# MANUAL TÉCNICO

## 1. Información del sistema:

- • OS: Ubuntu 22.04.2 LTS
- • OS Type: 64-bit
- • CPU: Intel core i5 2.50 GHz
- • GPU: Mesa Intel HD Graphics 4400
- • Versión de java: 17
- • RAM: 8 GB
- • IDE: IntelliJ idea
- • Control de versiones: git 2.34.1
- • Github: https://github.com/Hatsune02/QuizCraft.git

## 2. Descripción del proyecto:

Consiste en la implementación de JFlex y CUP para la creación de analizadores léxicos y sintácticos. Esto con el fin de poder relizar una aplicación web que pueda aceptar un lenguaje para crear, modificar y eliminar: Usuarios, trivias y componentes, Asi como también servirán para leer estructuras de datos, especialmente la XSON que es una estructura que combina el xml y json. También se implementa lo necesario para java web como jsp, servidores (tomcat), servlets y demás. Por ultimo se usa Kotlin para poder crear las vistas en la aplicación cliente.

## 3. Detalles Técnicos

### Patrón Model-View-Controller (MVC)

La arquitectura del proyecto está basada en el patrón **Model-View-Controller (MVC)**, que divide el sistema en tres componentes principales:

- **Modelo (Model)**: Maneja la lógica de negocio y los datos. En este proyecto, el modelo se implementa en la capa de persistencia, donde se interactúa con la base de datos para almacenar y recuperar trivias, usuarios, y reportes. También incluye las consultas SQLKV y las funciones de importación/exportación de trivias.
- **Vista (View)**: Se encarga de mostrar la información al usuario. Para la interfaz web, se usaron **JSPs** que se comunican con los servlets para obtener los datos. En la versión móvil, se usaron **Layouts XML** en Android para presentar las trivias y resultados.
- **Controlador (Controller)**: Procesa las solicitudes de los usuarios y coordina la interacción entre la vista y el modelo. Los **servlets** en Java manejan esta capa, interpretando las acciones del usuario, ejecutando las operaciones correspondientes en el modelo, y devolviendo los resultados a la vista.

### Java Web (JSP y Servlets)

**Java Server Pages (JSP)** y **Servlets** fueron las tecnologías clave utilizadas para la capa de presentación y el manejo de la lógica de negocio en el servidor.

- **Servlets** se encargan de manejar las peticiones HTTP del usuario y la lógica de control del sistema. Implementé varios servlets para procesar las acciones de los usuarios, como iniciar sesión, ejecutar consultas y gestionar las trivias.
- **JSP** permitió la generación dinámica de contenido HTML, permitiendo la visualización de trivias y reportes en la interfaz web. Los JSP se encargan de interactuar con los objetos del modelo para mostrar datos actualizados y procesar entradas de los usuarios.

**Aplicación en el proyecto:**

- Los servlets implementan la capa de control en el patrón MVC, gestionando las peticiones del usuario y delegando las tareas a las capas de vista o modelo según sea necesario.
- Las páginas JSP constituyen la capa de vista, donde se muestra la información de las trivias, los reportes y las consultas realizadas.

## 2.3 Android Studio

**Android Studio** se utilizó para desarrollar la versión móvil de la plataforma de trivias, facilitando la creación de una interfaz amigable y adaptable a dispositivos móviles.

- La versión móvil incluye características básicas como la participación en trivias, visualización de resultados, y consultas a través del lenguaje SQLKV.
- Utilicé **Java** para la lógica interna de la aplicación y **XML** para definir la interfaz de usuario en Android, siguiendo el patrón de arquitectura MVC en el lado del cliente.

**Aplicación en el proyecto:**

- Las **Activities** actúan como controladores en la arquitectura MVC, gestionando la interacción del usuario con la interfaz.
- Las vistas en Android se implementaron con **layouts XML** y el modelo se conectó a la base de datos para obtener y enviar información de las trivias a través de un backend compartido.

## Descripción General del Uso de JFlex y CUP en el Programa

**Flex** y **CUP** son herramientas que permiten la generación automática de analizadores léxicos y sintácticos, respectivamente. Estas herramientas fueron fundamentales en la construcción del **parser** para procesar los lenguajes específicos del dominio (DSL) utilizados en la aplicación.

- **Flex** (JFlex) se utilizó para definir las reglas léxicas, permitiendo reconocer los tokens a partir de las entradas proporcionadas por el usuario. Esto fue útil para la implementación del lenguaje para objetos **XSON**, persistencia de datos **JSON** y del lenguaje de reportería **SQLKV**, que permite consultar los datos de las trivias de manera estructurada.
- **CUP** se encargó de la parte sintáctica del análisis. Definí una gramática que interpretara correctamente las consultas en el lenguaje XSON, JSON y SQLKV, validando la sintaxis y traduciendo la consulta a una forma manejable por el servidor para ejecutar las consultas en la base de datos.

**Aplicación en el proyecto:**

- Se utilizó JFlex para crear un **lexer** que identificara los tokens dentro de las consultas de XSON y SQLKV.
- CUP se usó para construir el **parser** que procesaba y ejecutaba estas consultas en la base de datos de trivias.

## Lenguaje XSON

## Lexer:

```
/* ___Reserved words___ */

   /* XMLSON labels */
xson = {x}{s}{o}{n}
equal = [=]
version = {v}{e}{r}{s}{i}{o}{n}
one = (1\.0)
q_mark = [?]
realizar_solicitud = {r}{e}{a}{l}{i}{z}{a}{r}_{s}{o}{l}{i}{c}{i}{t}{u}{d}
realizar_solicitudes = {realizar_solicitud}{e}{s}
fin = ({f}{i}{n})
solicitud = ({s}{o}{l}{i}{c}{i}{t}{u}{d})
solicitudes = ({solicitud}{e}{s})
realizada = ({r}{e}{a}{l}{i}{z}{a}{d}{a})
fin_solicitud_realizada = ({fin}_{solicitud}_{realizada})
fin_solicitudes_realizada = ({fin}_{solicitudes}_{realizada})

   /* parameters */
usuario_nuevo =  USUARIO_NUEVO
datos_usuario =  DATOS_USUARIO
usuario =  USUARIO
password =  PASSWORD
insitucion =  INSTITUCION
nombre =  NOMBRE
fecha_creacion =  FECHA_CREACION

modificar_usuario =  MODIFICAR_USUARIO
usuario_antiguo =  USUARIO_ANTIGUO
nuevo_password =  NUEVO_PASSWORD
fecha_modificacion =  FECHA_MODIFICACION

eliminar_usuario =  ELIMINAR_USUARIO

login_usuario =  LOGIN_USUARIO
ver_trivias =  VER_TRIVIAS
add_data =  ADD_DATA

tiempo_total =  TIEMPO_TOTAL
estado =  ESTADO
punteo =  PUNTEO

nueva_trivia =  NUEVA_TRIVIA
parametros_trivia =  PARAMETROS_TRIVIA
id_trivia =  ID_TRIVIA
```

```
tiempo_pregunta =  TIEMPO_PREGUNTA
usuario_creacion =  USUARIO_CREACION
tema =  TEMA

eliminar_trivia =  ELIMINAR_TRIVIA
modificar_trivia =  MODIFICAR_TRIVIA
agregar_componente =  AGREGAR_COMPONENTE
parametros_componente =  PARAMETROS_COMPONENTE

id =  ID
trivia =  TRIVIA
clase =  CLASE
indice =  INDICE
texto_visible =  TEXTO_VISIBLE
opciones =  OPCIONES
filas =  FILAS
columnas =  COLUMNAS
respuesta =  RESPUESTA

campo_texto = CAMPO_TEXTO
area_texto = AREA_TEXTO
checkbox = CHECKBOX
radio = RADIO
fichero = FICHERO
combo = COMBO

eliminar_componente = ELIMINAR_COMPONENTE
modificar_componenente = MODIFICAR_COMPONENTE

/* Structures */

LBracket = [
RBracket = ]
LBrace = {
RBrace = }
LThan = <
GThan = >
EX = !
Colon = :
Comma = ,
VerticalBar = |

/* Strings */

Q = "
StringContent = [a-zA-Z0-9]+[-:;$a-zA-Z0-9]*
```

/* Others */

Identifier = [-_$][-_$a-zA-Z0-9]+
Integer = [0-9]+

## Parser

Producciones terminales:
{XSON, EQUAL, VERSION, ONE, REALIZAR_SOLICITUD, REALIZAR_SOLICITUDES, FIN_SOLICITUD_REALIZADA, FIN_SOLICITUDES_REALIZADA, MODIFICAR_USUARIO, USUARIO_ANTIGUO, NUEVO_PASSWORD, FECHA_MODIFICACION, ELIMINAR_USUARIO, LOGIN_USUARIO, NUEVA_TRIVIA, PARAMETROS_TRIVIA, ID_TRIVIA, TIEMPO_PREGUNTA, USUARIO_CREACION, TEMA, ELIMINAR_TRIVIA, MODIFICAR_TRIVIA, AGREGAR_COMPONENTE, PARAMETROS_COMPONENTE, ID, TRIVIA, CLASE, INDICE, TEXTO_VISIBLE, OPCIONES, FILAS, COLUMNAS, RESPUESTA, CAMPO_TEXTO, AREA_TEXTO, CHECKBOX, RADIO, FICHERO, COMBO, ELIMINAR_COMPONENTE, MODIFICAR_COMPONENTE, VERTICAL_BAR, VER_TRIVIAS, ADD_DATA, TIEMPO_TOTAL, ESTADO, PUNTEO, LBRACKET, RBRACKET, LBRACE, RBRACE, LT, GT, EX, Q, STRINGCONTENT, IDENTIFIER, USUARIO_NUEVO, DATOS_USUARIO, USUARIO, PASSWORD, INSTITUCION, NOMBRE, FECHA_CREACION, COLON, COMMA, QM, DIGIT}

Producciones no terminales:
{s, xson_label, open_request, close_request, open_requests, close_requests, open_user_data, close_data, open_trivia_data, open_component_data, parameter, user, password, name, institution, create_date, old_user, new_user, new_password, update_date, id_trivia, create_user, topic, id, trivia, visible_text, options, answer, question_time, clase, index, line, columns, data, data_user, data_trivia, data_component, new_user_request, update_user_request, delete_user_request, login_request, new_trivia_request, update_trivia_request, delete_trivia_request, new_component_request, update_component_request, delete_component_request, request, requestP, requests, view_trivias_request, add_data_request, data_collected_data, total_time, score,estado,clase_content, string_content, string_contentP, multiple_string}

start with s;

```
s ::=  xson_label requests
   | xson_label request;

requests ::= open_requests requestP close_requests ;

requestP ::= request
        | requestP request;

request ::=   new_user_request
        | new_trivia_request
        | new_component_request
        | update_user_request
        | update_trivia_request
        | update_component_request
```

```
        | delete_user_request
        | delete_trivia_request
        | delete_component_request
        | login_request
        | view_trivias_request
        | add_data_request;

xson_label ::= LT QM XSON VERSION EQUAL Q ONE Q QM GT;

open_requests ::= LT EX REALIZAR_SOLICITUDES GT;

close_requests ::= LT EX FIN_SOLICITUDES_REALIZADA GT;

open_request ::= LT EX REALIZAR_SOLICITUD COLON;

close_request ::= LT FIN_SOLICITUD_REALIZADA EX GT;

new_user_request ::= open_request Q USUARIO_NUEVO Q GT data_user close_request;

open_user_data ::= LBRACE Q DATOS_USUARIO Q COLON LBRACKET LBRACE;

close_data ::= RBRACE RBRACKET RBRACE;

data_user ::= open_user_data data close_data;

update_user_request ::= open_request Q MODIFICAR_USUARIO Q GT data_user close_request

delete_user_request ::= open_request Q ELIMINAR_USUARIO Q GT data_user close_request

login_request ::= open_request Q LOGIN_USUARIO Q GT data_user close_request

open_trivia_data ::= LBRACE Q PARAMETROS_TRIVIA Q COLON LBRACKET LBRACE;

new_trivia_request ::= open_request Q NUEVA_TRIVIA Q GT data_trivia close_request

data_trivia ::= open_trivia_data data close_data

update_trivia_request ::= open_request Q MODIFICAR_TRIVIA Q GT data_trivia close_request

delete_trivia_request ::= open_request Q ELIMINAR_TRIVIA Q GT data_trivia close_request

open_component_data ::= LBRACE Q PARAMETROS_COMPONENTE Q COLON LBRACKET LBRACE;

new_component_request ::= open_request Q AGREGAR_COMPONENTE Q GT data_component
close_request

data_component ::= open_component_data data close_data
```

update_component_request ::= open_request Q MODIFICAR_COMPONENTE Q GT data_component close_request

delete_component_request ::= open_request Q ELIMINAR_COMPONENTE Q GT data_component close_request

view_trivias_request ::= open_request Q VER_TRIVIAS Q GT close_request

add_data_request ::= open_request Q ADD_DATA Q GT data_collected_data close_request

data_collected_data ::= LBRACE user COMMA trivia COMMA total_time COMMA estado COMMA score RBRACE

total_time ::= Q TIEMPO_TOTAL Q COLON DIGIT

estado ::= Q ESTADO Q COLON Q string_contentP Q

score ::= Q PUNTEO Q COLON DIGIT:d

data ::= parameter
| data COMMA parameter

parameter ::= user
| password
| name
| institution
| create_date
| old_user
| new_user
| new_password
| update_date
| id_trivia
| question_time
| create_user
| topic
| id
| trivia
| clase
| index
| visible_text
| options
| line
| columns
| answer

user ::= Q USUARIO Q COLON Q string_content Q

password ::= Q PASSWORD Q COLON Q string_content Q

name ::= Q NOMBRE Q COLON Q string_contentP Q

institution ::= Q INSTITUCION Q COLON Q string_contentP Q

create_date ::= Q FECHA_CREACION Q COLON Q string_content Q

old_user ::= Q USUARIO_ANTIGUO Q COLON Q string_content Q

new_user ::= Q USUARIO_NUEVO Q COLON Q string_content Q

new_password ::= Q NUEVO_PASSWORD Q COLON Q string_content Q

update_date ::= Q FECHA_MODIFICACION Q COLON Q string_content Q

id_trivia ::= Q ID_TRIVIA Q COLON Q IDENTIFIER Q

question_time ::= Q TIEMPO_PREGUNTA Q COLON DIGIT

create_user ::= Q USUARIO_CREACION Q COLON Q string_content Q

topic ::= Q TEMA Q COLON Q string_contentP: Q

id ::= Q ID Q COLON Q IDENTIFIER Q

trivia ::= Q TRIVIA Q COLON Q IDENTIFIER Q

clase ::= Q CLASE Q COLON Q clase_content Q

index ::= Q INDICE Q COLON Q DIGIT Q

visible_text ::= Q TEXTO_VISIBLE Q COLON Q string_contentP Q

options ::= Q OPCIONES Q COLON Q multiple_string Q

line ::= Q FILAS Q COLON Q DIGIT Q

columns ::= Q COLUMNAS Q COLON Q DIGIT Q

answer ::= Q RESPUESTA Q COLON Q multiple_string Q

clase_content ::= CAMPO_TEXTO
| AREA_TEXTO
| CHECKBOX
| RADIO
| FICHERO
| COMBO

```
multiple_string ::= string_contentP
        | multiple_string VERTICAL_BAR string_contentP

string_contentP ::= string_content
        | string_contentP string_content

string_content ::= STRINGCONTENT
        | IDENTIFIER
        | DIGIT
        | NOMBRE
        | COLON
        | COMMA
        | QM
```

## Lenguaje para la persistencia de datos

## Lexer:

```
/* ___Reserved words___ */
db_user =  db .user
db_trivia =  db .trivia
new_trivia =  new .trivia
   /* parameters */

usuario =   "USUARIO "
password =   "PASSWORD "
insitucion =   "INSTITUCION "
nombre =   "NOMBRE "
fecha_creacion =   "FECHA_CREACION "
fecha_modificacion =   "FECHA_MODIFICACION "

id_trivia =   "ID_TRIVIA "
tiempo_pregunta =   "TIEMPO_PREGUNTA "
usuario_creacion =   "USUARIO_CREACION "
tema =   "TEMA "

id =   "ID "
trivia =   "TRIVIA "
clase =   "CLASE "
indice =   "INDICE "
texto_visible =   "TEXTO_VISIBLE "
opciones =   "OPCIONES "
filas =   "FILAS "
columnas =   "COLUMNAS "
respuesta =   "RESPUESTA "

campo_texto =   "CAMPO_TEXTO "
area_texto =   "AREA_TEXTO "
checkbox =   "CHECKBOX "
radio =   "RADIO "
fichero =   "FICHERO "
combo =   "COMBO "
none =   "NONE "

estructura =   "ESTRUCTURA "
datos_recopilados =   "DATOS_RECOPILADOS "

tiempo_total =   "TIEMPO_TOTAL "
punteo =   "PUNTEO "
estado =   "ESTADO "

/* Structures */
```

```
LBrace =  {
RBrace =  }
LParen =  (
RParen =  )
Colon =  :
Comma =  ,

/* Strings */

Q =  "
StringContent =  ^ " +
String = {Q}{StringContent}{Q}
```

## Parser:

Producciones terminales
{DB_USER, DB_TRIVIA, NEW_TRIVIA, ID_TRIVIA, TIEMPO_PREGUNTA, USUARIO_CREACION, TEMA, ID, TRIVIA, CLASE, INDICE, TEXTO_VISIBLE, OPCIONES, FILAS, COLUMNAS, RESPUESTA, CAMPO_TEXTO, AREA_TEXTO, CHECKBOX, RADIO, FICHERO, COMBO, NONE,  LPAREN, RPAREN, LBRACE, RBRACE, ESTRUCTURA, DATOS_RECOPILADOS, TIEMPO_TOTAL, PUNTEO, ESTADO, STRING, USUARIO, PASSWORD, INSTITUCION, NOMBRE, FECHA_CREACION, FECHA_MODIFICACION, COLON, COMMA, DIGIT}

Producciones no terminals
{s, db_user, db_trivia, new_trivia, user, users, username, password, name, institution, creation_date, update_date, trivia, trivias, id_triva, topic, creation_user, question_time, component, components, structure, id_component, param_trivia, visible_text, options, answer, clase, class_type, lines, columns, data, collected_data, datas,total_time, score, index,done}

start with s;

```
s ::=  db_user
   | db_trivia
   | new_trivia

db_user ::= DB_USER LPAREN users RPAREN
      | DB_USER LPAREN RPAREN


users ::= user
      | users: COMMA user

user ::= LBRACE username password name institution creation_date update_date RBRACE

username ::= USUARIO COLON STRING COMMA
```

password ::= PASSWORD COLON STRING COMMA

name ::= NOMBRE COLON STRING COMMA

institution ::= INSTITUCION COLON STRING COMMA

creation_date ::= FECHA_CREACION COLON STRING COMMA

update_date ::= FECHA_MODIFICACION COLON STRING

db_trivia ::= DB_TRIVIA LPAREN trivias RPAREN
| DB_TRIVIA LPAREN RPAREN

new_trivia ::= NEW_TRIVIA LPAREN trivia RPAREN

trivias ::= trivia
| trivias: COMMA trivia

trivia ::= LBRACE id_trivia name topic question_time creation_user creation_date structure datas RBRACE

id_triva ::= ID_TRIVIA COLON STRING COMMA

topic ::= TEMA COLON STRING COMMA

question_time ::= TIEMPO_PREGUNTA COLON DIGIT COMMA

creation_user ::= USUARIO_CREACION COLON STRING COMMA

structure ::= ESTRUCTURA COLON LPAREN components RPAREN COMMA
| ESTRUCTURA COLON LPAREN RPAREN COMMA

components ::= component
| components COMMA component

component ::= LBRACE id_component param_trivia clase index visible_text options lines columns answer RBRACE

id_component ::= ID COLON STRING COMMA

param_trivia ::= TRIVIA COLON STRING:s COMMA

clase ::= CLASE COLON class_type:c COMMA

index ::= INDICE COLON DIGIT:d COMMA

visible_text ::= TEXTO_VISIBLE COLON STRING:s COMMA

```
options ::= OPCIONES COLON STRING:s COMMA

lines ::= FILAS COLON DIGIT:d COMMA

columns ::= COLUMNAS COLON DIGIT:d COMMA

answer ::= RESPUESTA COLON STRING:s

class_type ::= CAMPO_TEXTO
            | AREA_TEXTO
            | CHECKBOX
            | RADIO
            | FICHERO
            | COMBO
            | NONE
            ;

datas ::= DATOS_RECOPILADOS COLON LPAREN RPAREN
      | DATOS_RECOPILADOS COLON LPAREN collected_data RPAREN

collected_data ::= data
            | collected_data COMMA data

data ::= LBRACE username param_trivia total_time done score RBRACE

total_time ::= TIEMPO_TOTAL COLON DIGIT:d COMMA

done ::= ESTADO COLON STRING:s COMMA

score ::= PUNTEO COLON DIGIT:d
```

## Lenguaje para SQLKV

## Lexer:

```
/* Reserved words*/
seleccionar =  SELECCIONAR
reporte =  REPORTE
filtrar =  FILTRAR
por =  POR
usuario =  USUARIO
tiempo_total =  TIEMPO_TOTAL
punteo =  PUNTEO
/* Operators */
RelatedOperations = "<=" | ">=" | "=" | "<" | ">"
and =  AND
or =  OR

/* Numbers */
Digit = [0-9]+
Decimal = {Digit} .{Digit}

/* Strings */
Identifier = [-_$][-_$a-zA-Z0-9]+
Q = [ "]
q = [ ']
StringContent = [^ " '    ]*[-_$a-zA-Z][^ " '    ]*
String = {Q}{StringContent}{Q} | {q}{StringContent}{q}


/* Structures */
Comma = [,]
```

## Parser:

Producciones terminales
{SELECCIONAR, REPORTE, FILTRAR, POR, USUARIO, TIEMPO_TOTAL, PUNTEO, COMMA, AND, OR, ID, DIGIT, STRING, REL_OP, ERROR}

Producciones no terminales
{ instruction, expr, reserved_word, listC, condition, conditions, filter, s}

start with s;

```
s ::= instruction;

instruction ::= SELECCIONAR REPORTE
              | SELECCIONAR REPORTE listC
              | SELECCIONAR REPORTE filter
              | SELECCIONAR REPORTE listC filter

filter ::= FILTRAR POR conditions

conditions ::= condition
           | conditions AND condition
           | conditions OR condition

condition ::= reserved_word REL_OP expr

listC      ::= ID
           | listC COMMA ID

reserved_word ::= USUARIO
              | TIEMPO_TOTAL
              | PUNTEO

expr       ::= DIGIT
           | STRING
```