Diario di lavoro

Luogo	Canobbio
Data	10.12.2019

Lavori svolti

Durante questo week-end ho lavorato un po' al progetto iniziando a sviluppare il sistema di invio email per l'invio della password all'indirizzo mail di un utente creato. Oltre a questo lavoretto ho fatto un po' di *bug-fix* ed ottimizzazione del sistema.

Ieri a scuola, testando il progetto durante la pausa, mi sono accorto che c'erano dei problemi a livello del calendario: non riusciva a caricare i dati degli utenti di LDAP. Questo perchè, anche se l'utente era loggato con LDAP, cercavo i suoi dati all'interno del database locale. Il problema sembra banale ma mi ha portato a modificare tutta la struttura del sistema di login e del sistema dedicato al caricamento dei permessi dal database.

Ho avuto la necessità di creare un nuovo oggetto chiamato LdapUser, il quale salva al suo interno il nome, il cognome, lo username e l'email di un utente di LDAP al suo login. Quando invece un utente normale esegue un login viene creato un oggetto di tipo User (che non è altro un estensione di LdapUser che aggiunge campi supplementari). Entrambi gli oggetti vengono creati all'index "*user*" dell'array di sessione: $_SESSION["*user*"].
Le informazioni contenute nell'oggetto LdapUser vengono lette direttamente tramite LDAP in questo modo:

```
// Leggo la risposta della query LDAP e la converto in un array
$entries = ldap_get_entries($ldap, $result);
// Leggo le informazioni
$data = $entries[0];
$firstname = $data["givenname"][0];
$surname = $data["sn"][0];
$email = $data["mail"][0];

 // Costruisco l'oggetto LdapUser
$user = new LdapUser(
    $this->username,
    $firstname,
    $surname,
  $email
);
```

Questo oggetto mi permette di accedere alle informazioni dell'utente senza dover eseguire query nel database locale (nel caso fosse un utente locale) o nel database di Active Directory scolastico (nel caso fosse un utente LDAP).

Dopo aver risolto questo ho testato il sistema sia con il mio utente personale LDAP (luca.dibello) sia con quello admin (luca.dibello): i permessi vengono caricati correttamente ed il sistema risulta funzionante.

Dopo aver fatto questo ho iniziato a sviluppare un applicativo in *Python* per il raspberry. Quando viene fatto partire esso carica il file config.json (il quale contiene le impostazioni del applicativo) ed apre un webserver sulla porta specificata nella configurazione.

Ho deciso di sviluppare questo sistema con Flask (modulo di Python) dato che l'ho già utilizzato e lo sviluppo di applicativi web risulta molto semplice e veloce. Esso è ben documentato e fornisce di default un *templating language* chiamanto *Jinja.*
Per ora lo script è impostato in verbose mode:

```
 * Serving Flask app "rasp-cptmrs" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5005/ (Press CTRL+C to quit)
 * Restarting with stat
[!] Config loaded correctly
 * Debugger is active!
 * Debugger PIN: 110-632-192
```

Problemi riscontrati e soluzioni adottate
Descritti nel capitolo precedente.

Punto della situazione rispetto alla pianificazione
Sono in perfetto orario secondo la tabella di marcia. Non avendo seguito a pieno la pianificazione iniziale ho potuto svolgere più attività in parallelo. Secondo il Gantt preventivo adesso dovrei sviluppare un sistema per la lettura dei dati dalle API, il quale ho già svolto per il caricamento degli eventi nel calendario.

Programma di massima per la prossima giornata di lavoro
Finire l'implementazione dei report mensili (Generazione query nel metodo *_generate_query)* e continuare la documentazione a casa.