





Improving the FERRY Meta-Data Management System

Antonio Segura **SIST Meeting** Monday, June 10, 2019

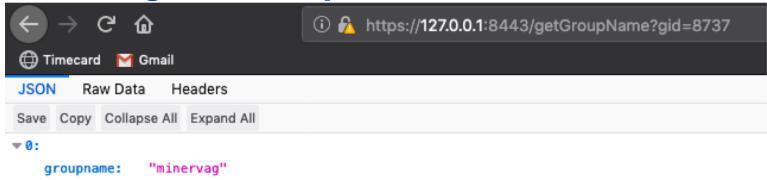
Project Overview

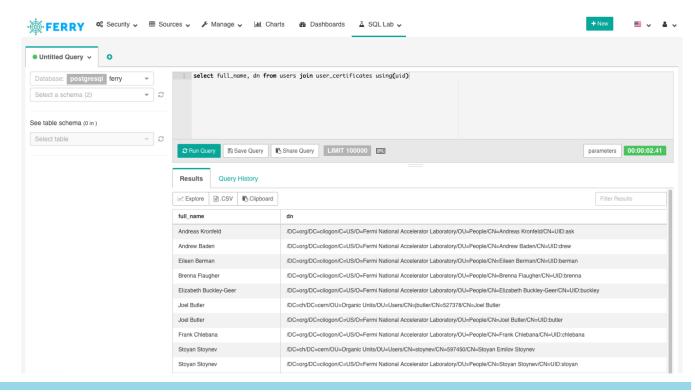
Improve the Frontier Experiments RegistRY (FERRY) metadata management system

- Refactor APIs (in the Go programming language) to reflect a more modular and standardized production of APIs for increased readability, additional conciseness, and overall consistent functionality
- Explore the visual representation of the FERRY database (currently using the Superset visualization tool by Apache)
- Develop additional functionality in Superset, regarding certificate authorization, relevant database views, and explore algebraic query writing- another layer of abstraction from SQL query writing for increased user-friendliness



Current Progress - Superset

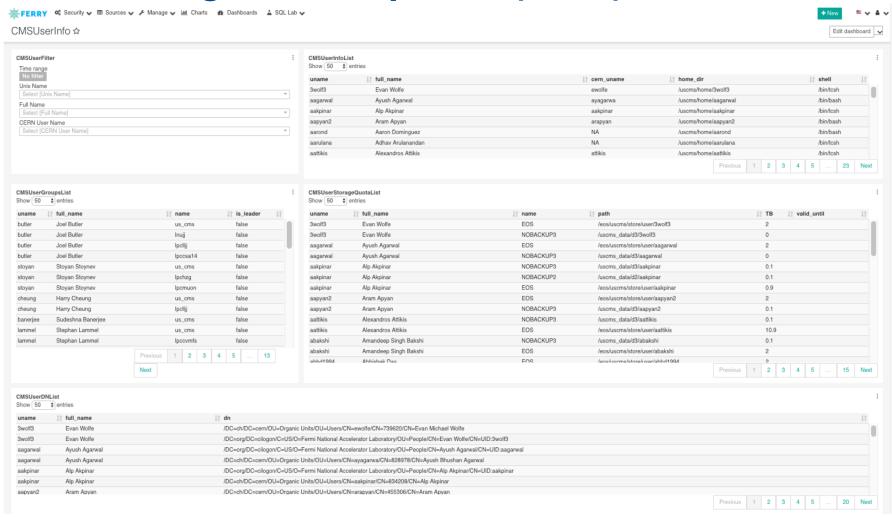






6/10/19

Current Progress – Superset (cont.)





Current Progress – Superset (cont.)

Expectations of Outcomes:

- To further understanding of SQL query language and relational database management
- To develop knowledge on web development using React.js, HTML, and CSS.
- Learn further cryptographic applications and practical usage via developing a certificate authorization system
- Create a significant advancement on the Superset data visualization application and add said progress to the open source contributions.



Current Progress - APIs

```
func getGroupsInAffiliationLegacy(w http.ResponseWriter, r *http.Request) {
4836
           startTime := time.Now()
4837
           w.Header().Set("Content-Type", "application/ison; charset=UTF-8")
4838
           q := r.URL.Query()
4839
           unitName := q.Get("unitname")
4840
4841
           if unitName == "" {
4842
               log.WithFields(QueryFields(r, startTime)).Error("No unit name specified in http query.")
4843
               fmt.Fprintf(w, "{ \"ferry_error\": \"No unitname specified.\" }")
4844
4845
4846
           lastupdate, parserr := stringToParsedTime(strings.TrimSpace(q.Get("last updated")))
4847
           if parserr != nil {
4848
               log.WithFields(QueryFields(r, startTime)).Error("Error parsing provided update time: " + parserr.Error())
4849
               fmt.Fprintf(w, "{ \"ferry_error\": \"Error parsing last_updated time. Check ferry logs. If provided, it should be an integer representing an epoch time.\"}")
4850
               return
4851
4852
                                                                                                                            4890
                                                                                                                                            for rows.Next() {
4853
                                                                                                                            4891
                                                                                                                                                var tmpGID int
4854
           checkerr := DBptr.QueryRow('select unitid from affiliation units where name=$1', unitName).Scan(&unitId)
                                                                                                                           4892
4855
                                                                                                                                                var tmpGname, tmpGtype string
                                                                                                                           4893
4856
           case checkerr == sql.ErrNoRows:
                                                                                                                                                var tmpGprimary bool
4857
               // set the header for success since we are already at the desired result
                                                                                                                            4894
                                                                                                                                                rows.Scan(&tmpGID, &tmpGname, &tmpGtype, &tmpGprimary)
               fmt.Fprintf(w, "{ \"ferry_error\": \"Affiliation unit does not exist.\" }")
4858
                                                                                                                            4895
                                                                                                                                                Entry.GId = tmpGID
4859
               log.WithFields(QueryFields(r, startTime)).Error("unit " + unitName + " not found in DB.")
                                                                                                                           4896
                                                                                                                                                Entry.GName = tmpGname
4860
               return
                                                                                                                           4897
                                                                                                                                                Entry.GType = tmpGtype
4861
           case checkerr != nil:
                                                                                                                           4898
                                                                                                                                                Entry.GPrimary = tmpGprimary
4862
               w.WriteHeader(http.StatusNotFound)
                                                                                                                           4899
                                                                                                                                                Out = append(Out, Entry)
4863
               fmt.Fprintf(w, "{ \"ferry_error\": \"Database error.\" }")
                                                                                                                            4900
4864
               log.WithFields(QueryFields(r, startTime)).Error("deleteUser: Error querying DB for unit " + unitName + ".")
                                                                                                                                            var output interface{}
4865
               return
                                                                                                                            4902
                                                                                                                                            if len(Out) == 0 {
4866
           default:
                                                                                                                                                type jsonerror struct {
4867
                                                                                                                            4904
                                                                                                                                                    Error string `json:"ferry_error"
4868
               rows, err := DBptr.Query('select gid, groups.name, groups.type, aug.is_primary
                                                                                                                           4905
4869
                                         from affiliation unit group as aug
                                                                                                                           4906
                                                                                                                                                var queryErr []jsonerror
4870
                                         join groups on aug.groupid = groups.groupid
                                                                                                                           4907
                                                                                                                                                queryErr = append(queryErr, jsonerror{"This affiliation unit has no groups."})
4871
                                         where aug.unitid=$1 and (aug.last_updated>=$2 or $2 is null)',
                                                                                                                           4908
                                                                                                                                                log.WithFields(QueryFields(r, startTime)).Error("This affiliation unit has no groups.")
4872
                   unitId, lastupdate)
                                                                                                                           4909
4873
               if err != nil {
                                                                                                                           4910
                                                                                                                                            } else {
4874
                   defer log.WithFields(QuervFields(r, startTime)).Error(err.Error())
                                                                                                                           4911
                                                                                                                                                log.WithFields(QueryFields(r, startTime)).Info("Success!")
4875
                   w.WriteHeader(http.StatusNotFound)
                                                                                                                           4912
4876
                   fmt.Fprintf(w, "{ \"ferry_error\": \"Error in DB query.\" }")
                                                                                                                            4913
4877
                   return
                                                                                                                           4914
                                                                                                                                            jsonoutput, err := json.Marshal(output)
4878
4879
                                                                                                                           4915
                                                                                                                                            if err != nil {
4880
               defer rows.Close()
                                                                                                                           4916
                                                                                                                                                log.WithFields(QueryFields(r, startTime)).Error(err.Error())
4881
               type jsonout struct {
                                                                                                                           4917
4882
                            int
                                   `ison:"gid"
                                                                                                                           4918
                                                                                                                                            fmt.Fprintf(w, string(jsonoutput))
4883
                   GName string 'json:"name"
                                                                                                                           4919
4884
                   GType string `json:"type"
                                                                                                                            4920
4885
                   GPrimary bool
                                   `json:"is_primary"
4886
4887
               var Entry jsonout
4888
               var Out []jsonout
```



Current Progress – APIs (cont.)

```
334
      func getGroupsInAffiliationUnit(c APIContext, i Input) (interface{}, []APIError) {
335
          var apiErr []APIError
336
337
          unitid := NewNullAttribute(UnitID)
338
339
          checkerr := c.DBtx.QueryRow(`select unitid from affiliation_units where name=$1`, i[UnitName]).Scan(&unitid)
340
          if checkerr != nil && checkerr != sql.ErrNoRows {
341
               log.WithFields(QueryFields(c.R, c.StartTime)).Error(checkerr)
342
               apiErr = append(apiErr, DefaultAPIError(ErrorDbQuery, nil))
343
               return nil, apiErr
344
345
          if !unitid.Valid {
346
               apiErr = append(apiErr, DefaultAPIError(ErrorDataNotFound, UnitName))
347
               return nil, apiErr
348
349
350
          rows, checkerr := c.DBtx.Query('select gid, groups.name, groups.type, aug.is_primary
351
          from affiliation unit group as aug
          join groups on aug.groupid = groups.groupid
352
353
          where aug.unitid=$1 and (aug.last_updated>=$2 or $2 is null)',
354
          unitid, i[LastUpdated])
355
          if checkerr != nil {
356
               log.WithFields(QueryFields(c.R, c.StartTime)).Error(checkerr)
357
               apiErr = append(apiErr, DefaultAPIError(ErrorDbQuery, nil))
358
               return nil, apiErr
359
360
          defer rows.Close()
361
362
          type jsonentry map[Attribute] interface{}
363
          out := make([]jsonentry, 0)
364
365
          for rows.Next() {
366
               row := NewMapNullAttribute(GID, GroupName, GroupType, Primary)
               rows.Scan(row[GID], row[GroupName], row[GroupType], row[Primary])
367
368
               entry := jsonentry {
369
                  GID:
                               row[GID].Data,
370
                  GroupName: row[GroupName].Data,
371
                  GroupType: row[GroupType].Data,
372
                  Primary:
                              row[Primary].Data,
373
374
              out = append(out, entry)
375
376
          return out, nil
377
```



Current Progress – APIs (cont.)

Expectations of Outcomes:

- Furthered understanding of better Software Engineering practices
- Increased knowledge of Go, Python, and SQL programming languages
- Practical understanding of management of a database system
- Overall increased knowledge across multiple fields of Computer Science and various programming languages

