ACCESO A DATOS
PARTE IMPARTIDA POR EKAITZ MARTINEZ

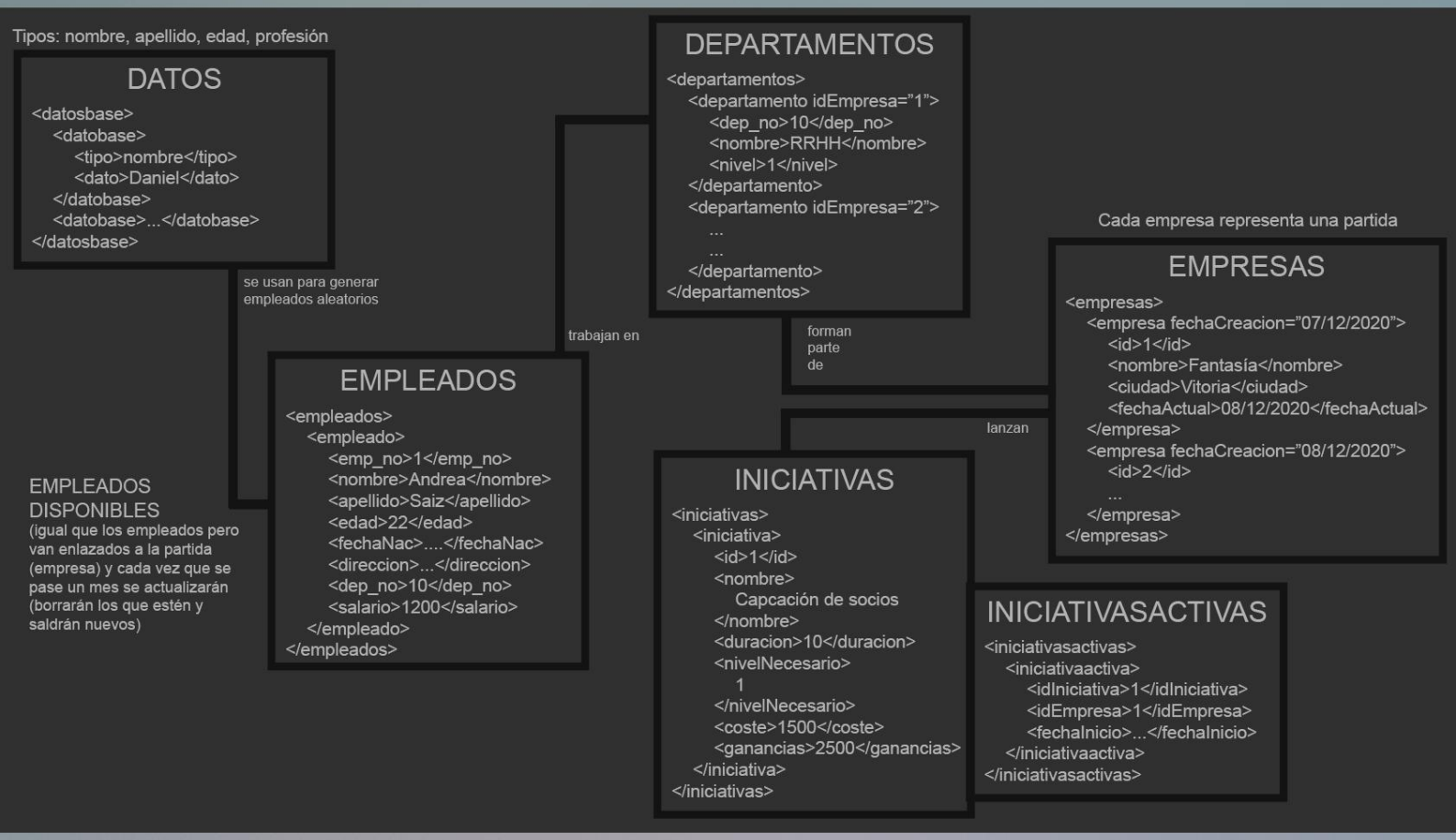
PROYECTO FINAL

0. Idea original del programa
1. Funcionamiento del programa
2. Algunos ejemplos de Querys
3. Algunos métodos de la aplicación
4. Logs y Querys almacenados
5. Otros

Daniel Tamargo

0. Idea original del programa

He basado la idea en el siguiente diagrama:



Se trataría de un juego donde tu objetivo es reventar el mundo haciendo la empresa más exitosa. Es muy fácil inflarse a dinero y la finalidad es hacer sentir al jugador que es un empresario de éxito.

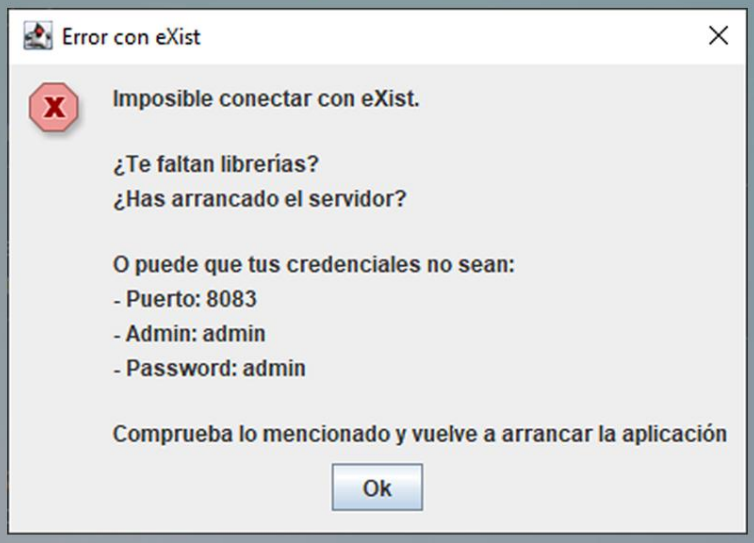
Al crear una empresa estás creando una partida, donde tendrás tus departamentos y podrás contratar EmpleadosDisponibles que pasarán a ser EmpleadosContratados y darán utilidad al departamento.

Ventas, Marketing y Estudio de Mercado generarán dinero mientras que RRHH hará que puedas contratar más y más gente y Salud hará que tus empleados no enfermen (realmente salud no hace nada, es una empresa de éxito sin enfermos)

Daniel Tamargo

1. Funcionamiento del programa

La aplicación se arranca desde el archivo Main.



Una vez arrancas la aplicación, esta comprueba que puede conectarse con eXist, si no puede, te lo notifica y te muestra un mensaje con posibles errores, finalizando después la aplicación sin que salte un error que la rompa.

Si la aplicación se puede conectar a eXist, lo primero que hará será comprobar que la colección eXiste (perdón por el chiste malo), si no existe, generará todos los datos base en ficheros binarios, luego los leerá y usará para generar ficheros xml a través de xStream, y después, añadirá esos ficheros xml generados a la colección, todo de forma automática y en un instante, sin que tengas que hacer nada. Podrás ver el feedback de este proceso en la consola.

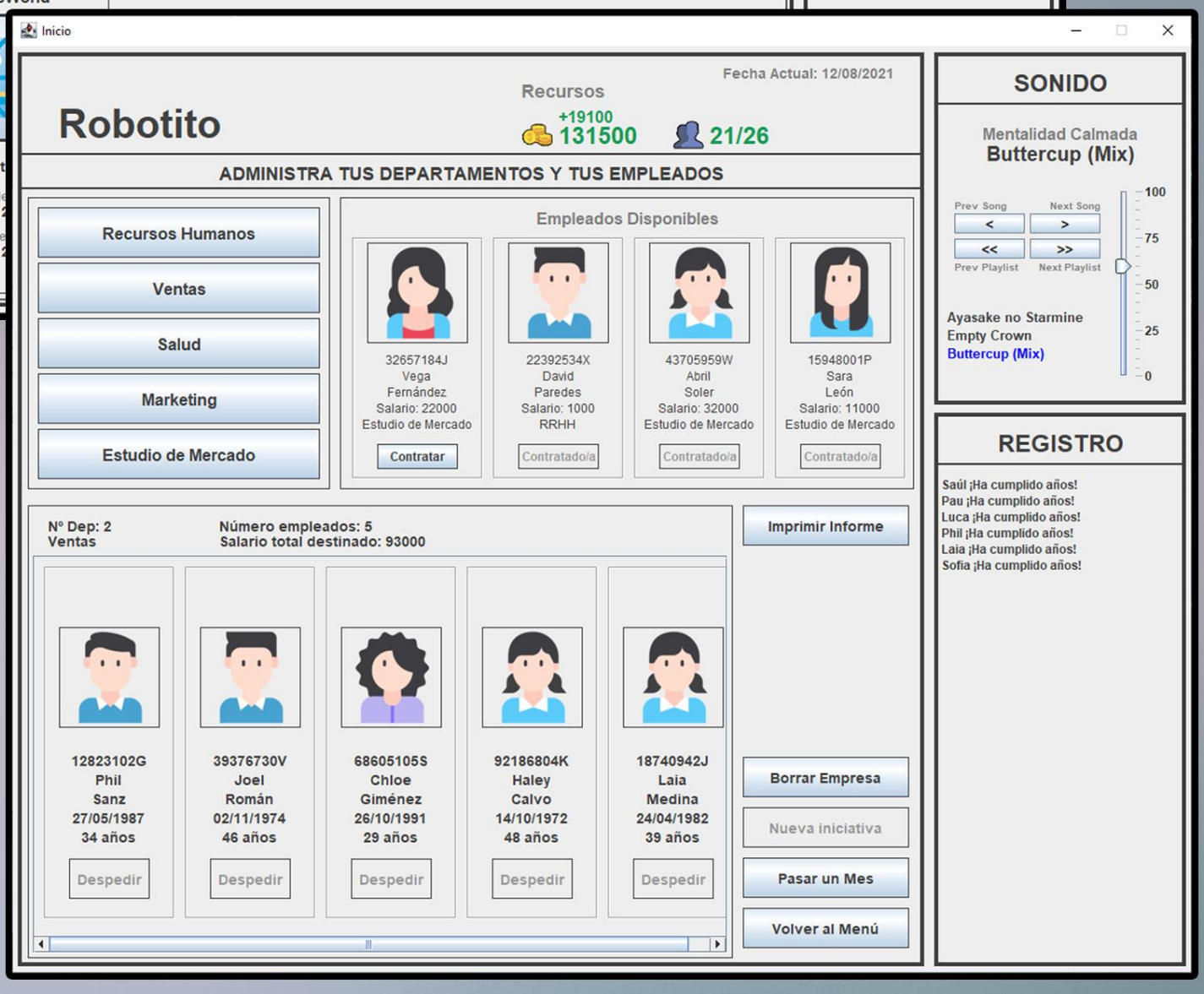
Si ya has arrancado el programa, ya estás dentro del juego, verás una ventana donde puedes crear una empresa o seleccionar una de las existentes. Cada vez que generes una empresa esta se guardará en el fichero empresas.xml, que a su vez generará los 5 departamentos que también se guardarán en departamentos.xml y además generarás 2 empleados que son tus empleados iniciales que se guardarán en empleadosContratados.xml

Cuando empiezas o cada vez que pasas un mes, se generan empleadosDisponibles que se guardan, y una vez los contratas o pasas de mes, se borran de empleadosDisponibles.xml y se generan otros nuevos.

Daniel Tamargo

1. Funcionamiento del programa

Veamos algunas imágenes.



2. Algunos ejemplos de Querys

```
private static boolean comprobarEmpresa(int id, Collection col) {...}

private static boolean comprobarDepartamento(int depNo, Collection col) {...}

private static boolean comprobarEmpleadoContratado(int empNo, Collection col) {...}
```

```
//...
public static boolean borrarEmpresa(int id) {...}

// BORRAR LAS INICIATIVAS DE UNA EMPRESA
private static void borrarIniciativasActivasEmpresa(int idEmpresa, XPathQueryService servicio) throws XMLDBException {...}

// BORRAR LOS EMPLEADOS DISPONIBLES DE UNA EMPRESA
public static void borrarEmpleadosDisponiblesEmpresa(int idEmpresa, XPathQueryService servicio) throws XMLDBException {...}

// BORRAR EMPLEADO DISPONIBLE DIRECTAMENTE
public static void borrarEmpleadoDisponible(int idEmpleado, XPathQueryService servicio) {...}

// BORRAR LOS DEPARTAMENTOS DE UNA EMPRESA
```

```
//...
public static boolean editarEdadEmpleadoContratado(EmpleadoContratado empleado) {...}

// EDITAR NIVEL DEPARTAMENTO
public static boolean editarNivelDepartamento(Departamento departamento) {...}

// EDITAR EMPRESA ENTERA (cambiar nombre / ciu
public static boolean editarEmpresa(Empresa em

// EDITAR EMPRESA SOLO LA FECHA
public static boolean editarFechaEmpresa(Empre
```

```
//...
public static boolean insertarEmpleadoContratado(EmpleadoContratado empleado) {...}

// INSERTAR EMPLEADO DISPONIBLE
public static boolean insertarEmpleadoDisponible(EmpleadoDisponible empleado) {...}

// INSERTAR DEPARTAMENTO
```

```
//...
public static ArrayList<Empresa> recogerEmpresas() {...}

public static ArrayList<Dato> recogerDatos() {...}

public static ArrayList<Iniciativa> recogerIniciativas() {...}

public static ArrayList<IniciativaActiva> recogerIniciativasActivas() {...}

public static ArrayList<Departamento> recogerDepartamentos

public static ArrayList<EmpleadoContratado> recogerEmplead

public static ArrayList<EmpleadoDisponible> recogerEmplead
```

```
public static int idEmpleadoDisponibleMasAlto() {...}
public static int idEmpresaMasAlto() {...}
public static int depNoDepartamentoMasAlto() {...}
public static int empNoEmpleadoContratadoMasAlto() {...}
public static int numDepartamentos() {...}
public static int numEmpleadosDepartamento(int depNo) {...}
public static long salarioTotalDepartamento(int depNo) {...}
```

```
String query = "" +
        "for $id in /empleadosDisponibles/max(empleadoDisponible/id) return $id";
```

```
String query = "" +
        "for $empr in /empresas/empresa " +
        " let $id:=$empr/id, " +
        " $salario:=$empr/salarioDisponible, " +
        " $gastos:=$empr/gastosMensuales, " +
        " $ganancias:=$empr/gananciasMensuales, " +
        " $nombre:=$empr/nombre, " +
        " $ciudad:=$empr/ciudad, " +
        " $logo:=$empr/logo, " +
        " $fechaActual:=$empr/fechaActual, " +
        " $fechaCreacion:=$empr/fechaCreacion " +
        " return concat($id, '|', $salario, '|', $gastos, '|', $ganan
        "$ciudad, '|', $logo, '|', $fechaActual, '|', $fechaCreacion)"
```

```
return "update insert " +
        "<empresa>" +
            "<id>" + id + "</id>" +
            "<salarioDisponible>" + salarioDisponible + "</salarioDisponible>" +
            "<gastosMensuales>" + gastosMensuales + "</gastosMensuales>" +
            "<gananciasMensuales>" + gananciasMensuales + "</gananciasMensuales>" +
            "<nombre>" + nombre + "</nombre>" +
            "<ciudad>" + ciudad + "</ciudad>" +
            "<logo>" + logo + "</logo>" +
            "<fechaActual>" + fechaActualStr + "</fechaActual>" +
            "<fechaCreacion>" + fechaCreacionStr + "</fechaCreacion>" +
        "</empresa>" +
        " into /empresas";
```

Daniel Tamargo

3. Algunos métodos de la aplicación

```java
private boolean comprobarDatos(JTextField nombre, JTextField ciudad) {
    Pattern patronNombre = Pattern.compile("[a-z A-Z0-9'!¡¿?]{3,16}");
    Pattern patronCiudad = Pattern.compile("[a-zA-Z]{3,16}");
    if (patronNombre.matcher(nombre.getText()).matches() &&
    patronCiudad.matcher(ciudad.getText()).matches()) {
        return true;
    } else {
        mostrarJOptionPane( titulo: "Datos no válidos",  mensaje: """
                EH! Introduce unos datos válidos!

                Nombre: de 3 a 16 caracteres y/o números
                Ciudad: de 3 a 16 caracteres""",  tipo: 0);
        return false;
    }
}
```

```java
private void pasarMes() {
    mesPasado = true;

    try {
        Coleccion.borrarEmpleadosDisponiblesEmpresa(empresa.getId(),  servicio: null);
    } catch (XMLDBException ignored) {}

    empresa.setFechaActual(empresa.getFechaActual().plusMonths(1));
    empresa.setSalarioDisponible(empresa.getSalarioDisponible() + (empresa.getGananciasMensuales() - empresa.getGastosMensuales()));
    Coleccion.editarEmpresa(empresa);

    for (EmpleadoContratado empleadoContratado : empleadosEmpresa) {
        if (empleadoContratado.haCumplidoAnyos(empresa.getFechaActual())) {
            lineasRegistro.add( index: 0,  element: empleadoContratado.getNombre() + " ¡Ha cumplido años!");
            Coleccion.editarEdadEmpleadoContratado(empleadoContratado);
        }
    }

    volcarDatosTextPaneRegistro();

    cargarVentanaPartida();
```

```java
private void generarEmpleadosDisponibles() {
    if (mesPasado) {
        for (int i = 0; i < 4; i++) {
            EmpleadoDisponible emp;
            String nombre;
            String avatar;
            if (new Random().nextBoolean()) {
                nombre = datosNombresMasc.get(new Random().nextInt(datosNombresMasc.size())).getDato();
                avatar = datosAvataresMasc.get(new Random().nextInt(datosAvataresMasc.size())).getDato();
            } else {
                nombre = datosNombresFem.get(new Random().nextInt(datosNombresFem.size())).getDato();
                avatar = datosAvataresFem.get(new Random().nextInt(datosAvataresFem.size())).getDato();
            }

            emp = new EmpleadoDisponible( id: Coleccion.idEmpleadoDisponibleMasAlto() + 1,
                    datosDnis.get(new Random().nextInt(datosDnis.size())).getDato(),
                    nombre,
                    datosApellidos.get(new Random().nextInt(datosApellidos.size())).getDato(),
                    LocalDate.now().minusMonths(new Random().nextInt( bound: 12)).minusDays(new Random().nextInt( bound: 31)).minusYears(new Random().nextInt( bound
                     salario: (new Random().nextInt( bound: 32) + 1) * 1000,
                    avatar,
                    datosProfesiones.get(new Random().nextInt(datosProfesiones.size())).getDato(),
                    empresa.getId(),
                    empresa.getFechaActual());
            Coleccion.insertarEmpleadoDisponible(emp);
            empleadosDisponiblesEmpresa.add(emp);
        }
        mesPasado = false;
    }
}
```

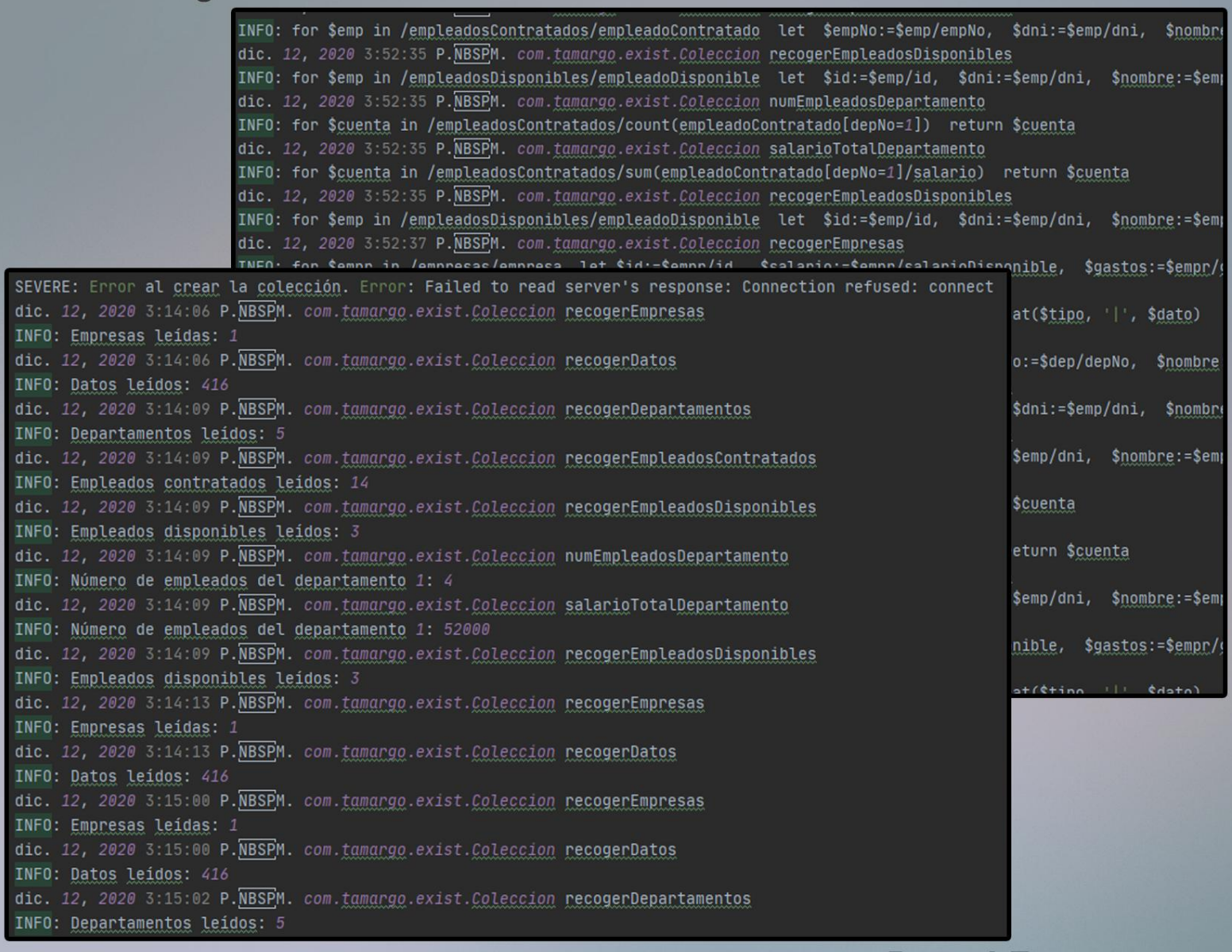
Daniel Tamargo

4. Logs y Querys almacenados

Los logs se almacenan en la ruta ./ficheros/logs y las querys realizadas se almacenan en la ruta ./ficheros/querys

Se pueden visualizar para obtener feedback del funcionamiento de la aplicación y de las querys realizadas.

Al utilizar la clase Logger para guardar los logs y las querys obtenemos una información extra valiosa: hora de la ejecución, nivel del log (lo definiremos nosotros pero generalmente SEVERE significará error y FINE, FINEST o INFO significará correcto o simplemente notificación) y desde qué clase y método ha sido generado.



Daniel Tamargo

5. Otros

Mi idea original del programa era mucho más elaborada y avanzada que el resultado pero por falta de tiempo y saturación de proyectos no he conseguido mi objetivo (por ejemplo, en el resultado no funciona correctamente el panel de ver los empleados contratados de un departamento porque el ScrollPane que genero por código se buggea, haciendo que el botón despedir no funcione correctamente, o que al deslizar hacia los lados se buggee y no muestre bien los empleados).

Cabe destacar que todos los empleados se generan con datos aleatorios, para esto usamos el fichero datos.xml donde tendremos almacenados todos esos datos a usar, con su tipo y su información.

El programa implementa un panel de sonido, donde podremos escuchar diferentes canciones de diferentes Playlists.
Estoy orgulloso de esta parte puesto que es un hilo que he creado y que a través de la clase Clip, toca los diferentes archivos de música almacenados. Me enorgullece haber podido conseguir que reproduzca diferentes Playlists sin dar errores y controlando en todo momento en qué canción está.

Tras realizar querys de todo tipo con XPath y XQuery, creo esta base de datos NoSQL es interesante y si te acostumbras a hacerlas y te mentalizas con cómo es el proceso y dónde suelen estar los fallos tontos más habituales, puede ser muy práctica y útil, pero requiere de cierta atención al detalle, donde si al final tejes una gran cadena de etiquetas, puede acabar siendo un infierno recoger, insertar o editar los datos correctamente, la clave con esta BBDD está en plantearlo lo mejor posible desde un inicio.