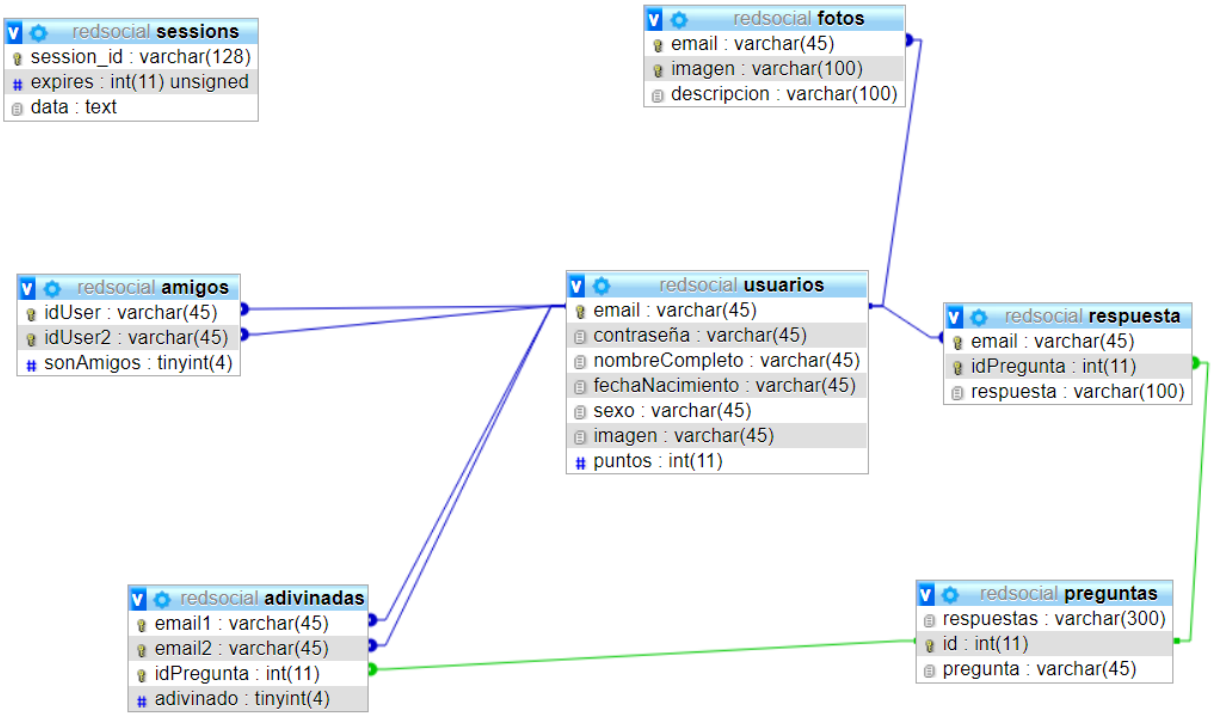
Práctica 1: Red social
Facebluff

Grupo 16

Álvaro Noriega Moreno

José Antonio Garrido Sualdea

Diseño de la base de datos



Implementación de la sesión para un usuario logueado.

Hemos utilizado peticiones GET, POST y rutas paramétricas.

Estructura de la aplicación

Hemos estructurado la aplicación en diferentes modulos:

- Dao: se compone de DaoUsuarios y DaoPreguntas. DaoUsuarios realiza las consultas de la base de datos y las diferentes operaciones de la capa de integración que corresponden a los usuarios. DaoPreguntas realiza las consultas de la base de datos y las diferentes operaciones de la capa de integración que corresponden a las preguntas.

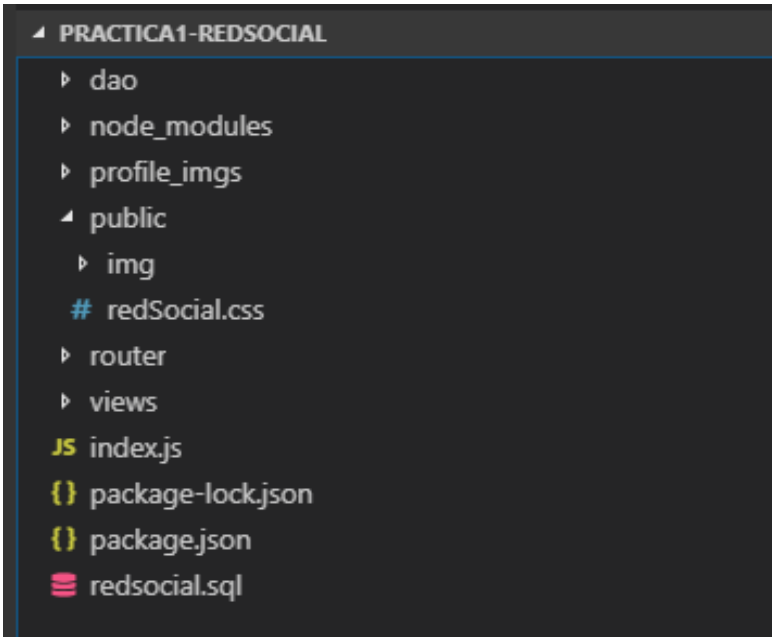
- Router: hemos utilizado dos routers para agrupar los diferentes manejadores de ruta, routerPreguntas para los manejadores correspondientes a las preguntas y routerUsuarios para los manejadores correspondientes a los usuarios.

- Views: donde ubicamos las plantillas (.ejs) de las vistas.

- Profile_imgs: contiene las imágenes de usuario y las imágenes que se pueden subir a través de la aplicación.

- Public: contiene los diferentes hojas de estilo (.css) utilizados por las vistas y un directorio “img” donde se encuentra la imagen que mostramos cuando un usuario no tiene imagen de perfil.

- Node_modules: donde se descargan y se meten los paquetes necesarios para la implementación y utilización de nuestra aplicación.



Listado de la rutas gestionadas por el servidor.

- Usuarios:
  - “/usuarios/login.html”: ruta inicial para el inicio de sesión.
  - “/usuarios/newUser.html” : ruta para la creación de un nuevo usuario.
  - “/usuarios/form_login.html”: ruta para procesar el formulario de inicio de sesión.
  - “/usuarios/form_newUser.html”: ruta para procesar el formulario de creación de un usuario.
  - “/usuarios/profile.html”: ruta que muestra el perfil de un usuario.
  - “/usuarios/modProfile.html”: ruta que muestra la pantalla de modificar el perfil.
  - “/usuarios/form_modProfile.html”: ruta para procesar el formulario de modificación de un perfil.
  - “/usuarios/friends.html": ruta que muestra la pantalla de amigos del usuario.
  - “/usuarios/amigos/AceptarSolicitud": ruta para procesar la confirmación de una solicitud de amistad.
  - “/usuarios/amigos/RechazarSolicitud": ruta para procesar el rechazo de una solicitud de amistad.
  - “/usuarios/amigos/buscarUsuarios": ruta para mostrar la búsqueda de un usuario.
  - “/usuarios/amigos/enviarSolicitud": ruta para enviar la solicitud a un usuario buscado.
  - “/usuarios/amigos/perfilAmigos”: ruta que muestra el perfil de un amigo.
  - “/usuarios/form_subirFoto”: ruta que procesa el formulario para la subida de una foto al perfil.
  - “usuarios/imagenUsuario”: ruta que procesa la carga de la imagen de perfil de un usuario.
  - “usuarios/logOut”: ruta para desconectar al usuario.

- Preguntas:
  - “preguntas/tablero.html": ruta que carga la pantalla de preguntas con preguntas aleatorias.
  - “/preguntas/crearPregunta.html”: ruta que carga la pantalla de creación de una pregunta.
  - “/preguntas/adivinarPregunta": ruta que carga la pantalla para responder la pregunta de un amigo.
  - “/preguntas/insertarPregunta”: ruta que procesa el formulario de la creación de una pregunta.
  - “/preguntas/comprobarAdivinacion": ruta que procesa y comprueba la respuesta a una pregunta de un amigo.

- “/preguntas/seleccionarPregunta”: ruta que carga la pregunta seleccionada y la lista de amigos que han respondido o no a la pregunta.
- “/preguntas/insertarRespuesta”: ruta que procesa la respuesta seleccionada entre las previamente dadas.
- “/preguntas/insertarRespuesta”: ruta que procesa una nueva respuesta.

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

Para la restricción de acceso de los usuarios no logueados hemos utilizado un middleware verifyUser en cada router, el cual se encarga de redirigir a la página de login si el usuario no ha iniciado sesión.

```
function verifyUser(request,response,next){
    if(request.session.currentUser===undefined){
        response.redirect('/usuarios/login.html');
    }else{
        next();
    }
}
```

Gestión de los errores 404 y 500

Hemos utilizado dos manejadores de errores en el archivo index.js de nuestra aplicación que se encargan de gestionar los errores producidos, uno para el error 404 y otro para el error 500.

El manejador del error 404 carga la vista error404.js donde se muestra la dirección a la que no ha podido acceder.

El manejador del error 500 carga la vista error500.js donde el error ocurrido en la base de datos.

```
app.use(function(error, request, response, next) {
    // Código 500: Internal server error
    response.status(500);
    response.render("error500", {mensaje: error.message, pila: error.stack });
});
app.use(function(req,res,next){
    res.status(404);
    res.render("error404",{url:req.url});
});
```