```r
### SCRIPT 4 - R ###
# Obtención de crequisitos de cualificación empresarial desde ATOM
#
#########################################################
library(XML)
library(dplyr)
library(foreach)
library(doParallel)

#setup parallel backend to use many processors
cores=detectCores()
cl <- makeCluster(cores[1]-1, type = "PSOCK", outfile = "log.txt") #not to overload your computer #,outfile="log.txt"
registerDoParallel(cl)

f_ini <- Sys.time()

# Cargar el fichero atom
setwd("C:/Users/guillermo.alonso/Desktop/Tesis Guillermo/_BD actualizada/Datos atom/PLACSP")

# configuración del script
PATH <- "licitacionesPerfilesContratanteCompleto3_2025"
RData_FILE <- "4.30_TendererQualificationRequestet_2025.Rdata"

## valor del ID inical que se inserta en TQR_ID, cambiar cada ejecución.
ID_INI = "2025"


#'Busca el padre con el tag inicado
#' @param xml_a punto de partida
#' @param tag etiqueta del padre que busca
#'
DamePadre <- function(xml_a,tag){
  xml_p <- xml_a
  repeat{
    xml_p <- xmlParent(xml_p)
    if(xmlName(xml_p) == tag) break
  }
  return(xml_p)
}

# carga los nombres de los codigos RTC
    RTC_Codes <- xmlToDataFrame(getNodeSet(xmlParse("DeclarationTypeCode-2.0.gc.xml"), "//Row"))
    colnames(RTC_Codes) <- c("code","nombre","name")

#' retorno la descripcion del codigo
#' @param code codigo a buscar
#'
getRequirementTypeName <- function(code){
  ifelse(sum(RTC_Codes$code==code)==0 ,NA,RTC_Codes[RTC_Codes$code==code,"nombre"])
}


# carga los nombres de los codigos CC
CC_Codes <- xmlToDataFrame(getNodeSet(xmlParse(
"RequiredBusinessProfileCode-1.05.gc.xml"),"//Row"))
colnames(CC_Codes) <- c("code","name","nombre")

#' retorno la descripcion del codigo
#' @param code codigo a busca8ç
#'
getClassificationCategory <- function(code){
  ifelse(sum(CC_Codes$code==code)==0,NA,CC_Codes[CC_Codes$code==code,"nombre"])
}


# Creamos un data.frame vacio para añadir los atributos.
nombres_TQR <- c("ID","ContractID","entryID","updated","PersonalSituation", "Description")
nombres_CC  <- c("TQR_ID", "CodeValue","CodeValue_des")
nombres_RTC <- c("TQR_ID", "RequirementTypeCode","RequirementTypeName")
```

```
69
70  Proy_TQR <- data.frame(matrix(NA,0,length(nombres_TQR)))
71  colnames(Proy_TQR) <- nombres_TQR
72  Proy_CC <- data.frame(matrix(NA,0,length(nombres_CC)))
73  colnames(Proy_CC) <- nombres_CC
74  Proy_RTC <- data.frame(matrix(NA,0,length(nombres_RTC)))
75  colnames(Proy_RTC) <- nombres_RTC
76
77
78  arch <-  list.files(path = PATH, pattern = "\\.atom$", full.names = F)
79
80  Proylist <- foreach(ci = 1:length(arch), .packages = c("XML"), .verbose = F) %dopar% {
81
82    archivoTemp <- xmlParse(paste0(PATH,"/",arch[ci]))
83
84    #  list of the children or sub-elements of an XML node whose tag name matches the
      one specified
85    cat(paste(ci," Buscando 'ContractModification' en ",arch[ci],"\n"))
86    buscar <-  xmlElementsByTagName(xmlRoot(archivoTemp),"TendererQualificationRequest"
      , recursive = T)
87
88    Proy_TQR <- data.frame(matrix(NA,0,length(nombres_TQR)))
89    colnames(Proy_TQR) <- nombres_TQR
90    Proy_CC <- data.frame(matrix(NA,0,length(nombres_CC)))
91    colnames(Proy_CC) <- nombres_CC
92    Proy_RTC <- data.frame(matrix(NA,0,length(nombres_RTC)))
93    colnames(Proy_RTC) <- nombres_RTC
94
95    i=1
96    for (xmla in buscar){
97
98      foo_TQR <- Proy_TQR[1,]
99      foo_CC <- Proy_CC[1,]
100     foo_RTC <- Proy_RTC[1,]
101     foo_TQR[1,] <- foo_CC[1,] <- foo_RTC[1,] <- NA
102
103     foo_TQR$ID <- paste0(ID_INI,"_",ci,"_",i)     ; i=i+1
104     foo_TQR$PersonalSituation <- xmlValue(xmla[["PersonalSituation"]])
105     foo_TQR$Description <- xmlValue(xmla[["Description"]])
106
107     # busca el padre para el ContractFolderStatus sube por la cadena
        entry.ContractFolderStatus.ContractModification
108     xmlp <- DamePadre(xmla, "ContractFolderStatus")
109
110     foo_TQR$ContractID <- xmlValue( xmlChildren(xmlp)$ContractFolderID )
111
112     bar <- strsplit(xmlValue( xmlChildren(xmlParent(xmlp))$id ), "/")[[1]] #Split la
        URL por /
113     foo_TQR$entryID <- bar[length(bar)]                                          #
        que queda con el utlimo valor
114     foo_TQR$updated <- xmlValue( xmlChildren(xmlParent(xmlp))$updated )
115
116
117     ## ClassificationCategory
118     xmlcc <- xmlChildren(xmla)$RequiredBusinessClassificationScheme
119     if (!is.null(xmlcc)){
120       elements <- xmlElementsByTagName(xmlcc,"ClassificationCategory")
121       if(length(elements)>0){
122         for (idx in 1:length(elements)) {
123           e <- elements[[idx]]
124           foo_CC[idx,"CodeValue"] <- xmlValue(e[["CodeValue"]])
125           if (!is.na(foo_CC$CodeValue[idx])){
126             foo_CC$TQR_ID[idx] <- foo_TQR$ID
127             foo_CC$CodeValue_des[idx] <- xmlValue(e[["Description"]])
128             if (is.null(foo_CC$CodeValue_des[idx])) foo_CC$CodeValue_des[idx] <-
              getClassificationCategory(foo_CC$CodeValue_des[idx])
129           } else foo_CC <- foo_CC[-idx,]
130         }
131       } else foo_CC <- foo_CC[-1,]
132     } else foo_CC <- foo_CC[-1,]
133
134     ## SpecificTendererRequirement
135     elements <- xmlElementsByTagName(xmla,"SpecificTendererRequirement")
```

```
136       if(length(elements)>0){
137         for (idx in 1:length(elements)) {
138           e <- elements[[idx]]
139           foo_RTC[idx,"RequirementTypeCode"] <- xmlValue(e[["RequirementTypeCode"]])
140           if (!is.na(foo_RTC$RequirementTypeCode[idx])){
141             foo_RTC$TQR_ID[idx] <- foo_TQR$ID
142             # foo_RTC$RequirementTypeName <-
              xmlGetAttr(xmla[["RequirementTypeCode"]],"name")
143             foo_RTC$RequirementTypeName[idx] <- getRequirementTypeName(foo_RTC$
              RequirementTypeCode[idx])
144           } else foo_RTC <- foo_RTC[-idx,]
145         }
146       } else foo_RTC <- foo_RTC[-1,]
147
148       # acummulamos elementos encontrados
149       Proy_TQR <- rbind(Proy_TQR,as.data.frame(foo_TQR))
150       Proy_CC  <- rbind(Proy_CC,as.data.frame(foo_CC))
151       Proy_RTC <- rbind(Proy_RTC,as.data.frame(foo_RTC))
152
153     }
154     cat(paste(ci," \t\t\t\t\t-> Encontrados ",length(buscar),"\n")) #traza
155     list(Proy_TQR,Proy_CC,Proy_RTC)
156   }
157
158   #stop cluster
159   stopCluster(cl)
160
161   print("fusion")
162
163   for (f in 1:length(Proylist)){
164     cat(paste(f,","))
165     Proy_TQR <- rbind(Proy_TQR,Proylist[[f]][[1]])
166     Proy_CC  <- rbind(Proy_CC,Proylist[[f]][[2]])
167     Proy_RTC <- rbind(Proy_RTC,Proylist[[f]][[3]])
168   }
169   print("salvar")
170   save(Proy_TQR,Proy_CC,Proy_RTC,nombres_CC, nombres_RTC, nombres_TQR, file = RData_FILE
      )
171
172   #print("DB")
173   # library(RODBC)
174   # mdb <- odbcDriverConnect(paste0("Driver={Microsoft Access Driver (*.mdb,
      *.accdb)};DBQ=",DB))
175   # sqlSave(mdb, Proy_TQR, tablename = "TendererQualificationRequest", append = T,
      rownames = F)
176   # sqlSave(mdb, Proy_CC,  tablename = "ClassificationCategory", append = T, rownames =
      F)
177   #
178   # ri <- 1
179   # rl <- nrow(Proy_RTC)
180   # repeat {
181   #   rf <- ifelse(ri+10000 < rl,ri+10000,rl)
182   #   sqlSave(mdb, Proy_RTC[ri:rf,], tablename = "SpecificTendererRequirement", append
      = T, rownames = F, fast = T)
183   #   ri <- rf
184   #
185   #   if (ri==rl) break
186   #   else print(round(rf/rl*100,digits = 2))
187   # }
188   # odbcClose(mdb)
189
190
191   #load(file =  RData_FILE)
192   #library(DBI)
193   #con <- dbConnect(RMariaDB::MariaDB(), user="root", dbname="atomdb")
194   #dbWriteTable(con,"TendererQualificationRequest", Proy_TQR,  row.names = F, append=T)
195   #dbWriteTable(con,"ClassificationCategory", Proy_CC,  row.names = F, append=T)
196   #dbWriteTable(con,"SpecificTendererRequirement", Proy_RTC,  row.names = F, append=T)
197   #dbDisconnect(con)
198
199
200
```