Proyecto del tercer parcial para Programación Web 2017A.
Profesor: José Figueroa Martínez
Fecha: 07/12/2016

El tercer parcial se va a calificar con un proyecto de desarrollo.
Se trata de hacer un sistema independiente de autenticación y roles, así como un sistema manejador de contenidos orientado a consultas SQL y visualizaciones de consultas SQL.

JUSTIFICACIÓN

Primer sistema: **Autenticador**
En varias instituciones se van desarrollando poco a poco sistemas pequeños para resolver necesidades inmediatas. Cada uno de esos sistemas maneja su propia base de datos y sus propios usuarios. Progresivamente se van desarrollando más sistemas y cada uno sigue desarrollándose con sus propios usuarios.
Con el paso del tiempo un usuario normal puede tener diferentes roles en la institución y tener que usar varios de esos sistemas individuales, teniendo que mantener su información en cada uno de ellos por separado. Cambiar un password significa cambiarlo en todos los servicios que utiliza. Actualizar sus roles significa cambiarlos en varios sistemas. Al final de cuentas, un usuario puede acceder a varios sistemas de su institución.
La solución planteada en el primer sistema es una manera de mantener la información de autenticación y los roles del usuario en un solo sistema cuyo propósito sea solo ese.
Con un sistema de autenticación interna se pueden desarrollar otros sistemas que utilicen dicho servicio de autenticación para no tener que guardar datos personales o contraseñas.

Segundo sistema: **Consultor**
El otro sistema que se propone realizar es un manejador de consultas y visualizaciones.
Un sistema evoluciona y cambia con el tiempo. De la misma manera sus datos históricos aumentan y tal vez sus necesidades de datos aumenten por nuevas normativas o por cambios en la lógica del negocio.
Lo más normal es que se requieran reportes que vayan adaptándose también a los cambios en cuestión de cantidad y formas de los datos. Pero no solamente eso. Se requerirán diferentes consultas con diversas formas de visualización, desde reportes hasta varios tipos de *charts* o gráficas.
La manera tradicional es que un grupo de desarrolladores implementen las consultas directamente en el sistema. Luego otro grupo implementa las visualizaciones correspondientes. Sin embargo, lo único realmente requerido es tener acceso (de solo lectura) a la base de datos.
Se propone la realización de un sistema que pueda conectarse a cualquier manejador de bases de datos, que permita crear consultas desde la propia aplicación y que genere una api basada en servicios para acceder a los datos devueltos por la consulta.
De la misma manera, dicho sistema debe proveer los medios para crear visualizaciones que accedan a sus propios servicios de datos y que se creen y modifiquen a través de la aplicación. Como un CMS (Content Management System) pero para consultas y visualizaciones. Una aplicación así es de mucho valor para cualquier institución o empresa, ya que libera a los analístas y genentes o usuarios normales de la necesidad de tocar el código de su aplicación, si es que lo tienen o lo entienden.

En el marco de la asignatura de Programación Web, se buscará que dichos sistemas les planteen las dificultades prácticas más comunes y necesarias a la hora de desarrollar aplicaciones web, de manera que cubran la mayoría del conocimiento requerido por un desarrollador web en el *backend*.

Autenticador

Entidades del sistema

La entidad **aservice** representa un servicio en el cual se puede autenticar un usuario. Por ejemplo, un servidor ftp.

aservice:
- id
- name (unique)
- ip
- protocol

La entidad **service** representa un servicio o sistema que utiliza al servidor de autenticación. De esta manera, un usuario pued estar ligado a un servicio de autenticación para tener una contraseña específica para dicho servicio.

service:
- id
- name

La entidad **user** representa a los posibles usuarios globales en la institución. Los datos de identificación son básicos. El *gpassword* es un password global para cualquier servicio y se usa siempre y cuando no halla un password para el servicio sobre el cual se quiere autenticar con login. Se guarda aplicando al password real la función sha1. El atributo *gaservice_id* representa un servicio de autenticación global para el usuario.

user:
- id
- first_name text
- last_name text
- email varchar(100)
- gpassword varchar(100)
- gaservice_id
- registered_at - Date

La entidad **userservice** liga usuarios y servicios de autenticación. Esta entidad le permite a un usuario tener formas individuales para autenticarse en diferentes servicios. Por ejemplo, yo puedo tener una contraseña o servicio de autenticación global, pero quiero usar una contraseña especial para un servicio particular. Esta entidad me permite eso y sobre escribe la forma de autenticarse en éste servicio particular. No debe haber más de una tupla para un usuario y un mismo servicio, pues habría ambigüedad.

useraservice:
- id
- user_id (usuario)
- service_id (servicio para el que quiere este passwd/aservice)
- aservice_id (para autenticar su password de este service_id)
- username (solo en caso de tener un aservice_id)
- password (contraseña para este servicio)
- hint (tip para recordar la contraseña)

La entidad **rol** representa los tipos de roles o cargos que un usuario puede tener o desempeñar en la organización. El código es la representación *programática* del rol. Por ejemplo, el rol con nombre “Jefe de Recursos Escolares” puede tener el código “JEFE_ESCOLARES”. Son como identificadores para dichos roles que pueden utilizar otros sistemas. Estos códigos no deben cambiar.

rol:
- id
- name
- code

La entidad **userrol** representa los roles que tiene o ha tenido el usuario. Los roles cuya fecha de finalización son NULL son actuales.

userrol:

id	
rol_id	
user_id	
started_by	(usuario que autorizó el inicio)
finished_by	(usuario que autorizó la finalización)
started_at	(fecha de inicio)
finished_at	(fecha de finalización)
finished_reason	(razón para terminarlo)
comment	(comentarios)

La entidad **session** representa sesiones que sirven para que los usuarios registrados puedan cambiar o crear sus contraseñas, o para que los administradores puedan usar el sistema para asignar roles o hacer tareas de mantenimiento.

session:

id		
sid	varchar(100)	Session ID (único). Cadena generada al azar o UUID.
email	varchar(100)	Usuario ligado a la sesión.
rols	string	Roles separados por espacios.
created_at	datetime	
closed_at	datetime	Al principio es nulo.
duration	int	En minutos.
data	string	JSON u otro formato de serialización de datos si se requiere.

Rutas

Autenticación básica
POST /auth
- Params: email, password, service (nombre)
- Retorna un “true” o “false”
- * El nombre puede ser auth.php o auth.X

Mezcla entre autenticación y solicitud de datos del usuario:
POST /login

Params: email, password, service

Retorna un JSON con los datos del usuario o “null” en caso de no coincidir los passwords:
{Iid”: X, “email”: Y, “first_name”: Z, … “rols”: [“JEFE_ESCOLARES”]}

* Puede ser *login.php o login.X*

Obtener los datos de un usuario por su email.
GET /user?email=X

Retorna un JSON con los datos del usuario:
{Iid”: X, “email”: Y, “first_name”: Z, … “rols”: [“JEFE_ESCOLARES”]}

Obtener una lista de todos los usuarios registrados:
GET /user

Obtener todos los roles existentes:
GET /rol

Retorna un JSON con los roles:
[{“id”: X, “name”: “Jefe de Escolares”, “code”: “JEFE_ESCOLARES”}]

Obtener todos los servicios registrados:
GET /service

Retorna un JSON con los servicios registrados.

Roles

Los roles para el sistema son los siguientes: **user**, **admin**. Los dos requieren autenticarse en el sistema y manejar una sesión para sus tareas posibles. Las apis de arriba son públicas.

Un user puede hacer login, registrarse, crear una contraseña propia o cambiarse a un servidor de autenticación, o puede crear una contraseña o registrar una configuración de autenticación para un servicio específico.
Un admin puede crear, modificar y borrar usuarios, roles, servicios y servicios de autenticación. También puede asignar y revocar roles a los usuarios.
Se espera que el administrador tenga un rol propio que lo identifique. Esto se asigna inicialmente directamente en la base de datos.
Los passwords se guardan aplicándoles la función hash *sha1*.
Todo lo que hace el admin y el usuario debe loguearse en archivos.

Consultor

Entidades de datos

La entidad **db** representa una base de datos a la cual conectarse.

db:

name		
description		
user		
password		
dbname		
dsn		
hostsocket		(host o socket)
classname		(clase del driver: JDBC)
subprotocol		(JDBC: mysql, pgsql)
params	string	

La entidad **dbaccess** representa los usuarios o roles que pueden acceder a la *db*. Si no hay ninguno para una *db*, significa que es de acceso libre.

dbaccess

id		
db_id		
email	varchar(100)	Un email (usuario) o rol.
rol		

La entidad **query** representa una consulta en SQL, la cual tiene una categoría (para mostrarse en ella), nombre único, descripción, base de datos o conexión a la que pertenece, contenido SQL, estado de error (para apoyar al debugging, vacía sin error), tipos de datos que devuelve separados por espacios, email de quien mantiene dicho query y fechahora de creación.
Los ptypes pueden ser por ahora: int, string, date, float, bigint, bigdec, json

query:

id		
category		
name (unique)		
description		Debe describir también los parámetros que usa.
db_id		
sql		
error		
ptypes		
email	varchar(100)	
created_at		

La entidad **view** representa una vista SQL de la cual depende un *query.* Dicha vista tiene un nombre propio y único, un enlace al *query* al que pertenece, un contenido SQL, un error de creación (se debe borrar la vieja view y crear en cada cambio) y una fechahora de creación.

Una vista se crea en la base de datos cuando se guarda o edita. Si hay errores se rellena el atributo *error.*

view:

- id
- name (unique)
- query_id
- sql
- error
- created_at

La entidad **ui** representa una interfaz de visualización de uno o varios datos.

ui:

id	
name	(unico)
description	
index	Nombre del documento principal de la UI o índice.
email	Usuario que creo la UI.
ekey	Clave de edición de los documentos de la ui. Se utiliza para actualizar el content de cualquier document de esta ui subiendo el archivo.
created_at	Fechahora de creación.

document:

id	
ui_id	
name	(unico en la UI)
mime	
description	
content	
created_at	Fechahora de creación
updated_at	Fechahora de última actualización (cada vez que se guarda o sube)

La entidad **session** representa una sesión actual y tal vez datos guardados en ella.

session:

id		
sid	varchar(100)	Session ID (único). Cadena generada al azar o UUID.
email	varchar(100)	Usuario ligado a la sesión.
rols	string	Roles separados por espacios.
created_at	datetime	
closed_at	datetime	Al principio es nulo.
duration	int	En minutos.
data	string	JSON u otro formato de serialización de datos si se requiere.

Rutas básicas

1. listo
Ejecución de las consultas:
GET /query/:nombre_query?sid=SID&param1=X&param2=Y
* El sid es el id de sesión y es opcional, ya que hay consultas de acceso público.
* Si la consulta requiere permisos y no se proporciona un SID con los permisos adecuados en cuestión de roles, se devolverá un arreglo vación en json y el status 403.

2. listo
Redirección al index o principal de una UI:
GET /ui/:nombre/?sid=SID

3. listo
Ejecución de un documento individual
GET /ui/:nombre_ui/:nombre_doc?sid=SID

4. listo
Subir documents de forma externa
POST /ui/:nombre_ui/:nombre_document
Params: content (string), (string)

5.listo
Lista de conexiones
GET /db??sid=SID
* Solo se listan las conecciónes que el usuario pueda ver. Las públicas las pueden ver todos.
* No se muestran datos de conexión como host, socket, user o password.

6. listo
Lista de consultas
GET /query?sid=SID
* Se manda el sid porque algunas consultas no se deben de poder visualizar públicamente.

7. listo
Lista de vistas por query:
GET /view?query=X&sid=SID

8. listo
Configuraciones de la aplicación (url base propia, etc.)
GET /config
Retorna un objeto JSON con las siguientes configuraciones:
server_url La url del servicio: http://localhost:3000/consultor1/
Se utiliza como base para otros servicios.

Escenario

Imagina que creas una visualización para crear la gráfica de barras de los profesores por instituto en la UTM. Tu *ui* tiene la siguiente URL virtual: **/ui/profesxinstituto**
La *ui* tendría por *index* a “index.htm”, el cual es un *document* dado de alta en la base de datos para dicha *ui*.
Cada componente podría ser de la siguiente manera:

/ui/profesxinstituto/index.htm?sid=99800b85d3383e3a2fb45eb7d0066a4879a9dad0
El nombre del *document* es “index.htm” y es el principal en la *ui.* El código dentro de él puede incluir sus archivos directamente con su nombre pero no puede utilizar recursos en subcarpetas, ya que no hay subcarpetas en realidad, ya que son recursos generados dinámicamente a partir de lo que hay en la base de datos.
Su mime es *text/html.*
Puede acceder al query en la siguiente dirección de ejemplo:
../../query/201412-profesoresxcarrera?anio=2016
Su id de sesión está expresada en la variable GET *sid*. No siempre se requiere tener sesión para acceder a una consulta.
Se usa un *sid* para poder construir las llamadas internas a los *queries* correspondientes, los cuales pueden requerir llevar el parámetro *sid* como parte de sus parámetros GET.

/ui/profesxinstituto/charts.js?sid=99800b85d3383e3a2fb45eb7d0066a4879a9dad0
Este *document* contiene el código fuente de la librería y es de tipo *text/javascript* y se debe de editar con dicho coloreo de código. Este documento es público realmente y no le afecta si hay un sid o no.

/ui/profesxinstituto/estilos.css?sid=99800b85d3383e3a2fb45eb7d0066a4879a9dad0
Este *document* contiene estilos y se debe editar con la sintaxis coloreada para estilos. Es de tipo *text/css*. Este documento es público y no le afecta si hay un *sid* o no.

La consulta a la que acceden se encuentra en la siguiente URL:
/query/201602-profesxinstituto
Sin embargo, si es que requiere permisos para acceder a ella, se pondría así:
/query/201602-profesxinstituto?sid=99800b85d3383e3a2fb45eb7d0066a4879a9dad0

Recuerda que los paths en las urls son relativos al *server_url.* Osea, la url completa podría ser:
http://localhost:3000/consultor1/query/201602-profesxinstituto?sid=99800b85d3383e3a2fb45eb7d0066a4879a9dad0

Configuración

Todas las aplicaciones requieren algún archivo de configuración en donde se encuentre su url base (*server_url*), sus contraseñas para la base de datos, entre otras configuraciones propias del entorno de ejecución en el que se encuentran. Tu puedes tener una instalación de tu app en un servidor con ciertas configuraciones y otra en producción con configuraciones diferentes.