

**App Marvel con Firebase: Búsqueda, Autenticación y Gestión de Favoritos Integrados en una SPA con Backbone y Marionette**

Eliana Milagros Lucas Gómez

**RESUMEN.**

Este trabajo presenta una aplicación web de página única (SPA) que permite a los usuarios buscar personajes, cómics y series del universo Marvel utilizando la API pública de Marvel, así como registrarse, iniciar sesión y guardar sus favoritos en Firebase Firestore. La aplicación está desarrollada utilizando tecnologías modernas del ecosistema JavaScript: Backbone.js y Marionette.js para la estructura del frontend, Firebase para la autenticación y base de datos, y Tailwind CSS para el diseño visual. Se aborda el diseño modular, la integración de servicios externos y la experiencia de usuario centrada en la simplicidad y funcionalidad.

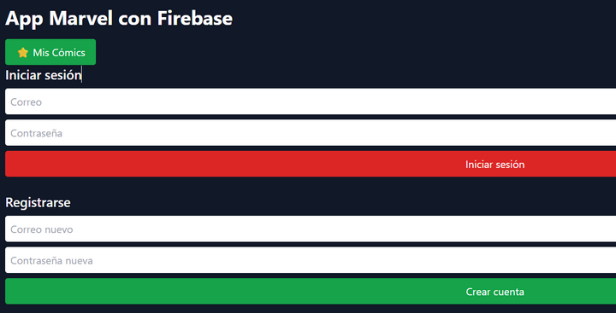


Ilustración 1. Api Marvel

**PALABRAS CLAVES:**

Backbone, firebase , cryptoJS, Marvel API.

**ABSTRACT**

This work features a single-page web application (SPA) that allows users to search for characters, comics, and series from the Marvel universe using Marvel's public API, as well as register, log in, and save their favorites to Firebase Firestore. The application is developed using modern technologies from the JavaScript ecosystem: Backbone.js and Marionette.js for the frontend structure, Firebase for authentication and database, and

Tailwind CSS for visual design. Modular design, integration of external services and user experience focused on simplicity and functionality are addressed.

**KEY WORDS**

Backbone, firebase , cryptoJS, Marvel API.

1. **INTRODUCCIÓN.**

El creciente interés en las aplicaciones web dinámicas ha impulsado la adopción de frameworks JavaScript como Backbone.js y Marionette.js, que permiten estructurar interfaces complejas de forma escalable. Este proyecto integra estas tecnologías con la API de Marvel para consumir contenido oficial y con Firebase para gestión de usuarios y persistencia en la nube. El objetivo es desarrollar una SPA educativa y entretenida que permita explorar el universo Marvel con opciones personalizadas para cada usuario.

1. **TECNOLOGÍAS UTILIZADAS**

* **Backbone.js & Marionette.js**: Para el control de vistas, modelos y colecciones.
* **Firebase Authentication**: Manejo de registros e inicio de sesión de usuarios.
* **Firebase Firestore:** Base de datos en tiempo real para almacenar los favoritos.
* **Marvel API**: Fuente oficial para la obtención de cómics, personajes y series.
* **Tailwind CSS:** Estilizado rápido, responsivo y moderno.
* **CryptoJS:** Para generar el hash de autenticación requerido por Marvel.

1. **ESTRUCTURA DEL PROYECTO**

El proyecto se divide en los siguientes módulos principales:

* **index.html:** Define la interfaz principal, formularios de login y registro, selector de búsqueda y secciones de resultados y favoritos.
* **firebase-config.js:** Archivo de configuración e inicialización de Firebase (modular ESModules).
* **app.js**: Punto de entrada principal que inicia la vista global Marionette.
* **Vistas Marionette:** Incluyen vistas para login, resultados de búsqueda y gestión de favoritos.
* **Controladores**: Gestionan la lógica de autenticación, consultas a la API de Marvel y operaciones en Firestore.

**4. FUNCIONALIDADES**

**4.1 Registro e Inicio de Sesión**  
Mediante Firebase Authentication, los usuarios pueden crear cuentas o iniciar sesión. Se valida el correo y contraseña desde el frontend. El sistema reacciona automáticamente al estado de sesión gracias a onAuthStateChanged.



Ilustración 2. Interfaz de Login

**4.2 Búsqueda en Marvel API**  
El usuario puede seleccionar si desea buscar personajes, cómics o series. Al realizar una búsqueda, se genera el hash de seguridad requerido por la API de Marvel con CryptoJS y se construye la URL correspondiente. Los resultados se muestran dinámicamente con botones de acción.

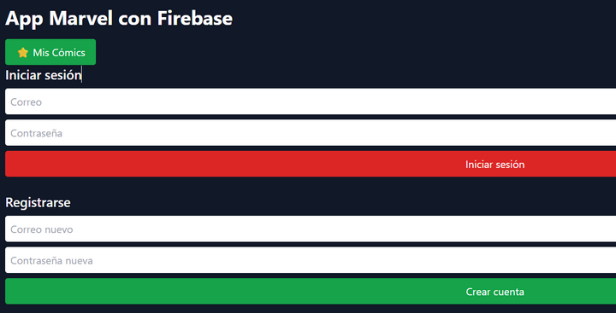


Ilustración 3. Búsqueda según lo que se desee

**4.3 Gestión de Favoritos**  
Cada usuario puede agregar un elemento a sus favoritos. Estos se almacenan bajo una subcolección del usuario autenticado en Firestore. Los favoritos pueden visualizarse y eliminarse, con sincronización en tiempo real.



Ilustración 4. Apartado de favoritos

**5. Diseño e Interfaz de Usuario** La aplicación utiliza Tailwind CSS para lograr un diseño responsivo, moderno y accesible. Se aplican colores oscuros con contrastes para simular el estilo Marvel, e íconos visuales () para mejorar la experiencia del usuario.

**6. Resultados** La aplicación fue probada con múltiples usuarios y combinaciones de búsquedas, cumpliendo con las siguientes metas:

* Registro y autenticación funcional
* Acceso controlado al contenido (solo usuarios autenticados)
* Búsqueda efectiva en la API de Marvel
* Persistencia de favoritos personalizada por usuario
* Interfaz clara y funcional

**7. Conclusiones** Este proyecto demuestra la capacidad de integrar servicios modernos como Firebase y APIs externas dentro de un entorno estructurado con Backbone.js y Marionette. A pesar de que estas tecnologías no son las más nuevas, su correcta aplicación sigue permitiendo desarrollos modulares y mantenibles. La utilización de Firebase permitió simplificar enormemente la gestión de backend.

**8. Trabajo Futuro** Para continuar mejorando la aplicación se proponen:

* Implementar una vista de detalles para personajes, cómics y series.
* Mejorar la seguridad ocultando claves privadas mediante funciones en servidor (Cloud Functions).
* Añadir filtros por fechas, autores y más metadatos.
* Soporte offline y uso de Firebase Storage para subir imágenes personalizadas.

**9. ANEXOS**

**1. Código main.js**

Este código permite iniciar sesión en una app web usando Firebase Authentication. Al enviar el formulario, toma el correo y la contraseña ingresados, intenta autenticarlos con Firebase, y si son correctos, redirige al usuario a la página principal; si hay un error, muestra un mensaje indicando que las credenciales son incorrectas.

|  |
| --- |
| import { auth } from './firebase-config.js';  import { signInWithEmailAndPassword } from 'firebase/auth';  document.addEventListener("DOMContentLoaded", () => {    const loginForm = document.getElementById("login-form");    const errorMessage = document.getElementById("error-message");    loginForm.addEventListener("submit", (e) => {      e.preventDefault();      const email = document.getElementById("email").value;      const password = document.getElementById("password").value;      signInWithEmailAndPassword(auth, email, password)        .then((userCredential) => {          // Login exitoso          const user = userCredential.user;          console.log("Usuario logueado:", user.email);          window.location.href = "index.html"; // Redirige a la página principal de la app        })        .catch((error) => {          console.error("Error de autenticación:", error.message);          errorMessage.textContent = "Correo o contraseña incorrectos.";        });    });  }); |

**2. Código app.js**

Inicializa la aplicación web creando una vista principal llamada VistaGlobal usando Backbone y Marionette, y luego la renderiza e inserta en el elemento <body> del documento para que se muestre la interfaz de usuario.

|  |
| --- |
| // app.js: Inicializa la app y la vista global  $(function () {    // Creamos la vista global    const vistaGlobal = new Marvel.Views.VistaGlobal();    // Insertamos la vista global en el body    vistaGlobal.render().$el.appendTo('body');  }); |

**3.Código collections comic.js**

Este código define una colección de cómics en una aplicación Backbone. Utiliza la API de Marvel para buscar cómics cuyo título comience con un texto dado.

|  |
| --- |
| Marvel.Collections = Marvel.Collections || {};  Marvel.Collections.Comics = Backbone.Collection.extend({    model: Marvel.Models.Comic,    buscar: function(titulo) {      const ts = Date.now().toString();      const publicKey = '1ef11f3874e9325cdb1600201a7ecdc6';      const privateKey = 'f019ac886777717927fcd82231b3b948ab6af9a0';      // Usa CryptoJS para generar el hash: md5(ts + privateKey + publicKey)      const hash = CryptoJS.MD5(ts + privateKey + publicKey).toString();      this.url = `https://gateway.marvel.com/v1/public/comics?titleStartsWith=${encodeURIComponent(titulo)}&ts=${ts}&apikey=${publicKey}&hash=${hash}`;      this.fetch({        reset: true,        success: function() { console.log("¡Búsqueda completa!"); },        error: function() { console.error("Error en la búsqueda."); }      });    },    parse: function(response) {      return response.data.results.map(comic => ({        id: comic.id,        title: comic.title,        description: comic.description || "Sin descripción",        thumbnail: comic.thumbnail.path + '/standard\_medium.' + comic.thumbnail.extension,        thumbnailGrande: comic.thumbnail.path + '/portrait\_uncanny.' + comic.thumbnail.extension      }));    }  }); |

**4.Código model Comics.js**

|  |
| --- |
| Marvel.Models = Marvel.Models || {};  Marvel.Models.Comic = Backbone.Model.extend({}); |

**5.Código views**

* **Código VistabuscarComics.js**

|  |
| --- |
| Marvel.Views.VistaBuscarComics = Marionette.View.extend({    template: '#VistaBuscarComicsTmpl',    events: {      'click #botonBuscar': 'buscar'    },    initialize: function() {      this.collection = new Marvel.Collections.Comics();      this.listenTo(this.collection, 'sync', this.busquedaCompletada.bind(this));    },    buscar: function(e) {      e.preventDefault();      const texto = this.$('#textoBuscar').val().trim();      if (texto) this.collection.buscar(texto);    },    busquedaCompletada: function() {      this.triggerMethod('completed:search', this.collection);    }  }); |

* **Código VistaComic.js**

|  |
| --- |
| Marvel.Views = Marvel.Views || {};  Marvel.Views.VistaComic = Marionette.View.extend({    tagName: 'div',    className: 'item-comic',    template: '#VistaComicTmpl',    events: {      'click .ver-detalles': 'verDetalles'    },    verDetalles: function(e) {      e.preventDefault();      this.triggerMethod('show:details', this.model);    }  }); |

* **Código VistaDEtallesComic.js**

|  |
| --- |
| // VistaDetallesComic.js  import Marionette from 'backbone.marionette';  import { auth, db } from './firebase-config.js';  import { collection, addDoc } from "firebase/firestore";  export default Marionette.View.extend({    template: \_.template(`      <h2><%- title %></h2>      <img src="<%- thumbnail %>" />      <button id="btn-favorito">Agregar a Favoritos</button>      <div id="mensaje"></div>    `),    events: {      'click #btn-favorito': 'agregarAFavoritos'    },    agregarAFavoritos() {      const user = auth.currentUser;      if (!user) {        this.$('#mensaje').text('Debes iniciar sesión para agregar favoritos.');        return;      }      const comicData = {        userId: user.uid,        comicId: this.model.get('id'),        title: this.model.get('title'),        thumbnail: this.model.get('thumbnail')      };      addDoc(collection(db, "favoritos"), comicData)        .then(() => {          this.$('#mensaje').text('¡Agregado a favoritos!');        })        .catch(err => {          this.$('#mensaje').text(`Error: ${err.message}`);        });    }  }); |

* **Código VistaFavoritos.js**

|  |
| --- |
| // VistaMisComics.js  import Marionette from 'backbone.marionette';  import { auth, db } from './firebase-config.js';  import { collection, query, where, getDocs } from "firebase/firestore";  export default Marionette.View.extend({    template: \_.template(`      <h2>Mis Cómics Favoritos</h2>      <ul id="lista-favoritos"></ul>      <div id="mensaje"></div>    `),    onRender() {      this.cargarFavoritos();    },    async cargarFavoritos() {      const user = auth.currentUser;      if (!user) {        this.$('#mensaje').text('Debes iniciar sesión para ver tus favoritos.');        return;      }      const favoritosRef = collection(db, "favoritos");      const q = query(favoritosRef, where("userId", "==", user.uid));      try {        const querySnapshot = await getDocs(q);        const lista = this.$('#lista-favoritos');        lista.empty();        querySnapshot.forEach(doc => {          const data = doc.data();          lista.append(`<li>${data.title} <img src="${data.thumbnail}" width="50"></li>`);        });        if (querySnapshot.empty) {          this.$('#mensaje').text('No tienes favoritos aún.');        } else {          this.$('#mensaje').empty();        }      } catch (error) {        this.$('#mensaje').text(`Error: ${error.message}`);      }    }  }); |

* **Código VistaGlobal.js**

|  |
| --- |
| Marvel.Views.VistaGlobal = Marionette.View.extend({    el: 'body',    regions: {      cabecera: '#cabecera',      formBusqueda: '#formBusqueda',      listado: '#listado'    },    childEvents: {      'completed:search': function(child, col) {        this.vistaLista = new Marvel.Views.VistaComics({ collection: col });        this.showChildView('listado', this.vistaLista);      },      'show:details': function(child, model) {        const detalles = new Marvel.Views.VistaDetallesComic({ model: model });        this.getRegion('listado').show(detalles, { preventDestroy: true });      },      'hide:details': function() {        this.showChildView('listado', this.vistaLista);      }    }  }); |

* **Código VistaLogin.js**

|  |
| --- |
| import Marionette from 'backbone.marionette';  import { auth } from './firebase-config.js';  import { signInWithEmailAndPassword, signOut } from "firebase/auth";  export default Marionette.View.extend({    template: \_.template(`      <form id="form-login">        <input type="email" id="email" placeholder="Email" required />        <input type="password" id="password" placeholder="Password" required />        <button type="submit">Login</button>        <button type="button" id="logout" style="display:none;">Logout</button>        <div id="mensaje"></div>      </form>    `),    events: {      'submit #form-login': 'onLogin',      'click #logout': 'onLogout'    },    onLogin(e) {      e.preventDefault();      const email = this.$('#email').val();      const password = this.$('#password').val();      signInWithEmailAndPassword(auth, email, password)        .then(userCredential => {          this.$('#mensaje').text(`Bienvenido ${userCredential.user.email}`);          this.$('#logout').show();        })        .catch(err => {          this.$('#mensaje').text(`Error: ${err.message}`);        });    },    onLogout() {      signOut(auth).then(() => {        this.$('#mensaje').text('Sesión cerrada');        this.$('#logout').hide();      });    }  }); |

**\***

* **Código VistaMisComics.js**

|  |
| --- |
| import Marionette from 'backbone.marionette';  import { auth, db } from './firebase-config.js';  import { collection, addDoc } from "firebase/firestore";  export default Marionette.View.extend({    template: \_.template(`      <h2><%- title %></h2>      <img src="<%- thumbnail %>" />      <button id="btn-favorito">Agregar a Favoritos</button>      <div id="mensaje"></div>    `),    events: {      'click #btn-favorito': 'agregarAFavoritos'    },    agregarAFavoritos() {      const user = auth.currentUser;      if (!user) {        this.$('#mensaje').text('Debes iniciar sesión para agregar favoritos.');        return;      }      const comicData = {        userId: user.uid,        comicId: this.model.get('id'),        title: this.model.get('title'),        thumbnail: this.model.get('thumbnail')      };      addDoc(collection(db, "favoritos"), comicData)        .then(() => {          this.$('#mensaje').text('¡Agregado a favoritos!');        })        .catch(err => {          this.$('#mensaje').text(`Error: ${err.message}`);        });    }  }); |

**10. REFERENCIAS**

1. Marvel Developer Portal – <https://developer.marvel.com>
2. Firebase Documentation – <https://firebase.google.com/docs>
3. Backbone.js Documentation – <https://backbonejs.org/>
4. Marionette.js Docs – <https://marionettejs.com>