**Información General**

**Proyecto:** Portal Cósmico.  
**Fecha:** Mayo 2025  
**Arquitectura:** REST API + Base de datos relacional  
**Tecnologías:** React, Vite, Node.js, Express, MySQL, Socket.io

**Objetivos del Sistema**

Red social que permite a los usuarios:

* Crear perfiles personales con información y fotos
* Publicar contenido (texto, imágenes, videos)
* Interactuar con publicaciones (likes, comentarios)
* Conectar con otros usuarios (seguidores)
* Mensajería privada en tiempo real
* Feed personalizado de publicaciones
* Sistema de notificaciones

**Diseño de Base de Datos**

**Diagrama Entidad-Relación**

Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.

**Definición de Tablas**

-- Estructura de tabla para la tabla `**categoria**`

--

CREATE TABLE `categoria` (

`id` int(11) NOT NULL,

`nombre` varchar(191) NOT NULL,

`descripcion` varchar(191) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**comentario**`

--

CREATE TABLE `comentario` (

`id` int(11) NOT NULL,

`content` varchar(191) NOT NULL,

`createdAt` datetime(3) NOT NULL DEFAULT current\_timestamp(3),

`postId` int(11) NOT NULL,

`userId` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**message**`

--

CREATE TABLE `message` (

`id` int(11) NOT NULL,

`content` varchar(191) NOT NULL,

`isRead` tinyint(1) NOT NULL DEFAULT 0,

`createdAt` datetime(3) NOT NULL DEFAULT current\_timestamp(3),

`senderId` int(11) NOT NULL,

`receiverId` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**notification**`

--

CREATE TABLE `notification` (

`id` int(11) NOT NULL,

`type` varchar(191) NOT NULL,

`isRead` tinyint(1) NOT NULL DEFAULT 0,

`createdAt` datetime(3) NOT NULL DEFAULT current\_timestamp(3),

`userId` int(11) NOT NULL,

`fromUserId` int(11) DEFAULT NULL,

`postId` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**post**`

--

CREATE TABLE `post` (

`id` int(11) NOT NULL,

`description` varchar(191) NOT NULL,

`content` longblob DEFAULT NULL,

`createdAt` datetime(3) NOT NULL DEFAULT current\_timestamp(3),

`userId` int(11) NOT NULL,

`categoryId` int(11) NOT NULL,

`contentType` varchar(191) DEFAULT NULL,

`emojiData` varchar(191) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**postlike**`

--

CREATE TABLE `postlike` (

`id` int(11) NOT NULL,

`userId` int(11) NOT NULL,

`postId` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**userfollower**`

--

CREATE TABLE `userfollower` (

`id` int(11) NOT NULL,

`followerId` int(11) NOT NULL,

`followedId` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- Estructura de tabla para la tabla `**usuario**`

--

CREATE TABLE `usuario` (

`id` int(11) NOT NULL,

`username` varchar(191) NOT NULL,

`email` varchar(191) NOT NULL,

`password` varchar(191) NOT NULL,

`profilePic` longblob DEFAULT NULL,

`bio` varchar(191) DEFAULT '',

`createdAt` datetime(3) NOT NULL DEFAULT current\_timestamp(3)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

**Rutas**

**/ ENDPOINTS DE USUARIO**

**// Registro de usuario**

**app.post('/register', upload.single('profilePic'), register);**

**// Inicio de sesión**

**app.post('/login', login);**

**// Obtener perfil de usuario**

**app.get('/profile', auth, getUserProfile);**

**// Obtener perfil de otro usuario**

**app.get('/profile/:userId', auth, getOtherUserProfile);**

**// Actualizar perfil de usuario**

**app.put('/updateprofile', auth, upload.single('profilePic'), updateUserProfile);**

**// Buscar usuarios**

**app.get('/search', searchUsers);**

**// Obtener todos los usuarios excepto el actual**

**app.get('/getAllUsersExceptCurrent', auth, getAllUsersExceptCurrent);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE CATEGORIAS**

**// Crear categoría**

**app.post('/createCategory', createCategory);**

**// Obtener una categoría por ID**

**app.get('/getCategory/:categoryId', getCategoryById);**

**// Obtener todas las categorías**

**app.get('/getAllCategories', getAllCategories);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE PUBLICACIONES**

**// Crear publicación**

**app.post('/createPost', auth, upload.single('file'), createPost);**

**// Obtener publicaciones recientes**

**app.get('/getRecentPosts', auth, getRecentPosts);**

**// Obtener publicaciones de los usuarios que sigo**

**app.get('/getFeedPosts', auth, getFeedPosts);**

**// Obtener publicaciones por categoría**

**app.get('/getPostsByCategory/:categoryId', auth, getPostsByCategory);**

**// Buscar publicaciones**

**app.get('/searchPosts', auth, searchPosts);**

**// Obtener publicaciones de un usuario**

**app.get('/getUserPosts/:userId', auth, getUserPosts);**

**// Obtener una publicación específica**

**app.get('/getPostById/:postId', auth, getPostById);**

**// Actualizar publicación**

**app.put('/updatePost/:postId', auth, upload.single('file'), updatePost);**

**// Eliminar publicación**

**app.delete('/deletePost/:postId', auth, deletePost);**

**// Obtener publicaciones de no seguidos**

**app.get('/getExplorePosts', auth, getExplorePosts);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE NOTIFICACIONES**

**// Obtener todas las notificaciones del usuario actual**

**app.get('/getUserNoti', auth, getUserNotifications);**

**// Obtener el número de notificaciones no leídas**

**app.get('/getUnreadNotiCount', auth, getUnreadNotificationsCount);**

**// Marcar una notificación como leída**

**app.put('/markNotiAsRead/:notificationId', auth, markNotificationAsRead);**

**// Marcar todas las notificaciones como leídas**

**app.put('/markAllNotiAsRead', auth, markAllNotificationsAsRead);**

**// Eliminar una notificación**

**app.delete('/deleteNoti/:notificationId', auth, deleteNotification);**

**// Eliminar todas las notificaciones**

**app.delete('/deleteAllNoti', auth, deleteAllNotifications);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE SEGUIDORES**

**// Seguir a un usuario**

**app.post('/followUser/:userId', auth, followUser);**

**// Dejar de seguir a un usuario**

**app.delete('/unfollowUser/:userId', auth, unfollowUser);**

**// Obtener seguidores de un usuario**

**app.get('/getUserFollowers/:userId', auth, getUserFollowers);**

**// Obtener usuarios seguidos por un usuario**

**app.get('/getUserFollowing/:userId', auth, getUserFollowing);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE LIKES**

**// Dar like a una publicación**

**app.post('/likePost/:postId', auth, likePost);**

**// Quitar like de una publicación**

**app.delete('/unlikePost/:postId', auth, unlikePost);**

**// Obtener usuarios que dieron like a una publicación**

**app.get('/getPostLikes/:postId', getPostLikes);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE COMENTARIOS**

**// Crear comentario**

**app.post('/createComment/:postId', auth, createComment);**

**// Actualizar comentario (nuevo endpoint)**

**app.put('/updateComment/:commentId', auth, updateComment);**

**// Obtener comentarios de una publicación**

**app.get('/getPostComments/:postId', getPostComments);**

**// Eliminar comentario**

**app.delete('/deleteComment/:commentId', auth, deleteComment);**

**//-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**// ENDPOINTS DE MENSAJES**

**// Enviar un mensaje a otro usuario**

**app.post('/sendMessage', auth, sendMessage);**

**// Obtener conversaciones del usuario**

**app.get('/getUserConversations', auth, getUserConversations);**

**// Obtener mensajes entre dos usuarios**

**app.get('/getConversationMessages/:userId', auth, getConversationMessages);**

**// Obtener el conteo de mensajes no leídos**

**app.get('/getUnreadMessageCount', auth, getUnreadMessageCount);**

**// Marcar todos los mensajes de una conversación como leídos**

**app.put('/markConversationAsRead/:userId', auth, markConversationAsRead);**

**// Eliminar un mensaje (solo el remitente puede eliminar)**

**app.delete('/deleteMessage/:messageId', auth, deleteMessage);**