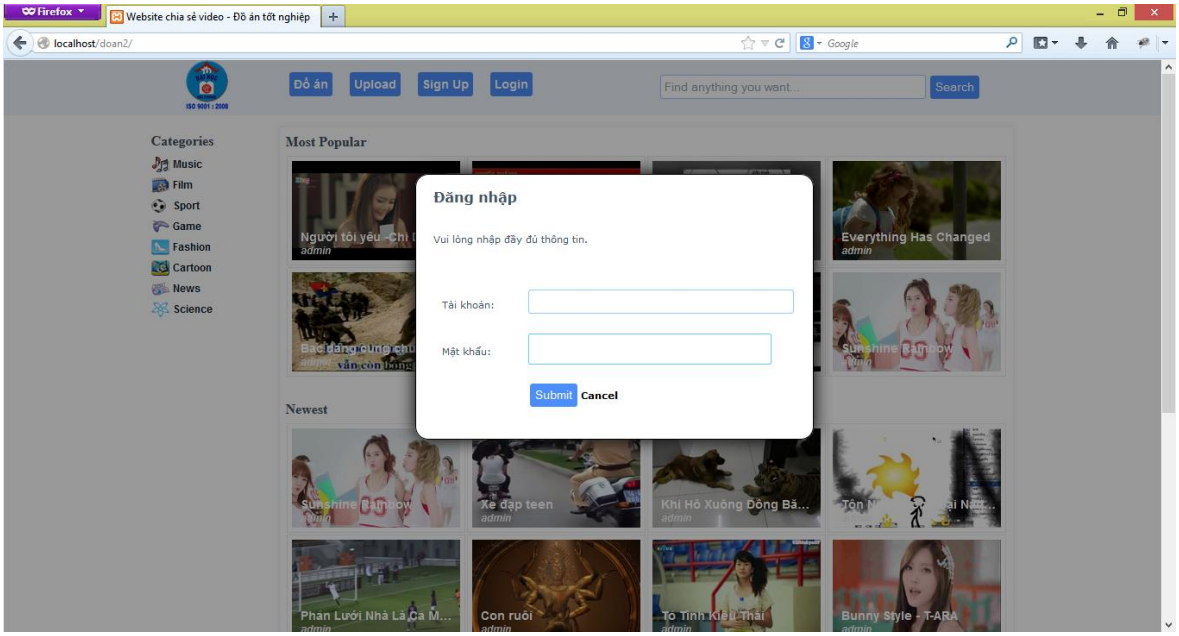
Website là là website cho phép đăng tải và chia sẻ video do Đặng Đức Tuyển – sinh viên khóa 13 xây dựng trong thời gian làm đề tài tốt nghiệp đợt 1. Website cho phép thành viên đăng ký, đăng nhập và đăng tải và chia sẻ video.



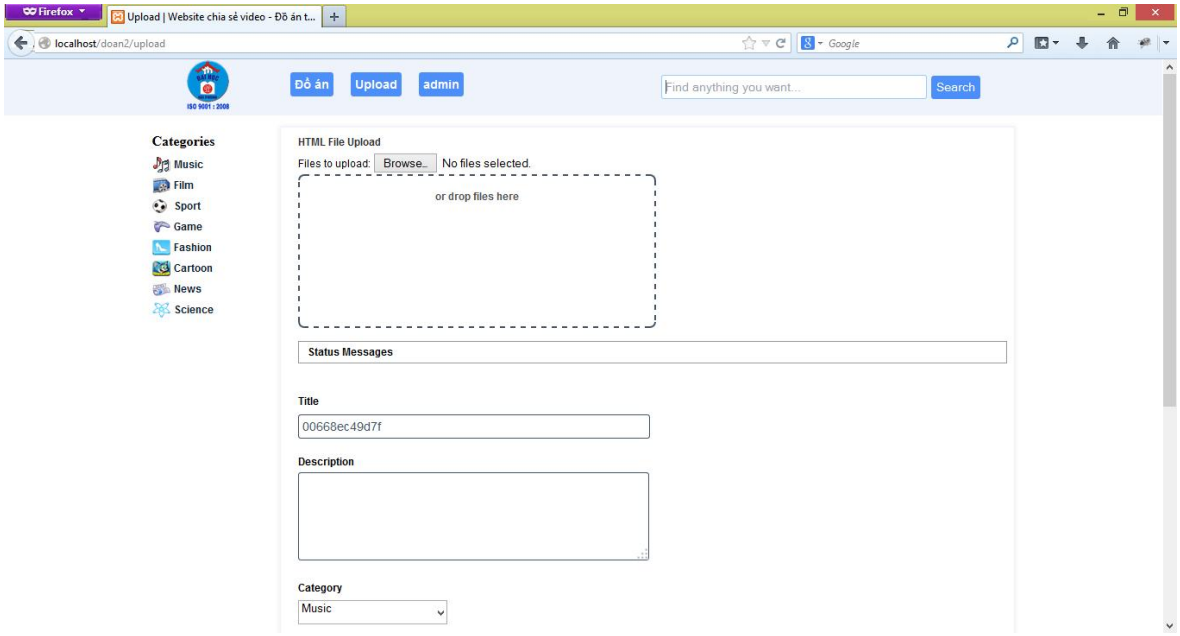
Hình 3.29: Trang chủ website 2.



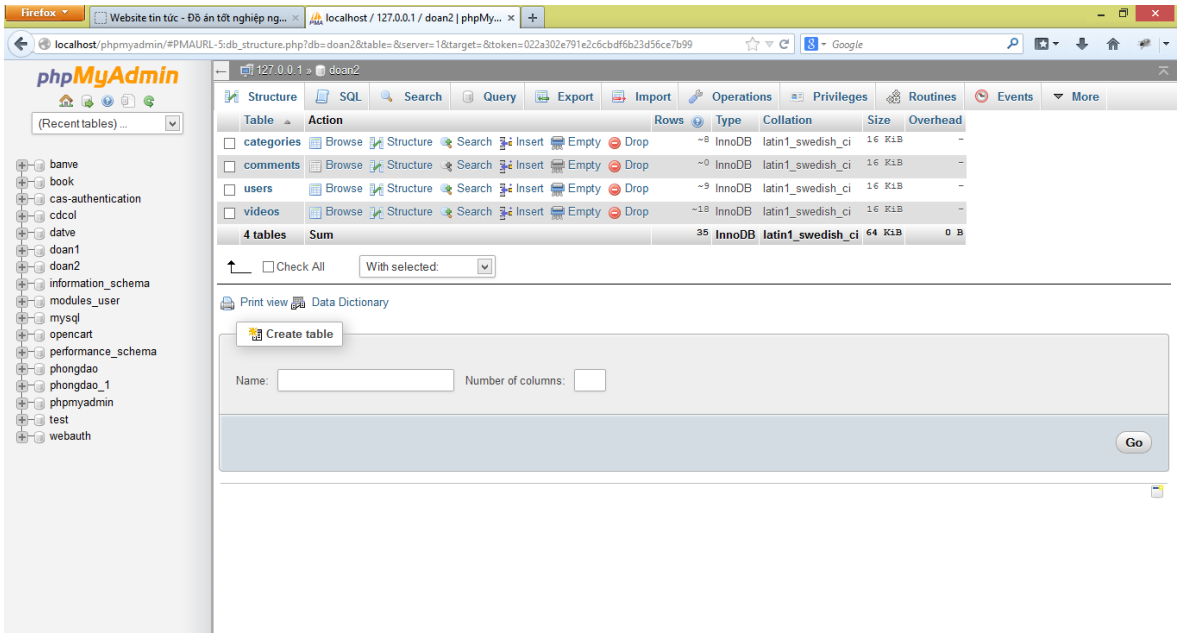
Hình 3.30: Đăng ký người dùng website 2.



Hình 3.31: Đăng nhập hệ thống website 2.



Hình 3.32:Trang upload video website 2.

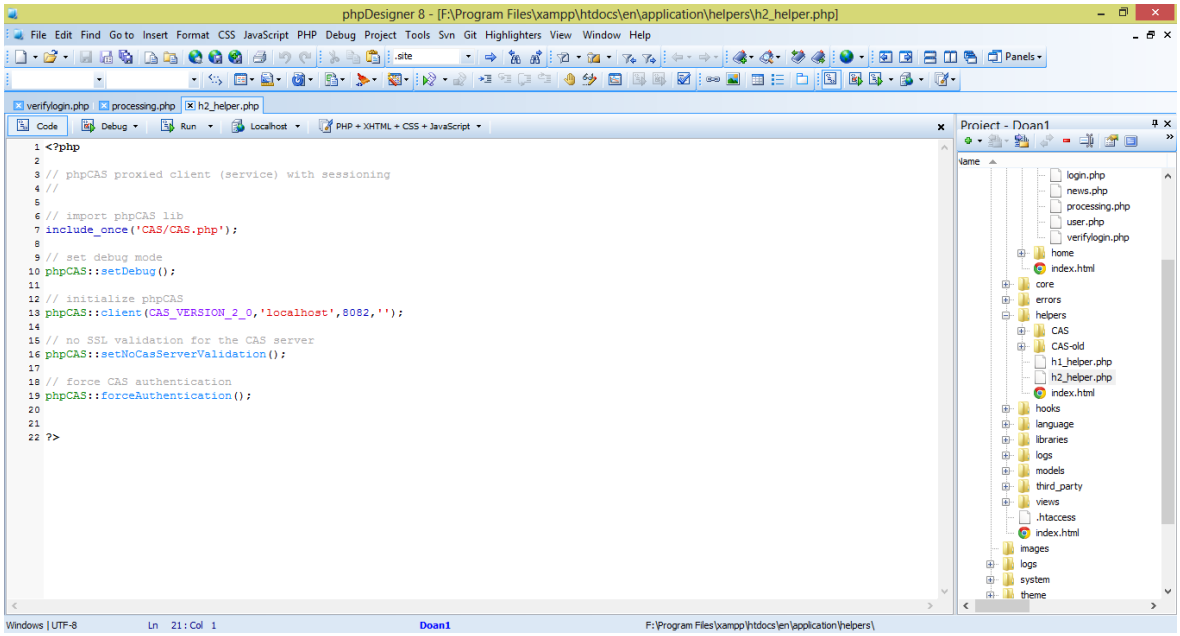


Hình 3.33: Cấu trúc CSDL website 2.

3.1.4.2.Cài đặt phpCAS.

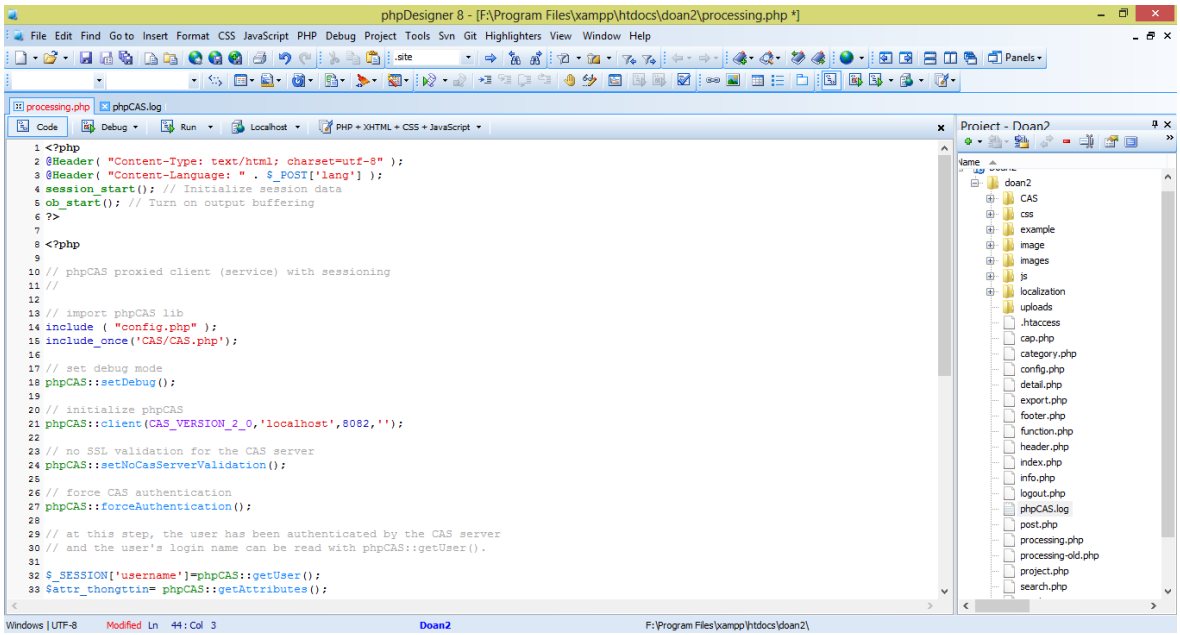
Tải thư viện phpCAS tại địa chỉ : http://downloads.jasig.org/cas-clients/php/current/. Giải nén vào include vào 2 website cần tích hợp. Với trường hợp 2 website tôi đưa ra thì như sau:

A. Website 1: Vui lòng xem phần tích hợp tại phụ lục C.



Hình 3.34: Tích hợp phpCAS vào website 1.

B. Website 2:Vui lòng xem phần tích hợp tại phụ lục D.



Hình 3.35: Tích hợp phpCAS website 2.

3.2. Các pha trong hệ thống khi user đăng nhập.

Pha 1: Usertồn tại trên 2 hệ thống thì quá trình sẽ diễn ra như thế nào?

Đối với trường hợp user tồn tại song song trên 2 hệ thống thì việc xác thực thông tin, CAS server gửi lại cho client username và extra_atttributes như bình

thường. Tại client thì việc xử lý thông tin nhận được từ CAS server diễn ra hoàn toàn bình thường. Xem hình 2.6: Nguyên tắc hoạt động phpCAS.

Pha 2: user chỉ tồn tại trong 1 trong 2 hệ thống thì quá trình diễn ra sẽ như thế nào với từng hệ thống, hệ thống 1 như thế nào?Hệ thống 2 như thế nào?

Trường hợp 1: User chỉ tồn tại trên CAS server.

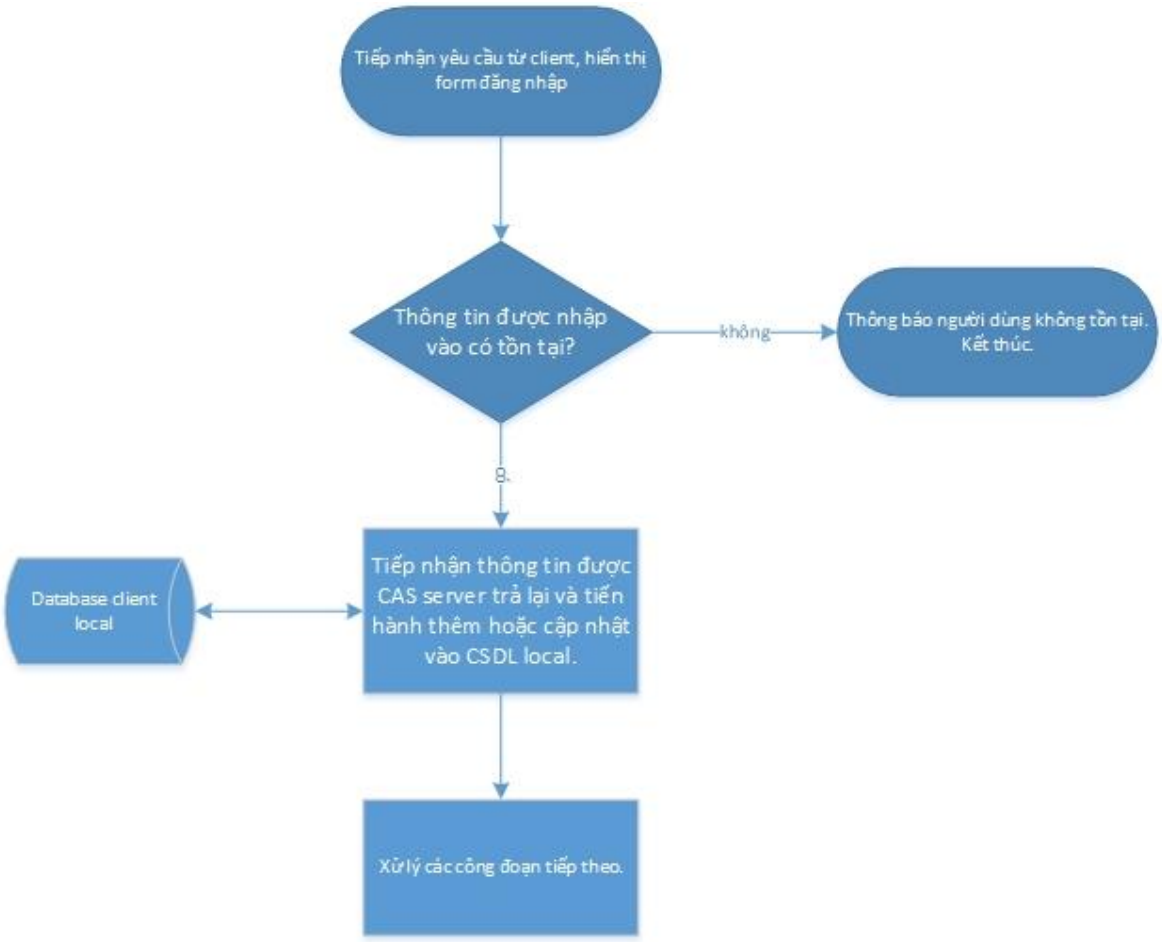
Khi người dùngmuốn xác thực thông tin để sử dụng ứng dụng, phpCAS sẽ chuyển hướng người dùng đến form đăng nhập của CAS server, tại đây người dùng nhập thông tin, CAS xác thực thông tin và trả lại cho client username và extra_Attributes(nếu có). Tại client tùy thuộc vào nhu cầu của ứng dụng mà có sử dụng thông tin nhận được để thêm vào CSDL của ứng dụng client hay không? Với trường hợp hệ thống của em tích hợp có nhu cầu thêm phần thông tin đã nhận được vào CSDL thì các bước xử lý sẽ như sau:

Bước 1: Client sẽ sử dụng username nhận được từ CAS server) làm điều kiện để truy vấn vào CSDL của ứng dụng client.

Bước 2: Kiểm tra kết quả truy vấn vào CSDL thì có 2 trường hợp:

Trường hợp 1: Không tồn tại bản ghi nào theo điều kiện đã đưa vào -> Tiến hành thêm username và các extra_attributes như email, address, status...(không bao gồm password vì lý do an toàn.) vào CSDL. Sau đó việc xử lý thông tin xác thực diễn ra như bình thường.

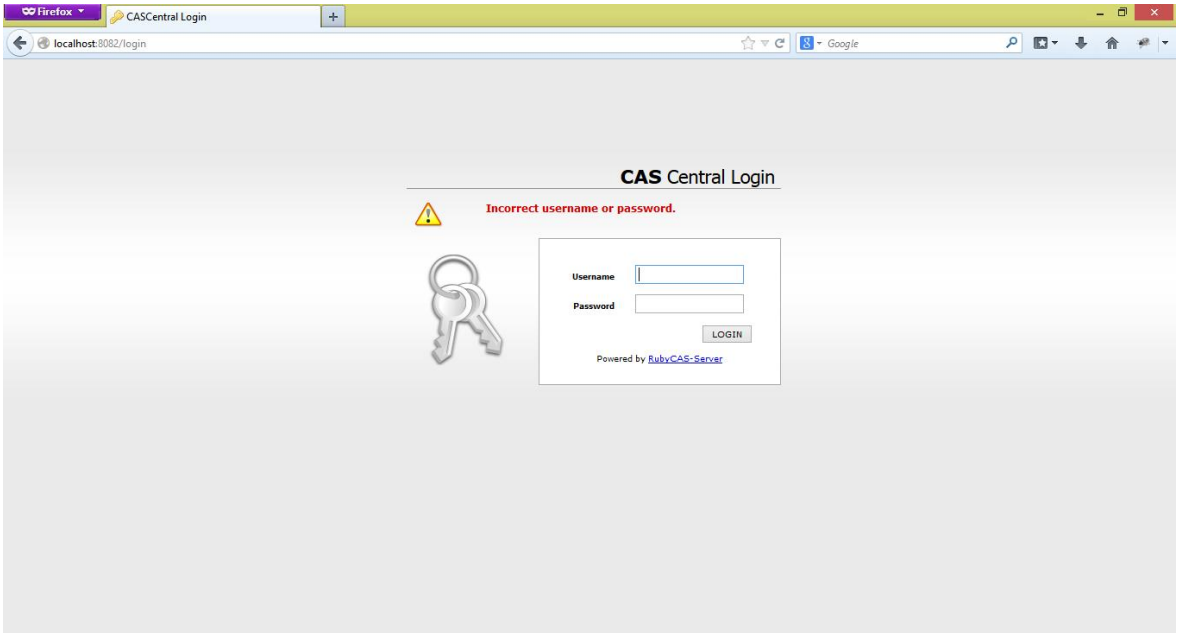
Trường hợp 2: Đã tồn tại bản ghi thì ta lại tiếp tục so sánh các thông tin bản ghi vừa truy vấn với các extra_attributes nếu giống nhau thì bỏ qua và tiến hành xử lý thông tin xác thực, nếu khác nhau thì tiến hành cập nhật lại các thông tin theo extra_attributes. Sau khi cập nhật xong thì lại tiếp tục xử lý thông tin xác thực.



Hình 3.36: Luồng xử lý khi client xin xác thực thông tin từ CAS server.

Trường hợp 2: User chỉ tồn tại trên client.

Với trường hợp này thì việc xác thực thông tin sẽ thất bại vì trong CSDL của CAS server không tồn tại thông tin của người dùng dẫn đến không có thông tin để xác thực.



Hình 3.37: Đăng nhập khi user không tồn tại ở CAS server.

Pha 3: User bị xóa hoàn toàn trên CSDL lưu trữ người dùng trên CAS server, vậy khi đăng nhập vào hệ thống sẽ như thế nào?

Đối với trường hợp này thì thì nó giống với trường hợp 2 của pha 2: User chỉ tồn tại trên client.

Pha 4: User bị no active nghĩa là trước đây đã là thành viên sau một thời gian cần phải tạm thời không cho user ấy đăng nhập sau đó một thời gian lại cho đăng nhập lại (VD: SVtrong trường tại thời điểm thi vì chưa hoàn thành các khoản tiền lên không thể đăng nhập vào hệ thống đó vào xem điểm của mình được. Sau khi hoàn thành các khoản tiền sinh viên lại được đăng nhập lại). Vấn đề nảy hệ thống thống sẽ được giải quyết thế nào?

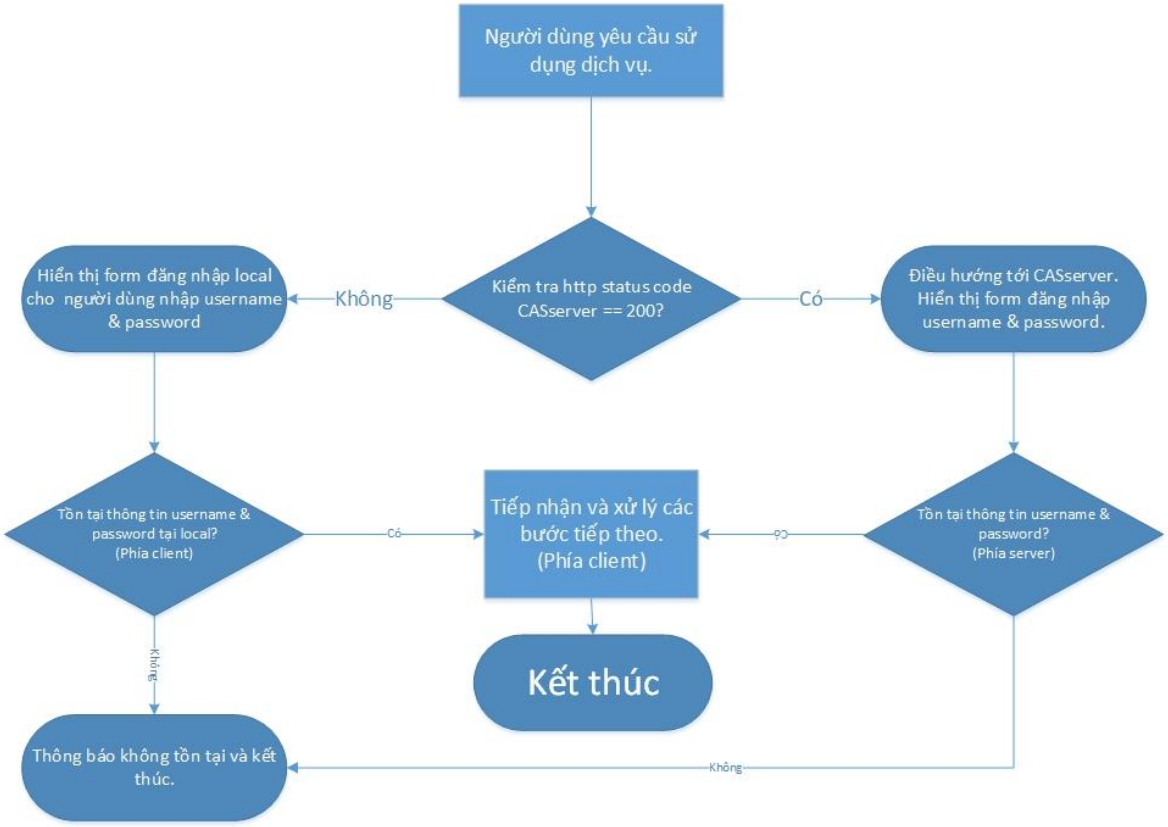
Xin thưa rằng CAS server chỉ có nhiệm vụ lưu trữ thông tin của 1 người dùng nào đó như username, password, email, role… tùy thuộc vào role mà xếp user đó thuộc vào nhóm người dùng nào (active, inactive, locked….) khi client có yêu cầu CAS sẽ trả lại cho client thông tin trong đó có role và tại đây việc xử lý tiếp theo tùy vào role mà triển khai.

Pha 5: Khi user thay đổi thông tin người dùng thì hệ thống sẽ xử lý như thế nào?

Giống như trường hợp 2 của pha 2.

Pha 6: Trường hợp khi CAS server ngừng hoạt động thì việc xác thực sẽ diễn ra như thế nào ?

Trước khi client điều người dùng tới CAS server thì sẽ kiểm tra http Status code do CAS server trả về. Nếu Status Code == 200 hoặc 303 thì điều hướng client đến CAS server còn ngược lại gặp những status code khác thì xác thực thông tin tại CSDL local.



Hình 3.38: Sơ đồ luồng pha 6 .

KẾT LUẬN

Trong đồ án này em tìm hiểu được cơ chế đăng nhập một lần (single sign on) và thử nghiệm dựa trên thư viện phpcas. Đồ án đã thực hiện được nhiệm vụ đề ra và đạt được các kết quả sau:

- Tìm hiểu tổng qua về cơ chế đăng nhập một lần, các thức lưu trữ, truy cập vào CSDL.
- Có thêm kiến thức về hệ thống đăng nhập 1 lần (SSO) và dịch vụ xác thực trung tâm (CAS).
- Triển khai thành công SSO thông qua RubyCAS.
- Tích hợp thành công thư viện phpCAS cho các website PHP.
- Kỹ năng lập trình, kỹ năng tìm hiểu và phân tích được nâng cao.

Trong quá trình tìm hiểu và thực nghiệm hệ thống thì cũng nảy sinh các vấn đề như sau:

- Hầu hết tài liệu được viết bằng tiếng Anh, vì thế trong quá trình tìm hiểu không tránh được sai sót nên mong sự góp ý của thầy cô và các bạn.
- Thời gian bị hạn chế nên chưa thực sự tìm hiểu thật chi tiết về hệ thống.
- Hệ thống SSO hoạt động thông qua cookies nên vấn đề phát sinh từ phía người dùng đó là người dùng vô ý hay cố ý tắt cookies trên trình duyệt nên hệ thống SSO sẽ không hoạt động.
- CAS cung cấp ticket tương ứng với 1 cookie trên trình duyệt : vì vậy nếu 2 người dùng ngồi vào một mày và sử dụng 1 trình duyệt thì không thể đăng nhập được vì không có khái niệm đăng nhập thêm user đó là 1 trong những điểm hạn chế so với các hệ thống đăng nhập tập trung khác như google....

Hướng phát triển sẽ là:

- Giải quyết các vấn đề còn tồn đọng trong quá trình nghiên cứu xây dựng hệ thống.
- Tiếp tục nghiên cứu và xây dựng hệ thống trở lên hoàn thiện hơn.
- Tích hợp hệ thống SSO và các nền tảng, ngôn ngữ khác nhau như .NET, JAVA, RUBY hay các hệ thống đóng.

TÀI LIỆU THAM KHẢO

[1] http://en.wikipedia.org/wiki/Single_sign-on
[2] http://en.wikipedia.org/wiki/List_of_single_sign-on_implementations
[3]http://vi.wikipedia.org/wiki/Phần_mềm_nguồn_mở
[4]http://www.jasig.org/cas/protocol
[5]http://en.wikipedia.org/wiki/Central_Authentication_Service
[6]https://github.com/rubycas/rubycas-server/wiki
[7] https://wiki.jasig.org/display/CASC/phpCAS

PHỤ LỤC

Phụ lục A: CAS phản hồi lược đồ XML.

```
<!--
  The following is the schema for the Yale Central Authentication
Service (CAS) version 2.0 protocol response.  This covers the responses
  for the following servlets:
   /serviceValidate
/proxyValidate
   /proxy
  This specification is subject to change.
  Author:  Drew Mazurek
 Version: $Id: cas2.xsd,v 1.1 2005/02/14 16:19:06 dmazurek Exp $-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:cas="http://www.yale.edu/tp/cas"
        targetNamespace="http://www.yale.edu/tp/cas"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="serviceResponse" type="cas:ServiceResponseType"/>
<xs:complexType name="ServiceResponseType">
<xs:choice>
<xs:element name="authenticationSuccess" type="cas:AuthenticationSuccessType"/>
<xs:element name="authenticationFailure" type="cas:AuthenticationFailureType"/>
<xs:element name="proxySuccess" type="cas:ProxySuccessType"/>
<xs:element name="proxyFailure" type="cas:ProxyFailureType"/>
</xs:choice>
</xs:complexType>
<xs:complexType name="AuthenticationSuccessType">
<xs:sequence>
<xs:element name="user" type="xs:string"/>
```

```
<xs:element name="proxyGrantingTicket" type="xs:string" minOccurs="0"/>
<xs:element name="proxies" type="cas:ProxiesType" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ProxiesType">
<xs:sequence>
<xs:element name="proxy" type="xs:string" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="AuthenticationFailureType">
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="code" type="xs:string" use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="ProxySuccessType">
<xs:sequence>
<xs:element name="proxyTicket" type="xs:string"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ProxyFailureType">
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="code" type="xs:string" use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
```

</xs:schema>

Phụ lục B: Chuyển hướng an toàn.

Sau khi đăng nhập thành công, chuyển hướng một cách an toàn cho client từ CAS đến đích cuối cùng của nó phải được xử lý cẩn thận. Trong hầu hết các trường hợp, client đã gửi thông tin đến máy chủ CAS trên một yêu cầu POST. Trong đặc tả này, máy chủ CAS sau đó phải chuyển người dùng đến các ứng dụng với một yêu cầu GET.

Các HTTP/1.1 cung cấp một mã phản hồi 303: Bên cạnh đó, nó cung cấp cho các hành vi mong muốn: một kịch bản tiếp nhận dữ liệu thông qua một yêu cầu POST, thông qua 303 redirection, chuyển tiếp trình duyệt đến một URL khác thông qua một GET request. Tuy nhiên, không phải tất cả các trình duyệt đã thực hiện hành vi này một cách chính xác.

Các phương pháp khuyến cáo chuyển hướng là dùng JavaScript. Một trang có chứa một window.location.href theo cách sau đây thực hiện đầy đủ:

<html>

<head>

<title>Yale Central Authentication Service</title>

<script> window.location.href="https://portal.yale.edu/Login?ticket=ST-..." mce_href="https://portal.yale.edu/Login?ticket=ST-...";

</script>

</head>

<body>

<noscript>

* CAS login successful. *

* Click here*

*to access the service you requested.
 *

</noscript>

</body>

</html>

Phụ Lục C: Phần code xử lý đăng nhập SSO hệ thống 1.

Phần xử lý đăng nhập trước khi tích hợp phpCAS.

```
<?php if (!defined('BASEPATH'))
   exit('No direct script access allowed');
class VerifyLogin extends CI_Controller
{
   function __construct()
   {
      parent::__construct();
      $this->load->library("form_validation");
   }
   public function index()
   {
      if ($this->my_auth->is_Login()) {
         redirect(base_url(). "admin/home");
         exit();
      }

      $this->form_validation->set_rules('username', 'Username',
         'trim|required|xss_clean');
      $this->form_validation->set_rules('password', 'Password',
         'trim|required|xss_clean');
      if ($this->form_validation->run() == false) {

         //Xac nhan that bai. Nguoi dung bi dieu huong toi trang dang nhap
         $this->load->view('admin/login');

      } else {

         $array = array('username' => $this->input->post('username'), 'password'
=> md5($this->
               input->post('password')));
```

```
                $result = $this->muser->checkLogin($array);

                if ($result) {
                    if (!$this->my_auth->is_Active($result['userid'])) {
                        $data['error'] = "Tài khoản chưa được kích hoạt !";
                        $this->load->view('admin/login', $data);

                    } else {

                        $data = array(
                            "username" => $result['username'],
                            "userid" => $result['userid'],
                            "permission" => $result['permission'],
                            );
                        $this->session->set_userdata('logged_in', $data);
                        redirect(base_url(). "admin/home");
                    }
                } else {
                    $this->load->view('admin/login', array("error" => "Username hoặc
Password sai"));
                }

            }
        }
    }
    ?>
```

Sau khi tích hợp:

```
    <?php if (!defined('BASEPATH'))
        exit('No direct script access allowed');
    class Processing extends CI_Controller
    {
        function __construct()
        {
            parent::__construct();
```

```
    $this->load->helper('h2');
}
public function index()
{

    if ($this->my_auth->is_Login()) {
        redirect(base_url(). "admin/home");
        exit();
    }


    $result = phpCAS::getAttributes();
    $data = array(
        "username" => $result['username'],
        "full_name" => $result['full_name'],
        "permission" => $result['permission'],
        );

    if ($result) {

        if ($this->muser->getInfo1($data['username']) != false) {

            if (!$this->my_auth->is_Active($result['username'])) {
                $data['error'] = "Tài khoản chưa được kích hoạt !";
                $this->load->view('admin/login', $data);

            } else {

                $this->session->set_userdata('logged_in', $data);
                redirect(base_url(). "admin/home");
            }
        } else {
            $this->muser->AddNewUser1($data);
```

```
                $this->session->set_userdata('logged_in', $data);
                redirect(base_url(). "admin/home");
            }
        } else {
            $this->load->view('admin/login', array("error" => "Username hoặc Password sai"));
        }
    }
}
?>
```

Trong đó:

$this->load->helper('h2') – load phần helper đã tích hợp phpCAS

Phụ Lục D: Phần code xử lý đăng nhập SSO hệ thống 2.

Phần xử lý trước khi tích hợp phpCAS

```
<?php
@Header( "Content-Type: text/html; charset=utf-8" );
@Header( "Content-Language: ". $_POST['lang'] );
session_start();

include ( "config.php" );

if( isset( $_POST['dangky'] ) )
{
  $username = $_POST['username'];
  $password = $_POST['pass'];
  $email = $_POST['email'];
  $fullname = $_POST['fullname'];
  // Kiểm tra tồn tại của user và email.
  $sql = "select * from users where username='". $username. "'";

  $query = mysql_query( $sql );
  if( mysql_num_rows( $query ) == 0 )
  {
```

```
            $sql = "select * from users where email='". $email. "'";
            $query = mysql_query( $sql );
            if( mysql_num_rows( $query ) == 0 )
            {
                  $sql = "INSERT INTO users (fullname, username, email, password,
permission)
      VALUES ('". $fullname. "','". $username. "','". $email. "','". $password. "','1')";
                  $query = mysql_query( $sql );
                  if( $query )
                  {
                        $_SESSION['username'] = $username;
                        $_SESSION['pass'] = $password;
                        echo '<script>alert("Đăng ký thành công. Bạn có thể tiếp
tục.")</script>';
                        echo
'<script>window.location.assign("http://localhost/doan2/")</script>';
                  }
            }
            else
            {
                  echo '<script>alert("Email đã tồn tại, vui lòng dùng email
khác.")</script>';
                  echo
'<script>window.location.assign("http://localhost/doan2/")</script>';
            }
      }
      else
      {
            echo '<script>alert("Tài khoản đã tồn tại, vui lòng dùng tài khoản
khác.")</script>';
            echo '<script>window.location.assign("http://localhost/doan2/")</script>';
      }
}
else
      if( isset( $_POST['dangnhap'] ) )
      {
```

```
$username = $_POST['username'];
$password = $_POST['pass'];
$sql = "select * from users where username='". $username. "' and password = '". $password. "'";
$query = mysql_query( $sql );
if( mysql_num_rows( $query ) != 0 )
{
    $_SESSION['username'] = $username;
    $_SESSION['pass'] = $password;
    echo '<script>alert("Đăng nhập thành công. Nhấn ok để trở về trang chủ.")</script>';
    echo '<script>window.location.assign("http://localhost/doan2/")</script>';
}
else
{
    echo '<script>alert("Tài khoản hoặc mật khẩu không chính xác, vui lòng kiểm tra lại.")</script>';
    echo '<script>window.location.assign("http://localhost/doan2/")</script>';
}
}
else
{
    echo '<script>alert("Có lỗi xảy ra. Vui lòng liên lạc tới người quản trị.")</script>';
    echo '<script>window.location.assign("http://localhost/doan2/")</script>';
}

?>
```

Sau khi tích hợp phpCAS.

```
<?php
@Header( "Content-Type: text/html; charset=utf-8" );
@Header( "Content-Language: ". $_POST['lang'] );
session_start(); // Initialize session data
```

```
ob_start(); // Turn on output buffering
?>

<?php

// phpCAS proxied client (service) with sessioning
//

// import phpCAS lib
include ( "config.php" );
include_once('CAS/CAS.php');

// set debug mode
phpCAS::setDebug();

// initialize phpCAS
phpCAS::client(CAS_VERSION_2_0,'localhost',8082,'');

// no SSL validation for the CAS server
phpCAS::setNoCasServerValidation();

// force CAS authentication
phpCAS::forceAuthentication();

// at this step, the user has been authenticated by the CAS server
// and the user's login name can be read with phpCAS::getUser().

$_SESSION['username']=phpCAS::getUser();
$attr_thongttin= phpCAS::getAttributes();
$fullname =$attr_thongttin['full_name'];
$email = "daovanphongkq@gmail.com";
$password ="123456";

if( isset( $_SESSION['username'] ) )
```

```
{
   // Kiểm tra tồn tại của user
        $sql = "select * from users where username='". $_SESSION['username']. "'";
        $query = mysql_query( $sql );
        if( mysql_num_rows( $query ) != 0 )
        {
          // echo '<script>alert("Đăng nhập thành công. Bạn có thể tiếp tục.")</script>';
      echo '<script>window.location.assign("http://localhost/doan2/")</script>';
        }else
   {
      $sql = "INSERT INTO users (fullname, username, email, password, permission)
      VALUES ('". $fullname. "','". $_SESSION['username']. "','". $email. "','". $password. "','1')";
      $query = mysql_query( $sql );
      if( $query )
                {
                        echo '<script>alert("Đăng nhập. Bạn có thể tiếp tục.")</script>';
                        echo '<script>window.location.assign("http://localhost/doan2/")</script>';
                }else{
                  echo '<script>alert("Có lỗi. Vui lòng kiểm tra lại hệ thống.")</script>';
                        echo '<script>window.location.assign("http://localhost/doan2/")</script>';
                }
   }
}
?>
```