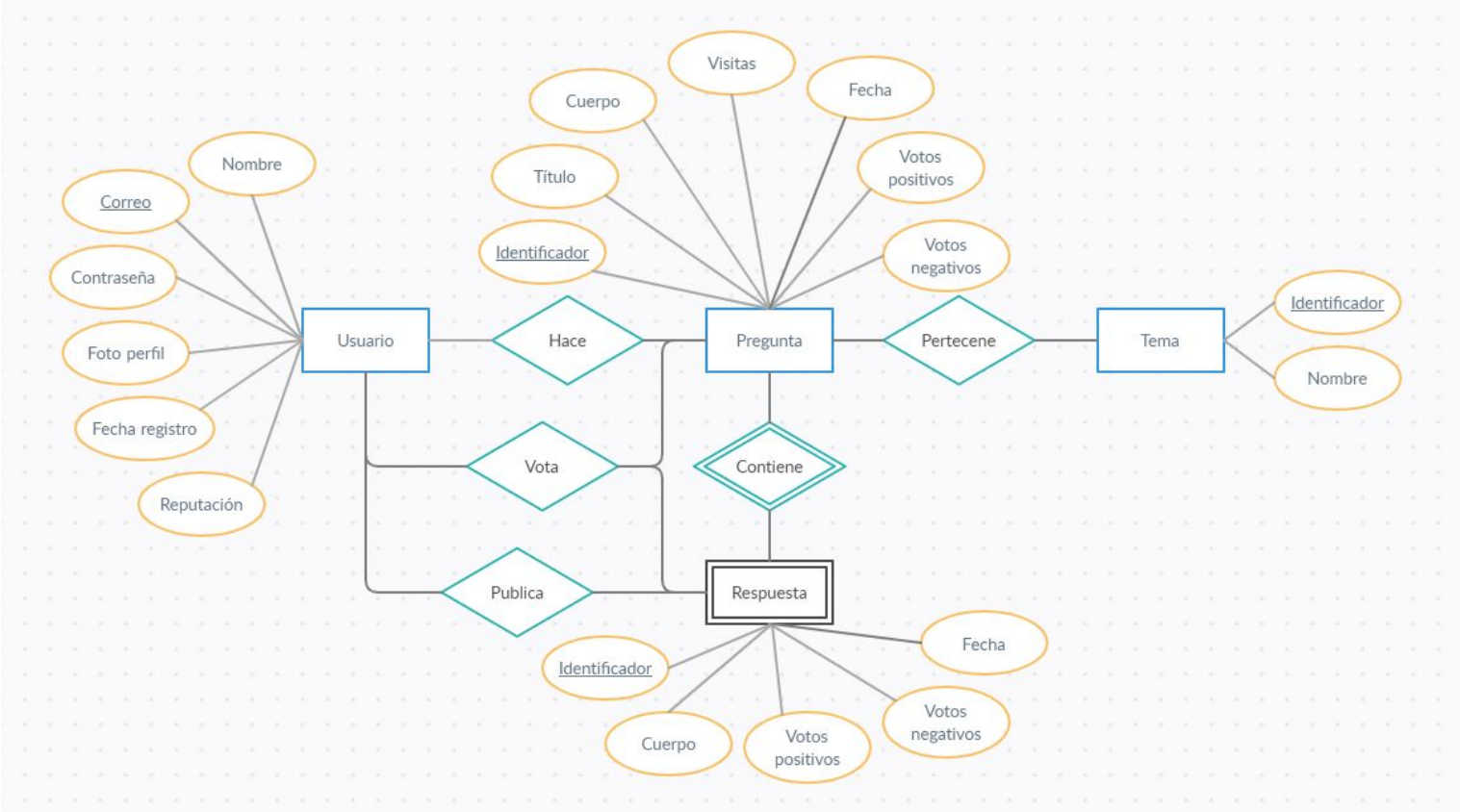
Presentación

Aplicación 404

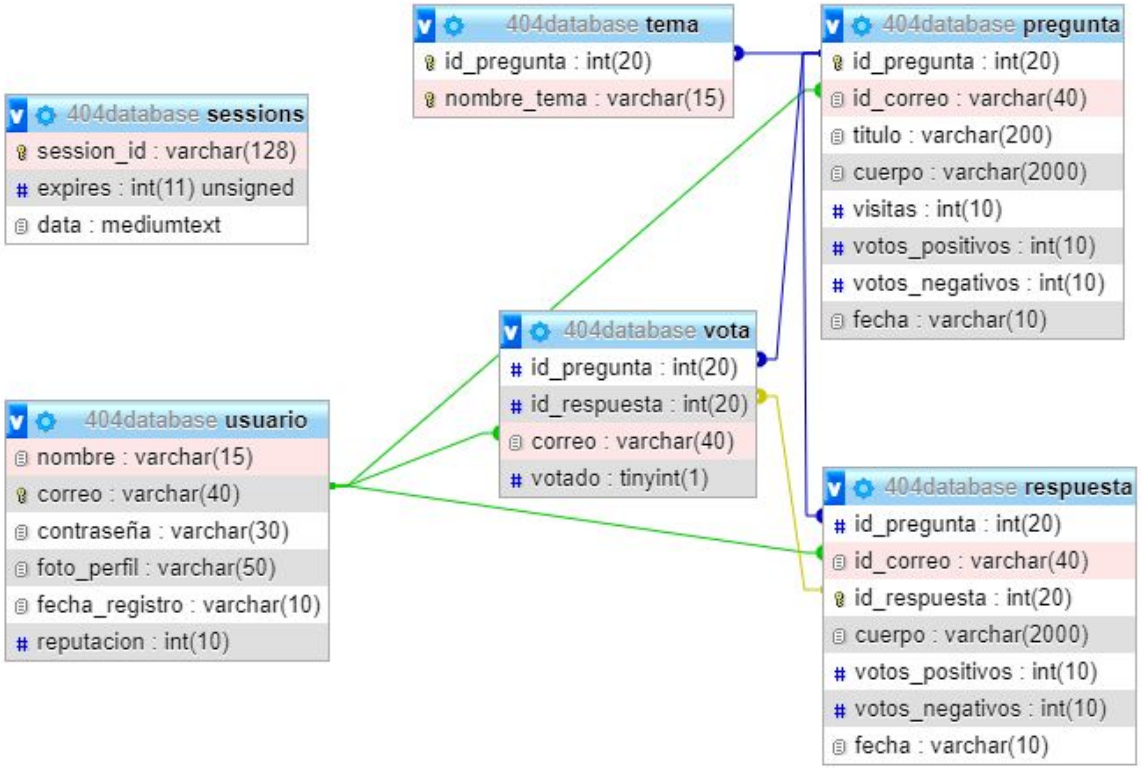
Grupo 19

Christian Esteban
Daniel Carrera

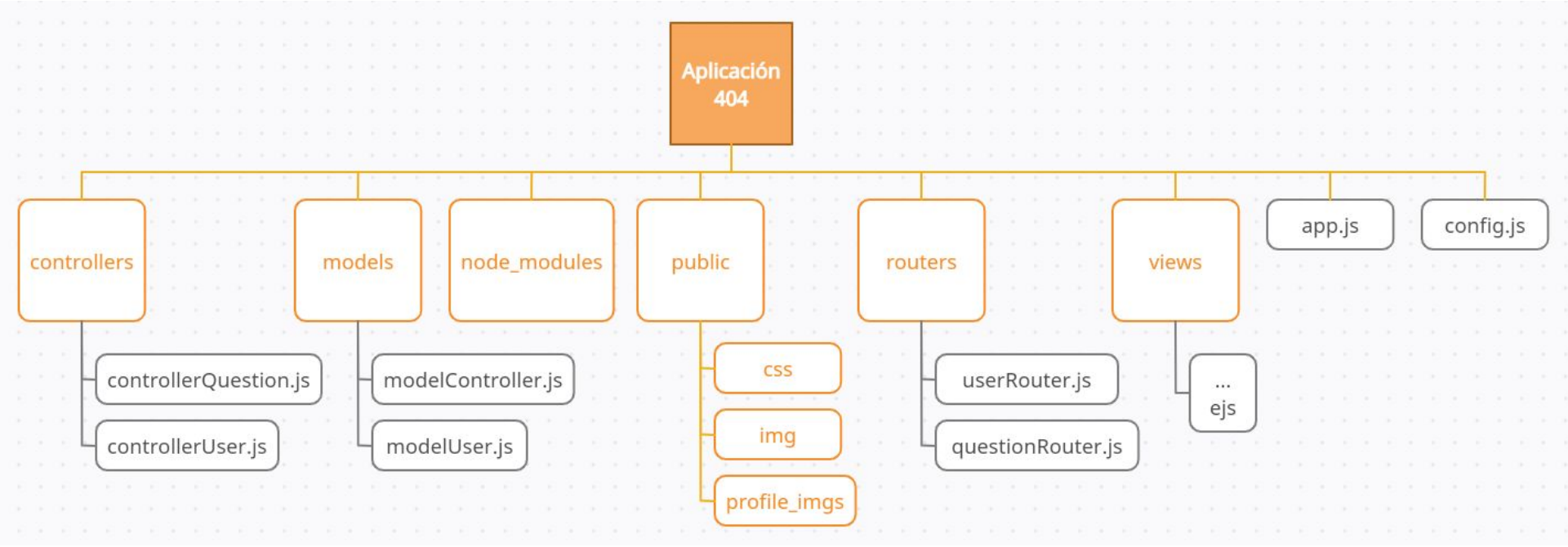
Modelo entidad-relación



Modelo entidad-relación



Estructura de la aplicación



Rutas - userRouter.js - “/”

- **/**
  - **get: redirecciona a /login**
- **/login**
  - **get: muestra la vista de inicio de sesión**
  - **post: comprueba el formulario de login y redirige a /index si es correcto**
- **/registro**
  - **get: muestra la vista de registro**
  - **post: comprueba el formulario de registro y redirige a /login si es correcto**
- **/logout**
  - **get: cierra sesión en la aplicación, elimina los datos de la sesión y redirige a /login**
- **/index**
  - **get: muestra la vista principal de la aplicación**
- **/imagenUsuario**
  - **get: devuelve la imagen, ya existente en el sistema, del usuario logueado**
- **/imagenUsuario/:correo**
  - **get: devuelve la imagen, ya existente en el sistema, de un usuario cualquiera**
- **/perfil**
  - **get: muestra la vista del perfil del usuario logueado**
- **/perfil/:correo**
  - **get: muestra la vista del perfil de cualquier usuario en el sistema**
- **/usuarios**
  - **get: muestra la vista de todos los usuarios registrados**
- **/usuarios/filtrar**
  - **post: filtra por usuarios desde el buscador de la vista de usuarios y renderiza la vista de usuarios registrados con los datos de los usuarios filtrados**

Rutas - questionRouter.js - “/preguntas”

- **/**
  - **get: muestra la vista con todas las preguntas**
- **/formular**
  - **get: muestra la vista para formular una nueva pregunta y redirige a /preguntas**
- **/nuevaPregunta**
  - **post: comprueba el formulario de realizar una nueva pregunta y redirige a /preguntas**
- **/sin_responder**
  - **get: muestra la vista con todas las preguntas que están sin responder**
- **/:id_pregunta**
  - **get: muestra la vista de la pregunta con sus respuestas, así como el formulario para añadir una nueva respuesta**
- **/subirRespuesta/:id_pregunta**
  - **post: sube una nueva respuesta asociada a la pregunta pasada como parámetro y redirige a /preguntas**
- **/preguntasFiltradas**
  - **post: filtra las preguntas por texto desde el buscador de la cabecera y renderiza la vista de todas las preguntas**
- **/filtrar_etiquetas/:nombre_tema**
  - **get: muestra las preguntas que tienen como etiqueta la pasada como parámetro**
- **/votar_positivo/:tipo/:id**
  - **post: aumenta en 1 el contador de votos_positivos de la base de datos. Con :tipo vemos si el voto se ha hecho a una respuesta o a una pregunta y con :id su identificador único de la base de datos.**
- **/votar_negativo/:tipo/:id**
  - **post: aumenta en 1 el contador de votos_negativos de la base de datos. Con :tipo vemos si el voto se ha hecho a una respuesta o a una pregunta y con :id su identificador único de la base de datos.**

Restringir rutas a usuarios no logueados

Tenemos un middleware, llamado `middlewareCurrentUser` que comprueba si la variables `request.session.currentUser y request.session.currentName` no están vacías (ambas se inicializan a la hora de hacer login correctamente), y en caso de no ser así, redirige a `/login`

```
const middlewareCurrentUser = function(request, response, next){
    if(request.session.currentUser && request.session.currentName){
        response.locals.userEmail = request.session.currentUser;
        response.locals.userName = request.session.currentName;
        next();
    }
    else{
        response.redirect("/login");
    }
}
```

Gestión de errores 404 y 500

El tratamiento de estos errores se han realizado a través de dos middlewares:

El propio del error 404, se ejecuta cuando el usuario introduce una url que no está entre las existentes, renderiza la vista error404, que muestra un aviso.

Mientras que los casos donde se produzca un error de tipo 500, renderiza la vista error500 y muestra un aviso y el error que ha ocurrido junto a su pila de ejecución.

```
function middlewareError500(error, request, response, next) {
    response.status(500);
    response.render("error500", {mensaje: error.message, pila:
error.stack});
}

function middlewareError404(request, response){
    response.status(404);
    response.render("error404", { url: request.url });
}
```

Paquetes utilizados

- body_parser: módulo que permite recoger los datos en los inputs en las vistas .ejs
- ejs: módulo que permite usar plantillas en archivos .html
- express: framework de JS que permite la creación de la aplicación web y que se ejecuta en el lado del servidor
- express-mysql-session: módulo que permite almacenar los datos de la sesión del usuario logueado en la base de datos
- express-session: módulo que permite almacenar los datos del usuario logueado en esa sesión, como su correo
- morgan: módulo que muestra por terminar información relevante acerca de las peticiones que se realizan
- mysql: módulo que permite establecer la conexión con nuestra base de datos
- path: módulo que nos ofrece herramientas para el uso de rutas de ficheros en nuestra aplicación