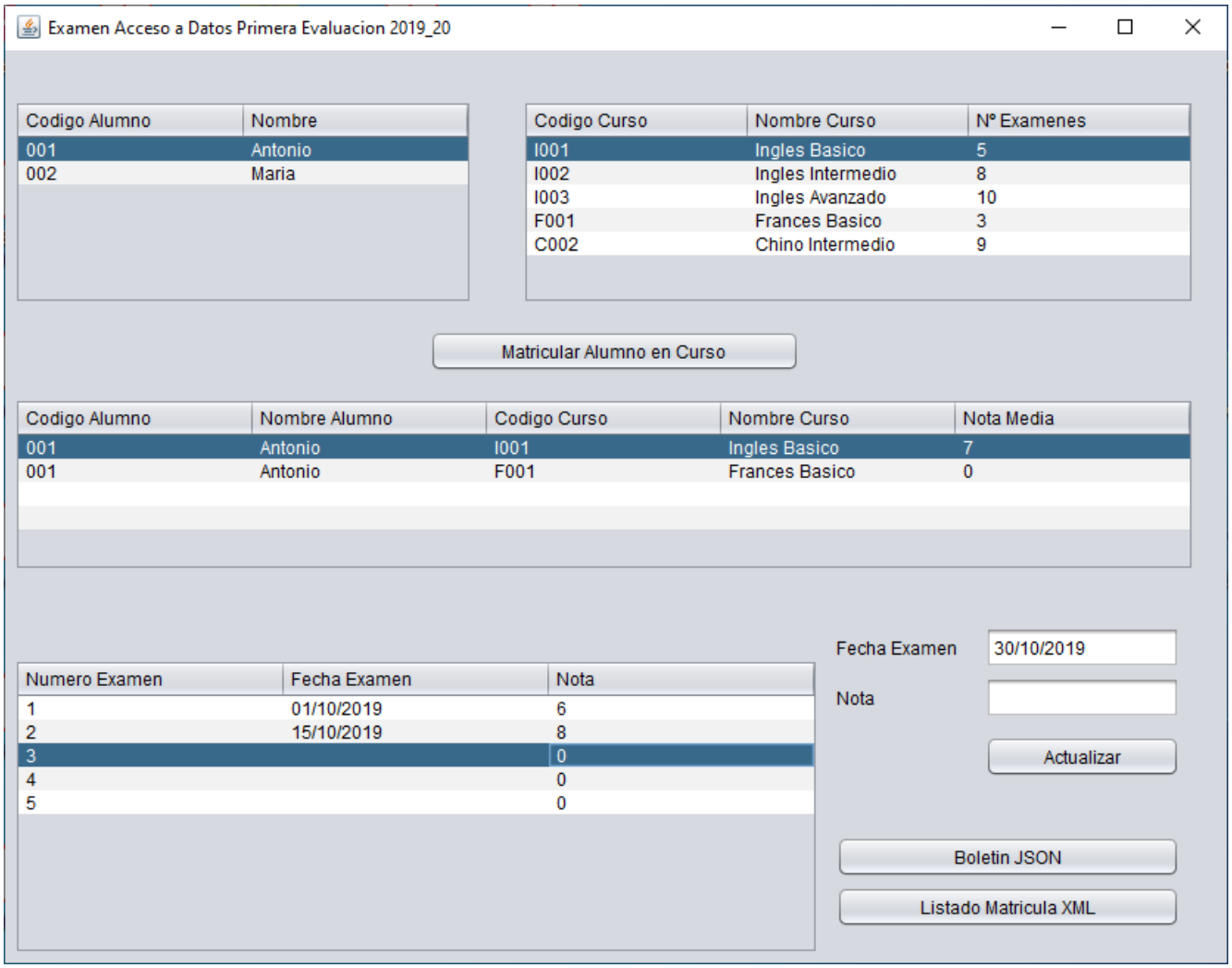
PRACTICA 01.

El alumno/a ha de realizar una aplicación en Java para llevar la gestión de una Academia de Idiomas. La base de datos se almacena en un SGBD Oracle.

La pantalla principal de la aplicación será la siguiente:



- Tendremos un JTable donde se mostrarán nuestros Alumnos (**JTableAlumnos**).
- Tendremos un JTable donde se mostrarán nuestros Cursos (**JTableCursos**).
- Tendremos un JTable donde se mostrarán los Cursos en los que está matriculado cada Alumno (**JTableMatriculas**). Al seleccionar un alumno del JTableAlumnos, se actualizara el JTable JTableMatriculas.
- Tendremos un JTable donde se mostrarán los Exámenes realizados por un Alumno en un Curso determinado (**JTableExamenes**). Al seleccionar una matrícula del JTableMatriculas, se actualizara el JTable JTableExamenes.
- Al seleccionar un examen, se mostrará en los JTextField correspondientes la fecha y la nota que se obtuvo en ese examen.

TAREAS A REALIZAR:

1. La aplicación se realizará utilizando MVC.
2. Al iniciar la aplicación ser realizará la carga de datos de los JTable de Alumnos y Cursos utilizando Statement.
3. Al presionar el botón de **Matricular Alumno en Curso** se ejecutará un procedimiento almacenado en Oracle que generará un registro en la tabla Matriculas para el alumno seleccionado en el Jtable de alumnos y el curso seleccionado en el Jtable de Cursos y tantos exámenes como indique el curso.
   Hay que controlar los posibles errores de la ejecución del procedimiento.
4. Al presionar el botón **Actualizar** se ejecutará una sentencia PreparedStatement que actualizará la fecha y nota en que un alumno realizó del examen. Para ello debemos tener un registro seleccionado en el JTable **JTableExamenes.**
   Crear trigger que actualice la nota media de la matrícula.
5. Al presionar el botón **Boletin JSON** se generará un fichero de texto con formato JSON donde se mostrarán los datos de los exámenes de la matricula seleccionada en su JTable.
6. Al presionar el botón **Listado Matriculas XML** se generará un fichero de texto con formato XML donde se mostrarán los datos de los alumnos con los cursos en los que está matriculado cada uno de ellos.
7. El nombre del proyecto será AD_Tema02_Practica01_Apellido1_Nombre del alumno/a.

¿Eso significa que hay que comprobar el contenido de la vista antes de matricular un alumno?

ENTREGAR AL PROFESOR: CARPETA COMPRIMIDA DEL PROYECTO, DONDE SE ALMACENERA EL PROYECTO JAVA, UN EJECUTABLE JAR, EL FICHERO SQL CON LOS CAMBIOS REALIZADOS Y UN FICHERO DE TEXTO CON LAS OBSERVACIONES PERTINENTES. TODOS LOS FICHEROS UTILIZADOS EN EL PROYECTO DEBEN TENER RUTA RELATIVA

```
CREATE USER AD_TEMA02 IDENTIFIED BY AD_TEMA02;
GRANT DBA TO AD_TEMA02;
--Desde el usuario/esquema AD_TEMA02
CREATE TABLE ALUMNOS(
   cCodAlu VARCHAR2(6) CONSTRAINT PK_ALUMNOS PRIMARY KEY,
   cNomAlu VARCHAR2(100) NOT NULL
);

CREATE TABLE CURSOS(
   cCodCurso VARCHAR2(6) CONSTRAINT PK_CURSOS PRIMARY KEY,
   cNomCurso VARCHAR2(100) NOT NULL,
   nNumExa   NUMBER(3) DEFAULT 1 NOT NULL
);

CREATE TABLE MATRICULAS(
   cCodAlu   VARCHAR2(6) NOT NULL,
   cCodCurso VARCHAR2(6) NOT NULL,
   nNotaMedia   NUMBER(3) DEFAULT 0 NOT NULL,
   CONSTRAINT PK_MATRICULAS PRIMARY KEY (cCodAlu,cCodCurso)
);
```

```
ALTER TABLE MATRICULAS ADD CONSTRAINT FK_MATRICULAS_ALUMNO FOREIGN KEY (cCodAlu)
REFERENCES ALUMNOS(cCodAlu);

ALTER TABLE MATRICULAS ADD CONSTRAINT FK_MATRICULAS_CURSOS FOREIGN KEY (cCodCurso)
REFERENCES CURSOS(cCodCurso);

CREATE TABLE EXAMENES(
   cCodAlu   VARCHAR2(6) NOT NULL,
   cCodCurso VARCHAR2(6) NOT NULL,
   nNumExam  NUMBER(3) DEFAULT 1 NOT NULL,
   dFecExam  DATE,
   nNotaExam NUMBER(6,2) DEFAULT 0 NOT NULL,
   CONSTRAINT PK_EXAMENES PRIMARY KEY (cCodAlu,cCodCurso,nNumExam)
);

ALTER TABLE EXAMENES ADD CONSTRAINT FK_EXAMENES_MATR FOREIGN KEY (cCodAlu,cCodCurso)
REFERENCES MATRICULAS(cCodAlu,cCodCurso);

CREATE OR REPLACE TRIGGER tr_Examenes_Upd
AFTER UPDATE
ON EXAMENES
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('');
END;

INSERT INTO ALUMNOS VALUES ('001','Antonio');
INSERT INTO ALUMNOS VALUES ('002','Maria');

INSERT INTO CURSOS VALUES ('I001','Ingles Basico',5);
INSERT INTO CURSOS VALUES ('I002','Ingles Intermedio',8);
INSERT INTO CURSOS VALUES ('I003','Ingles Avanzado',10);
INSERT INTO CURSOS VALUES ('F001','Frances Basico',3);
INSERT INTO CURSOS VALUES ('C002','Chino Intermedio',9);
COMMIT;

CREATE OR REPLACE PROCEDURE sp_AltaMatricula (xcCodAlu VARCHAR2, xcCodCurso VARCHAR2, xError OUT
NUMBER)
IS  xNR NUMBER;
BEGIN
EXCEPTION
  WHEN OTHERS THEN xError := -1;
END;

SET SERVEROUTPUT ON;
DECLARE xError NUMBER;
BEGIN
  sp_AltaMatricula('001','I001',xError);
  DBMS_OUTPUT.PUT_LINE(xError);
END;
```

PRACTICA 02.

La empresa de limpiezas LIMPIADAM nos pide dos aplicaciones Java para llevar el control de los fichajes de los empleados en un edificio de oficinas y la generación de nóminas de los empleados. La primera será para el gerente de la empresa y podrá acceder a todas las opciones. La segunda solo mostrará una pantalla para que el empleado fiche al principio y fin de la realización de un trabajo.

- Cada vez que un empleado entra a realizar un trabajo, se genera un fichaje con la fecha y hora de entrada y cuando termina de ese trabajo se modifica el fichaje anterior incluyendo una fecha de salida; de tal forma, que un empleado tiene 0 o 1 fichaje con fecha de salida a nulo.

- Un empleado puede realizar varios fichajes al día, pero solo puede tener como máximo uno abierto (sin fecha de salida).

- Cada empleado tiene asignado unas horas mínimas al día. Eso significa, que si supera esas horas mínimas diarias, el resto de horas se consideran horas extras. Por ejemplo, si un empleado tiene asignadas 4 horas mínimas al día, si trabaja 4 o menos horas, cada hora se le abona a un precio/hora; pero si trabaja más de 4 horas (por ejemplo 6), las primeras 4 se le abonan a un precio/hora y el resto (en este caso 2) se le abonan como horas extras.

- Si un empleado empieza un trabajo en un día festivo, todas las horas de ese trabajo se consideran como horas extras.

- Cada vez que en un fichaje se cambia la fecha de salida, mediante TRIGGERs se genera o modifica (en caso de que exista) un registro de horas del empleado en esa fecha. La aplicación del gerente podrá en cualquier momento cambiar las fechas de entrada y/o salida de cualquier fichaje.

- Tendremos un procedimiento almacenado al que se pasará como parámetro de entrada una fecha cuya misión será la creación de las nóminas de los empleados en el mes/año de la fecha que se pasa como parámetro de entrada. Si en ese mes/año el empleado ya tiene nómina, esta será modificada. Este procedimiento almacenado devolverá como parámetro de salida el total de las nóminas.

- Se podrán hacer todas las consultas (similares a la pantalla que se muestra) que permitan las tablas/campos almacenados en la tabla CAMPOS, teniendo en cuenta que :

  - ✓ Los campos numéricos utilizaran los operadores: > , < e =.
  - ✓ Los campos de cadenas de caracteres el operador LIKE (el programa añadirá automáticamente el comodín % tanto al principio como al final).
  - ✓ Los campos de tipo fecha utiliza el operador BETWEEN.
  - ✓ Opcionalmente el alumno/a podrá hacer consultas complejas con AND u OR.

- El alumno/a podrá diseñar la aplicación a su elección, SIEMPRE Y CUANDO, el usuario pueda realizar cualquier operación que sea LOGICA en alguna parte de la aplicación.
- El alumno/a podrá añadir otros procedimientos almacenados / triggers o vistas que estime oportunos.

```
CREATE USER AD_TEMA02_FICHAJES IDENTIFIED BY AD_TEMA02_FICHAJES;
GRANT DBA TO AD_TEMA02_FICHAJES;
COMMIT;
-- Crear una conexion para AD_TEMA02_FICHAJES,
-- conectarse y ejecutar el resto del script

CREATE TABLE Empleados (
   dni varchar2(10) CONSTRAINT pk_empleados PRIMARY KEY,
   nombre varchar2(100) NOT NULL,
   – - AÑADIR RESTO DE CAMPOS DEL EMPLEADO
   HorasMin number(12,2),
   precioHora number(12,2),
   precioHoraE number(12,2)
);

CREATE TABLE Fichajes (
   idFichaje number(10) CONSTRAINT pk_fichajes PRIMARY KEY,
   dni varchar2(10) NOT NULL,
   FecHoraIni date NOT NULL, --dd/mm/yyyy hh24:mi
   FecHoraFin date NOT NULL, --dd/mm/yyyy hh24:mi
   CONSTRAINT fk_Fichajes_Empl FOREIGN KEY (dni) REFERENCES Empleados (dni),
   CONSTRAINT ck_Horario CHECK (FecHoraIni <= FecHoraFin)
);

CREATE TABLE HorasDiarias (
   Fecha date NOT NULL, -- dd/mm/yyyy
   dni varchar2(10) NOT NULL,
   Horas number(12,2),
   HorasE number(12,2),
   CONSTRAINT pk_Horas PRIMARY KEY(Fecha,dni),
   CONSTRAINT fk_Horas_Empl FOREIGN KEY (dni) REFERENCES Empleados (dni)
);

CREATE TABLE Festivos (
    Fecha date NOT NULL CONSTRAINT pk_Festivos PRIMARY KEY
);

CREATE TABLE Nominas (
   Anio number(4) NOT NULL,
   Mes number(2) NOT NULL,
   dni varchar2(10) NOT NULL,
   SueldoH number(12,2), --sueldo horas
   SueldoHE number(12,2), –-sueldo horas extras
   CONSTRAINT pk_Nominas PRIMARY KEY(Anio,Mes,Dni),
   CONSTRAINT fk_Nominas_Empl FOREIGN KEY (dni) REFERENCES Empleados (dni)
);

CREATE TABLE Campos (
   Tabla varchar2(20) NOT NULL,
   Campo varchar2(20) NOT NULL,
   Descripcion varchar2(50) NOT NULL,
   CONSTRAINT pk_Campos PRIMARY KEY(Tabla,Campo)
);

INSERT INTO CAMPOS VALUES ('Empleados','dni','Dni del Empleado');
INSERT INTO CAMPOS VALUES ('Empleados','HorasMin','Horas Diarias Minimas del Empleado');
INSERT INTO CAMPOS VALUES ('Nominas','Anio','Año de la nomina');
…
```