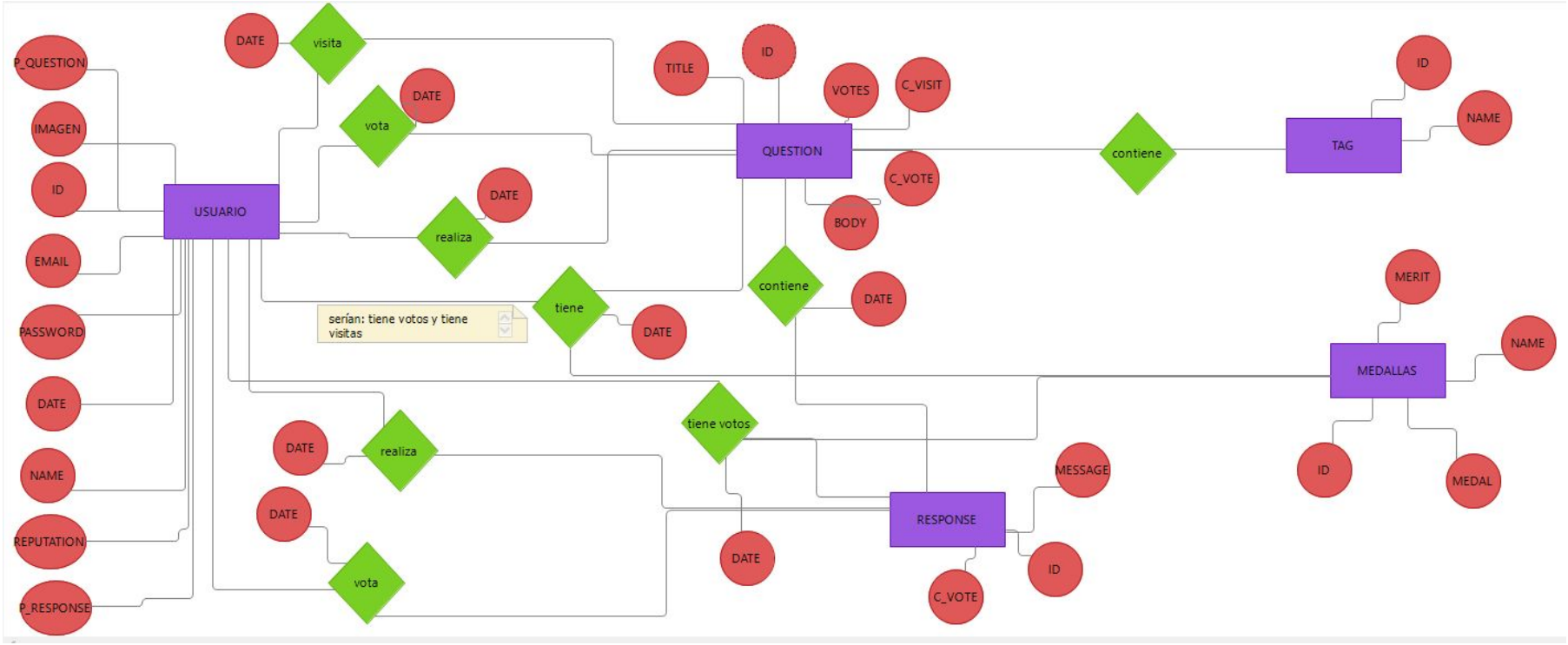
APLICACIONES WEB
-404-

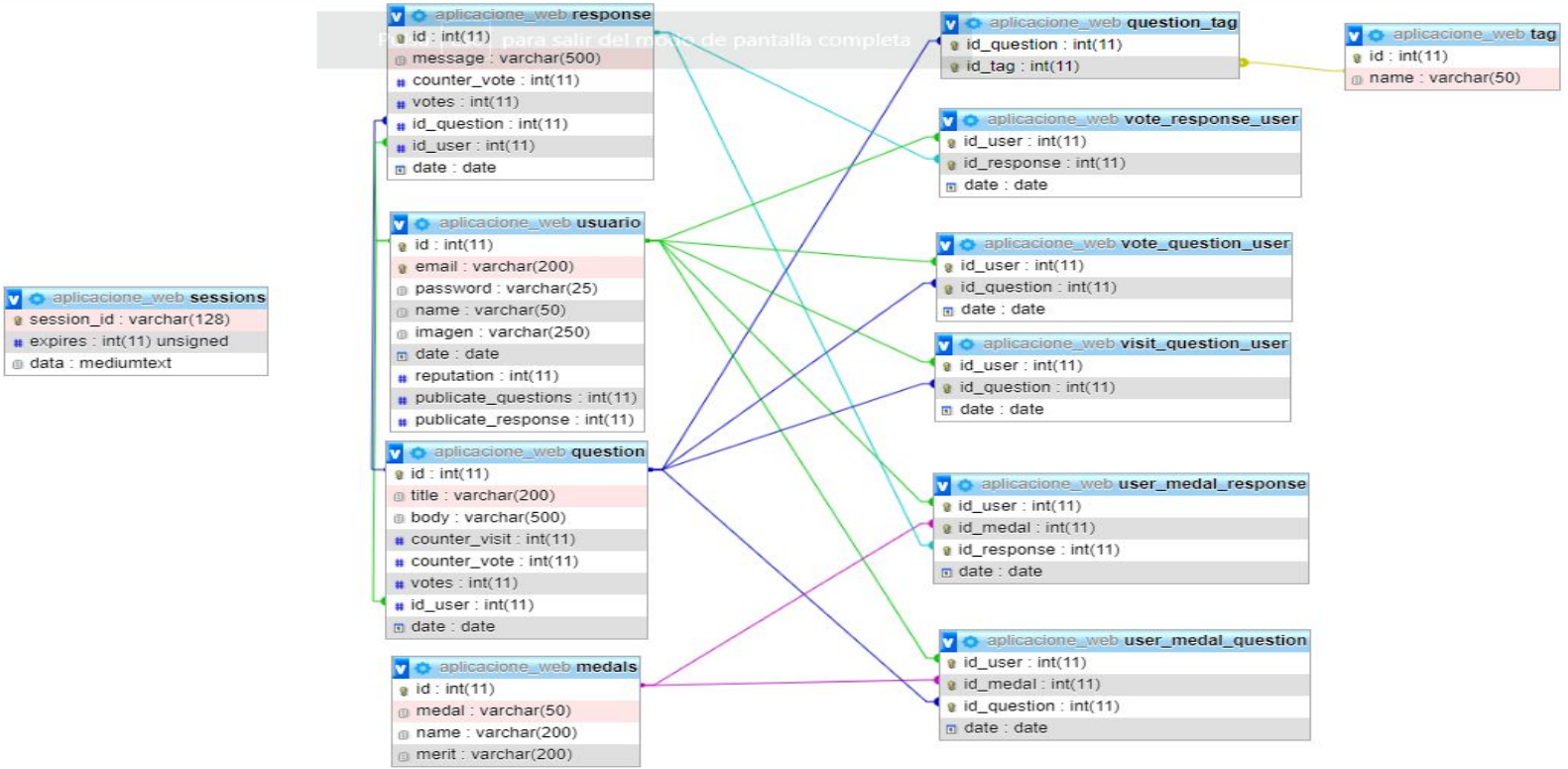
Daniela Nicoleta Boldureanu
Iker Burgoa Muñoz

- Modelo Entidad-Relación
- Modelo Relacional
- Estructura de la aplicación
- Paquetes
- Listado de rutas
- Restricción de acceso
- Gestión de errores 404/500

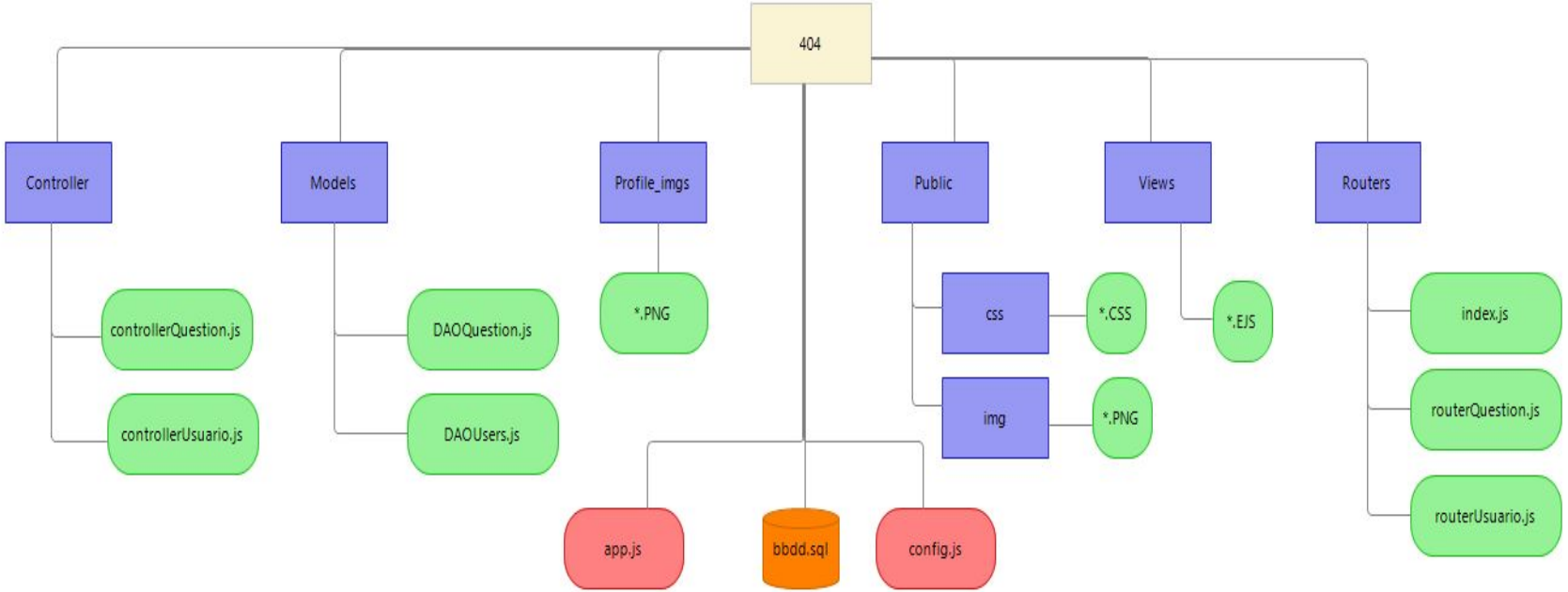
MODELO ENTIDAD-RELACIÓN



MODELO RELACIONAL



ESTRUCTURA DE LA APLICACIÓN



Listado de la rutas gestionadas por el servidor

index.js

- *“/”*
  get: renderiza la vista de login.

Listado de la rutas gestionadas por el servidor

routerUsuario.js “/usuarios/”

- login
  get: renderiza la vista de login.
  post: redirecciona a /usuarios/home una vez identificado correctamente al usuario.

- logout
  get: elimina los datos de la sesión, limpia las cookies y redirecciona a /.

- register
  get: renderiza la vista de register.
  post: En el caso de que se haya validado los campos del formulario y de que se haya creado el usuario correctamente nos redirecciona a /usuarios/home

Listado de la rutas gestionadas por el servidor

routerUsuario.js “/usuarios/”

- /

  get : renderiza la vista de search_users una vez obtenidos los usuarios a mostrar.

- home

  get: renderiza la vista de home con los datos del usuario logeado.

- searchUser

  get: renderiza la vista filter_users_name con los usuarios que coincidan en dicha busqueda.

- formQuestion

  get: renderiza la vista de home

Listado de la rutas gestionadas por el servidor

routerUsuario.js “/usuarios/”

- :id_user
  get : renderiza la vista de user_profile con los datos del usuario correspondiente al parámetro id_user.

- fotoId/:userId
  get: envía la imagen correspondiente del usuario identificada por el parámetro id.

- imagenUsuario
  get: envía la imagen que le corresponde al usuario que está logueado en la aplicación.

Listado de la rutas gestionadas por el servidor

routerQuestion.js “/questions/”

- /

  get : renderiza la vista de questions con las preguntas ordenadas cronológicamente a mostrar

- :id_question

  get: renderiza la vista de information_question de una determinada pregunta identificada por el parámetro id_question.

- searchText

  get: renderiza la vista de filter_question_text con las preguntas correspondientes a la búsqueda.

- searchTag

  get: renderiza la vista de filter_question_tag con las preguntas correspondientes al tag.

Listado de la rutas gestionadas por el servidor

routerQuestion.js “/questions/”

- sinRespuesta
  get : renderiza la vista de no_response_question con las preguntas sin respuesta a mostrar.

- likeQuestion/:id_like/:id_question
  get: redirecciona a la vista de /questions/:id_question una vez hayas reaccionado con like o dislike a la pregunta.

- like/:id_like/:id_response/:id_question
  get: redirecciona a la vista de /questions/:id_question una vez hayas reaccionado con like o dislike a la respuesta.

- formular
  get: renderiza la vista de form_question.

Listado de la rutas gestionadas por el servidor

routerQuestion.js “/questions/”

- formResponse
  post: redirecciona a la vista de /questions/:id_question una vez contestada la pregunta.

- formQuestion
  post: redirecciona a la vista /usuarios/home una vez creada la pregunta correctamente.

Restricción de acceso a las rutas para los usuarios no logueados.

- Se comprueba si el usuario está logueado con la variable request.session.currentUser. En caso de que el usuario esté logueado cogemos de la base de datos la información necesaria para guardarla en response.locals y en el caso de no estarlo, se redirige a la página de login.

- Se utiliza en las rutas que deberían de ser inaccesibles por un usuario que no esté logueado.

```
function comprobarUsuario(request, response, next) {

    if (request.session.currentUser) {
        daoU.getUserName(request.session.currentUser,
            function (err, res) {
                if (err) {
                    next(err);
                }
                else {

                    response.locals.userNombre = res.name;
                    response.locals.email = request.session.currentUser;
                    response.locals.id = res.id;
                    next();
                }
            });

    }
    else {
        response.redirect("/usuarios/login");
    }
}
```

Gestión de errores

error 404:

```
app.use(function(request,response,next){

        response.status(404).render(“404”); })
```

En el penúltimo lugar de la cadena de middlewares, se ha colocado el que gestiona los casos en los que una url no ha sido capturada por ningún mensaje anterior.

Gestión de errores

error 500:

```
app.use(function(error,request,response,next){

    response.status(500).render(“500”); })
```

Al final de la cadena de middlewares se ha colocado este, el cual muestra el error 500.

Listado de los paquetes utilizados

- body-parser: acceder a la información del cuerpo de una petición POST.
- cookie-parser: gestionar las cookies.
- ejs: poder usar plantillas en documentos HTML con marcadores especiales.
- express: creación de la aplicación web.
- express-mysql-session: almacenar la información de sesión en la base de datos.
- express-session: para usar el atributo session, el cual contiene los datos de sesión de un determinado usuario.
- multer: para analizar el contenido de los formularios con codificación multipart/form-data.
- mysql: conexión con la base de datos relacional.
- path:

GRACIAS