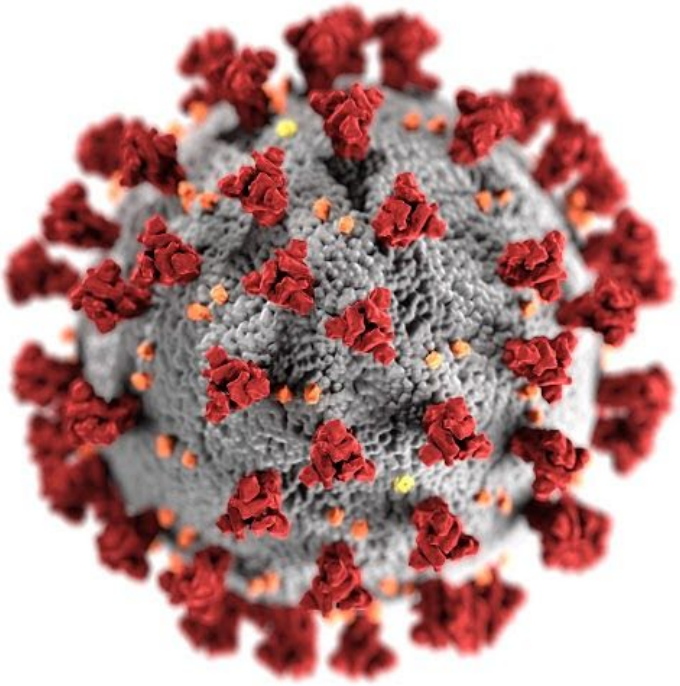
PRÁCTICA 3:

Diseño, desarrollo e implantación de un sistema de información Web: diseño e implementación de la capa de persistencia de datos



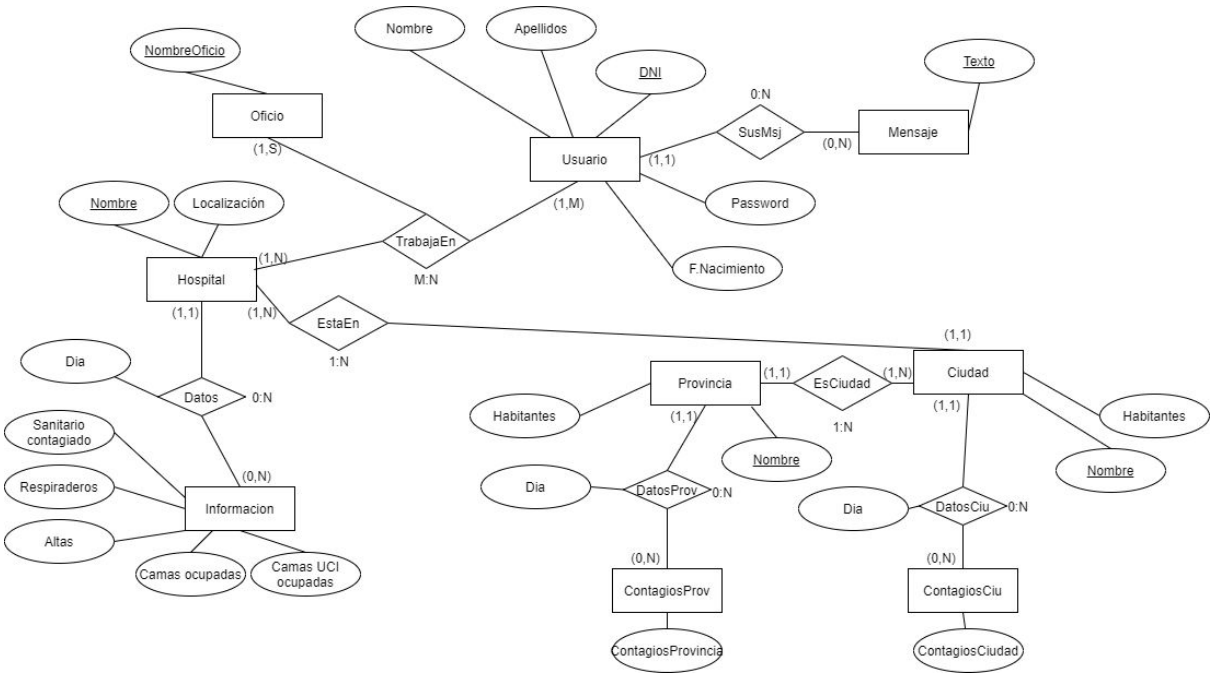
Andrés María Gilgado Barrachina 777638
Fernando Ballester Tolón 780205
Diego Guillén Serrano 779363

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

ÍNDICE

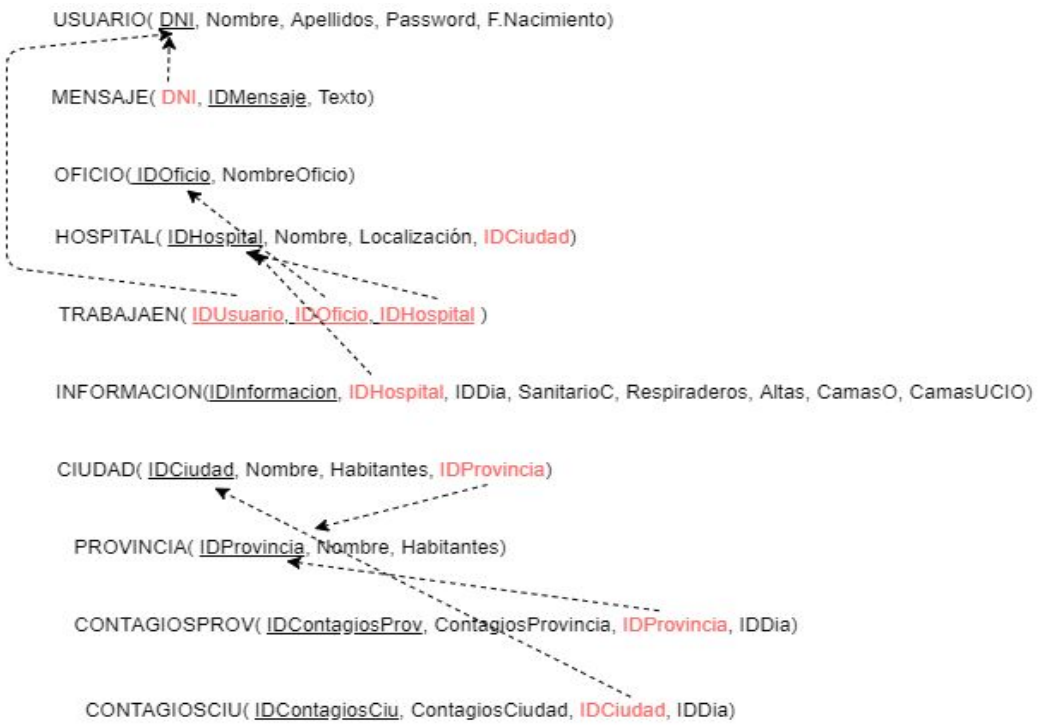
Primera semana: Primera parte

Modelo entidad-relación



En este modelo hemos añadido la información de la que vamos a disponer.

Modelo relacional



Hemos realizado el modelo relacional por medio de una normalización del modelo entidad-relación. En él hemos añadido distintas ID para diferenciar sin ninguna ambigüedad a cada una de las entidades, aunque asumimos que ninguna ciudad ni provincia va a llamarse igual que otra, es preferible utilizar estos IDs.

Creación de tablas

```
CREATE SCHEMA bbdd AUTHORIZATION postgres;
CREATE TABLE bbdd.provincia (
    id int4 NOT NULL,
    nombre varchar(100) NOT NULL,
    habitantes int4 NULL,
    CONSTRAINT provincia_pk PRIMARY KEY (id)
);


CREATE TABLE bbdd.ciudad (
    id int4 NOT NULL,
    ciudad varchar(100) NOT NULL,
    habitantes int4 NULL,
    provinciaid int4 NOT NULL,
    CONSTRAINT ciudad_pk PRIMARY KEY (id),
    CONSTRAINT ciudad_fk FOREIGN KEY (provinciaid) REFERENCES bbdd.provincia(id)
);


CREATE TABLE bbdd.hospital (
    id int4 NOT NULL,
    nombre varchar(100) NOT NULL,
    latitud varchar(30) NOT NULL,
    longitud varchar(30) NOT NULL,
    idciudad int4 NOT NULL,
    CONSTRAINT hospital_pk PRIMARY KEY (id),
    CONSTRAINT hospital_fk FOREIGN KEY (idciudad) REFERENCES bbdd.ciudad(id)
);

CREATE TABLE bbdd.informacion (
    id int4 NOT NULL,
    sanitariosc int4 NULL,
    camasuci int4 NULL,
    respiradores int4 NULL,
    altas int4 NULL,
    camasocu int4 NULL,
    idhospital int4 NOT NULL,
    dia date NOT NULL,
```

```
CREATE TABLE bbdd.usuarios (
    nombre varchar(100) NOT NULL,
    id int4 NOT NULL GENERATED ALWAYS AS IDENTITY,
    apellidos varchar(200) NOT NULL,
    "password" varchar(50) NOT NULL,
    "user" varchar(50) NOT NULL,
    CONSTRAINT usuarios_pk PRIMARY KEY (id)
);


CREATE TABLE bbdd.mensaje (
    id int4 NOT NULL,
    texto varchar(500) NOT NULL,
    iddni int4 NOT NULL,
    CONSTRAINT mensaje_pk PRIMARY KEY (id),
    CONSTRAINT mensaje_fk FOREIGN KEY (iddni) REFERENCES bbdd.usuarios(id)
);

CREATE TABLE bbdd.oficios (
    id int4 NOT NULL,
    nombre varchar(100) NOT NULL,
    CONSTRAINT oficios_pk PRIMARY KEY (id)
);

CREATE TABLE bbdd.trabajaen (
    idhospital int4 NOT NULL,
    idusuario int4 NOT NULL,
    idoficio int4 NOT NULL,
    CONSTRAINT trabajaen_pk PRIMARY KEY (idhospital, idusuario, idoficio),
    CONSTRAINT trabajaen_fk FOREIGN KEY (idusuario) REFERENCES bbdd.usuarios(id),
    CONSTRAINT trabajaen_fk_1 FOREIGN KEY (idhospital) REFERENCES bbdd.hospital(id),
    CONSTRAINT trabajaen_fk_2 FOREIGN KEY (idoficio) REFERENCES bbdd.oficios(id)
);
```

CONSTRAINT informacion_pk PRIMARY KEY (id), CONSTRAINT informacion_fk FOREIGN KEY (idhospital) REFERENCES bbdd.hospital(id));	

Segunda semana: Conexión

En esta segunda parte, nos hemos encargado de la conexión a la base de datos por medio de java, en este caso gracias al conector JDBC que hace posible la misma. Hemos barajado dos posibles opciones de conexión, la primera es una conexión por consulta y otra que es tener una pool de conexiones, en esta ocasión, como versión de prueba hemos elegido una conexión por consulta. Para comprobar el funcionamiento empezaremos con un login, de tal manera que realizando una consulta de cuántos usuarios con id propuesto en el formulario del html hay, de tal manera que si es igual a 1, será correcto y en caso contrario puede significar una inyección de código sql en la página o que simplemente no existe el usuario pedido.
Hemos optado que la búsqueda de si existe dicho usuario sea por id, ya que en la tabla es la clave primaria y esto quiere decir que no se va a repetir y no va a haber dos usuarios con el mismo id, en este caso el id es el número de identificación de cada persona del hospital que nos será facilitada por una persona de recursos humanos de cada hospital de Aragón.
En casos futuros, nuestra aplicación no solo permitirá el registro por medio de java con sql, sino que la gran mayoría de archivos multimedia serán referencias a nuestra base de datos, por no hablar de gráficos y datos diarios acerca del coronavirus, ya que estos serán actualizados diariamente por lo que debemos asegurar una consistencia en los datos y en el esquema entidad-relación.
Implementación llevada a cabo para los ficheros de la segunda parte:

- **Modificación fichero Server.xml:** Para habilitar una única conexión por vez de una página a la base de datos, hay que modificar este fichero poniendo el nombre de la base de datos correspondiente y su contraseña, además del driver que se va a utilizar que en nuestro caso es el JDBC. Y como el nombre, también es necesario la url para la conexión.

- **Modificación fichero context.xml:** Para habilitar una pool de conexiones a la base de datos hay que modificar este fichero, al cual hay que darle un alias público para que se haga posible la conexión.

- **Package es.unizar.sisinf.grp1.model:** En este paquete irán los ficheros que tienen que ver con las consultas, es decir, las clases que representan cada una de las

tablas creadas en la base de datos, en nuestro caso, hemos creado la clase que representa la tabla de usuarios y la consulta del login enlazada con la base de datos.

- **Package es.unizar.sisinf.grp1.db:** Este paquete alberga los ficheros que posibilitan la conexión de los ficheros java a la base de datos, gracias a las modificaciones en los ficheros Server.xml y context.xml.

Cierto es que estos ficheros, son los que realizan las consultas a las bases de datos, pero sigue faltando una conexión entre lo que es el código html y el java, por ello están los jsp que enlazan html y java. En el código de ejemplo que se proporciona, está el ejemplo para la conexión a un login de una página.

Recursos , herramientas Utilizadas y Dificultades

Para esta práctica hemos utilizado principalmente nuestros conocimientos sobre base de datos, cabe destacar que para la realización de nuestros modelos de Entidad - Relación y la normalización de este hemos utilizado la herramienta de draw.io, que nos facilitó bastante la realización . Hay que resaltar que en esta práctica hemos empezado a usar seriamente la herramienta Eclipse para la realización de las tablas y para realizar las conexiones JDBC.
Por último , hemos encontrado dificultades a la hora de la creación de tablas y las conexiones JDBC dado que al no estar muy familiarizados con el entorno Eclipse , al principio nos costó entender su funcionamiento.

Distribución de trabajo ,planificación y decisiones tomadas y horas de trabajo

En primer lugar, tenemos que destacar que para realizar esta práctica nos hemos citado en diferentes reuniones meet , para avanzar todos a un ritmo constante . Nuestra primera , fue unas horas antes de la clase práctica en la que dejamos terminado el modelo relacional y la normalización de este mismo. Nuestra segunda reunión tuvo lugar a la hora de prácticas, en esta reunión terminamos la creación de tablas y las adjuntamos en eclipse para dar forma a nuestro proyecto, en nuestra tercera reunión dejamos terminadas las conexiones JDBC y empezamos a trabajar la memoria, ya en nuestra última reunión dejamos terminada la práctica.
Por último, si hemos realizado 4 reuniones y cada reunión estuvimos aproximadamente dos horas trabajando , hemos determinado que estuvimos alrededor de 8 horas útiles para terminar esta práctica.