



Documento: Prueba de conocimientos.

Departamento: Recursos humanos.

Objetivos del documento.	1
Detalle del sistema a desarrollar.	1
Base de datos.....	2
Entregable.	2
ANEXO A. Esquema base de datos.....	3

Objetivos del documento.

Mediante el presente documento se detalla una prueba básica de programación en Java, con la que se pretende determinar los conocimientos que reúne el candidato/candidata al puesto de “Analista Programador en Java”. Los resultados serán vinculantes para la realización de una entrevista en la que hablaremos de otros puntos importantes para determinar la idoneidad del candidato/candidata al puesto.

En ningún caso este ejercicio, e incluso la no presentación del mismo, descarta o asegura ninguna opción que el candidato/candidata pueda tener de optar al puesto vacante.

El ejercicio, los requisitos y la documentación es escueta intencionadamente. Se pretende estimar la autonomía y creatividad del candidato, más allá de sus conocimientos técnicos del lenguaje y el entorno de desarrollo. No obstante, en rrhh@iten.es habrá respuestas a todas las preguntas.

Detalle del sistema a desarrollar.

El sistema a desarrollar consiste en un proyecto en Java. Para su desarrollo se utilizará el IDE Netbeans 6.7 de Sun Microsystems.

Modelo de negocio.

1.- Se pretende gestionar una base de datos de facturación, atendiendo a la estructura de base de datos definida en el Anexo A.

2.- Actualmente y atendiendo al modelo, los clientes de las facturas pueden ser particulares o empresas. No obstante, se desea dejar la posibilidad abierta a nuevos tipos de clientes en el futuro. El sistema será lo suficientemente consistente como para que la inclusión de un nuevo tipo de cliente no implique modificaciones al proceso de creación de facturas.

Datos a tener en cuenta en la implementación.

1.- **Clientes** debe declarar un método para chequear el documento fiscal (NIF, CIF, etc...), que deberá ser implementado en cada uno de los tipos de cliente. Devolverá un booleano que indicará si el documento asociado al cliente es correcto o no. No es importante si la implementación en las subclases se ajusta al mundo real o no.

2.- Para acceder al sistema se usará un nombre de usuario y contraseña, registrados en la tabla “Configuración”. Asimismo, cada vez que un usuario accede se registrará en dicha tabla la fecha del sistema.

Se valorará (aunque no es obligatorio).

1.- El uso de patrones de diseño (Singleton, Observer, etc...) allá donde sean de aplicación.

- 2.- La creación de un interfaz de usuario, usando cualquier medio (consola, Swing, etc...) que permita la interacción con el sistema.
- 3.- La construcción del sistema en inglés (nombres de clases, interfaces, propiedades y métodos, etc...)
- 4.- La documentación del sistema en inglés. Diagramas de uso, clases, entidad relación, documento técnico, etc...
- 5.- El uso de frameworks de desarrollo para persistencia, capa de presentación, etc...
- 6.- Cualquier otra optimización razonada que se le haga al sistema.

Base de datos.

La base de datos queda a elección del candidato/candidata, siendo recomendable, aunque no requerido, utilizar algún sistema ligero (SQLite, Hypersonic, etc...) por cuestiones de comodidad.

El esquema de la misma es detallado en el Anexo A.

Entregable.

Los proyectos serán enviados a la dirección de correo electrónico rrhh@iten.es, comprimidos en un fichero ZIP o RAR. Igualmente, cualquier duda o consulta acerca del ejercicio puede ser remitida a la dirección rrhh@iten.es, y será contestada lo antes posible.

ANEXO A. Esquema base de datos.

```
CREATE TABLE public.configuracion (
    fechaultimoacceso DATE NOT NULL,
    usuario VARCHAR(10) NOT NULL,
    clave VARCHAR(10) NOT NULL
);
CREATE TABLE public.clientes (
    iddecliente IDENTITY NOT NULL,
    cliente VARCHAR(30) NOT NULL,
    CONSTRAINT clientes_pk PRIMARY KEY (iddecliente)
);
CREATE TABLE public.facturas (
    iddefactura IDENTITY NOT NULL,
    fecha DATE NOT NULL,
    iddecliente INTEGER NOT NULL,
    CONSTRAINT facturas_pk PRIMARY KEY (iddefactura)
);
CREATE TABLE public.detallefacturas (
    iddedetalle IDENTITY NOT NULL,
    iddefactura INTEGER NOT NULL,
    cantidad DECIMAL(10,2) NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    concepto VARCHAR(25) NOT NULL,
    CONSTRAINT detallefacturas_pk PRIMARY KEY
(iddedetalle)
);
CREATE TABLE public.particulares (
    iddecliente INTEGER NOT NULL,
    cif VARCHAR(10) NOT NULL,
    CONSTRAINT particulares_pk PRIMARY KEY (iddecliente)
);
CREATE TABLE public.empresas (
    iddecliente INTEGER NOT NULL,
    nombrecomercial VARCHAR(30) NOT NULL,
    cif VARCHAR(10) NOT NULL,
    CONSTRAINT empresas_pk PRIMARY KEY (iddecliente)
);
ALTER TABLE public.empresas ADD CONSTRAINT clientes_empresas_fk
FOREIGN KEY (iddecliente)
REFERENCES public.clientes (iddecliente)
ON DELETE CASCADE
ON UPDATE CASCADE;
ALTER TABLE public.particulares ADD CONSTRAINT
clientes_particulares_fk
FOREIGN KEY (iddecliente)
REFERENCES public.clientes (iddecliente)
ON DELETE CASCADE
ON UPDATE CASCADE;
ALTER TABLE public.facturas ADD CONSTRAINT clientes_facturas_fk
FOREIGN KEY (iddecliente)
REFERENCES public.clientes (iddecliente)
ON DELETE NO ACTION
ON UPDATE NO ACTION;
```

```
ALTER TABLE public.detallefacturas ADD CONSTRAINT  
facturas_detallefacturas_fk  
FOREIGN KEY (iddefactura)  
REFERENCES public.facturas (iddefactura)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION;
```