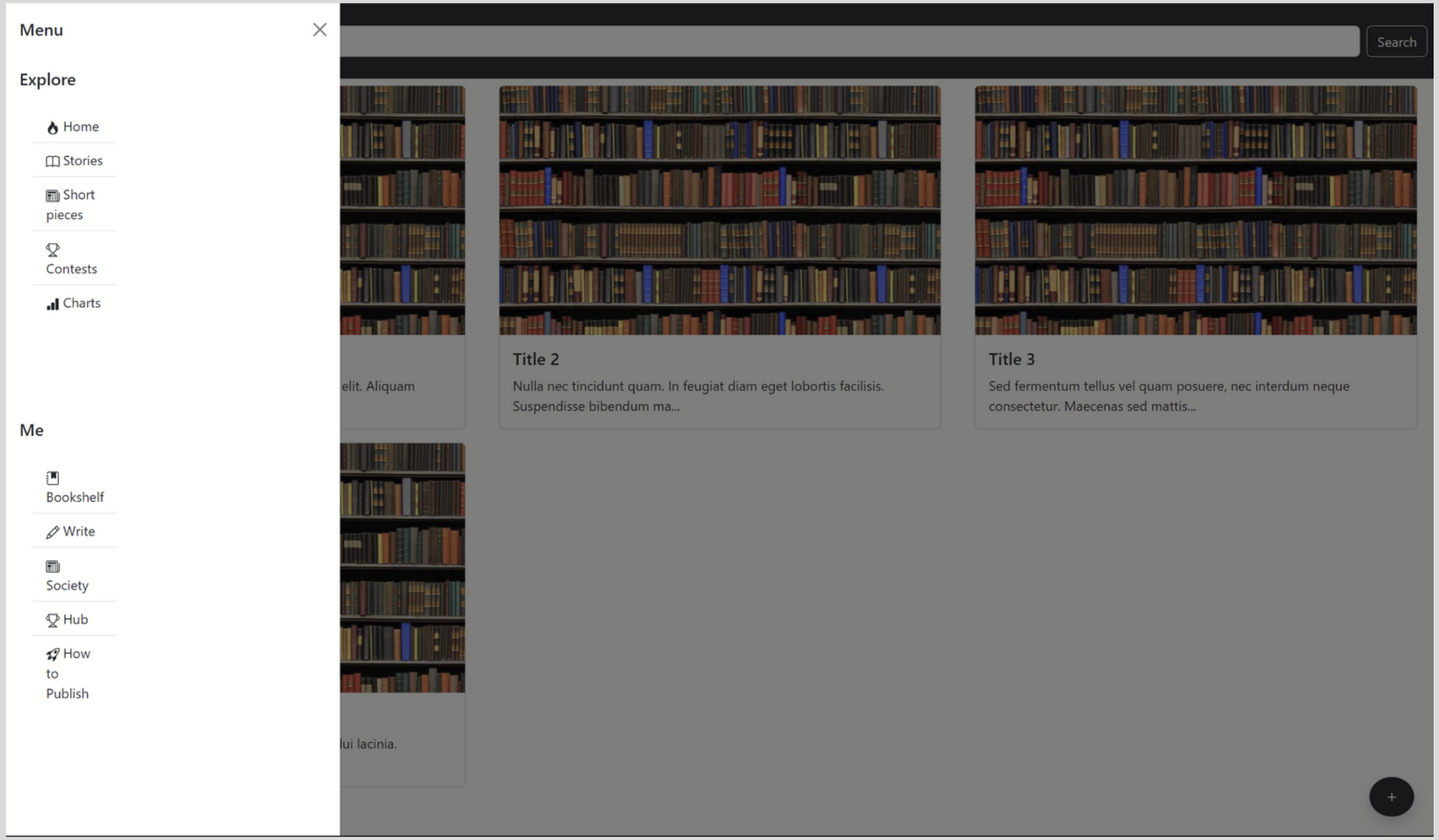
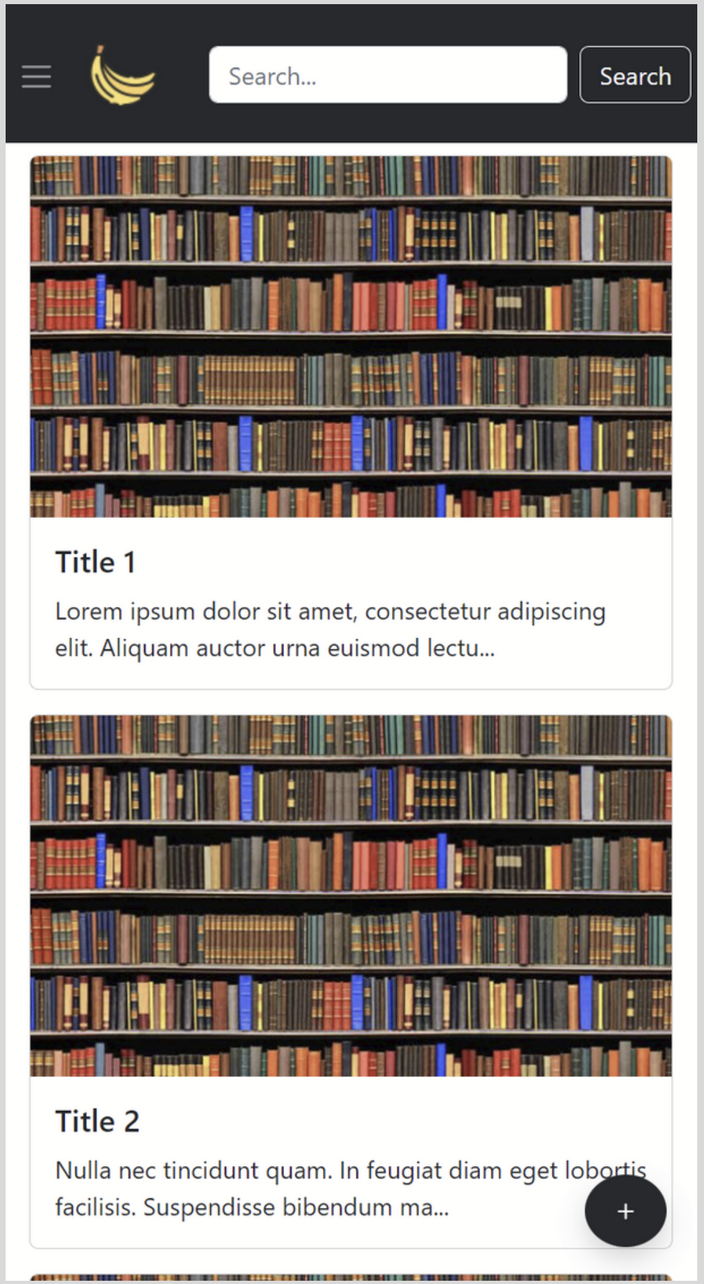
Equipo 09

Diseño app


Menu
Explore
Home
Stories
Short pieces
Contests
Charts
Me
Bookshelf
Write
Society
Hub
How to Publish
Search
Title 2
Nulla nec tincidunt quam. In feugiat diam eget lobortis facilisis. Suspendisse bibendum ma...
Title 3
Sed fermentum tellus vel quam posuere, nec interdum neque consectetur. Maecenas sed mattis...


Search...
Search
Title 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam auctor urna euismod lectu...
Title 2
Nulla nec tincidunt quam. In feugiat diam eget lobortis facilisis. Suspendisse bibendum ma...

Manifest

Se setean los íconos y se elige el display "standalone".

Problema inicial con los tamaños de los íconos.

```
{
    "name": "Banana App",
    "short_name": "Banana",
    "icons": [···
    ],
    "gcm_sender_id": "388679300373",

    "lang": "en-US",
    "scope": "/",
    "start_url": "/index.html",
    "display": "standalone",
    "background_color": "#fff",
    "theme_color": "#fff",
    "description": "Fiction works publications",
    "orientation": "portrait-primary",
    "dir": "ltr"
}
```

Service Worker

Basado en guía de proyecto hello pwa (github).

Se almacenan en caché los archivos especificados y se define install y fetch.

```
1   // Name of the Cache.
2   const CACHE = 'cacheV1';
3
4   // Select files for caching.
5 > let urlsToCache = [ ···
16  ];
17
18  // Cache all the selected items once application is installed.
19  self.addEventListener('install', (event) => {
20    event.waitUntil(
21      caches.open(CACHE).then((cache) => {
22        console.log('Caching started.');
23        return cache.addAll(urlsToCache);
24      })
25    );
26  });
```

Service Worker

```
self.addEventListener("fetch", (event) => {
    if (event.request.method === "GET") {
        event.respondWith(
            fetch(event.request)
            .then((fetchResponse) => {
                // Clone the response as it can be consumed only once.
                const clonedResponse = fetchResponse.clone();

                // Open the cache and add the cloned response to it.
                caches.open(CACHE).then((cache) => {
                    cache.put(event.request, clonedResponse);
                });

                return fetchResponse; // Return the fetched response.
            })
            .catch(() => {
                return caches.match(event.request); // Return the cached response if fetch fails.
            })
        );
    }
});
```

Offline

```
<!-- Banner desconexion -->
<div class="container">
    <div id="offline-banner" class="alert alert-secondary text-center" style="display: none;">
        Ups! Parece que no hay conexión. Algunas funciiones no estarán disponibles.
    </div>
</div>

<script>
    // Check if the app is offline
    function checkOfflineStatus() {
        const offlineBanner = document.getElementById('offline-banner');
        if (!navigator.onLine) {
            offlineBanner.style.display = 'block';
        } else {
            offlineBanner.style.display = 'none';
        }
    }

    // Initial check on page load
    checkOfflineStatus();

    // online-offline
    window.addEventListener('online', checkOfflineStatus);
    window.addEventListener('offline', checkOfflineStatus);
</script>
```

Push notifications

Se utilizó guia de thisdot.co para hacer push notifications con firebase.

Uso de postman para pruebas.

Uso de axios para enviar las notificaciones desde la app.

```
// Initialize Firebase
const firebaseConfig = {
    apiKey: 'AIzaSyC0MNLAzVhRZfTYljK58Nr-lJCZl_XRkt8',
    authDomain: 'pwa-grupo09.firebaseapp.com',
    projectId: 'pwa-grupo09',
    storageBucket: 'pwa-grupo09.appspot.com',
    messagingSenderId: '388679300373',
    appId: '1:388679300373:web:192f2ff3766f53da09651c',
    measurementId: 'G-98MN6WTRS6',
};

firebase.initializeApp(firebaseConfig);
//const db = firebase.firestore();
const messaging = firebase.messaging();
```

Push notifications

```
const messaging = firebase.messaging();
messaging.setBackgroundMessageHandler((payload) => {
    const notification = JSON.parse(payload.data.notification);
    const notificationTitle = notification.title;
    const notificationOptions = {
        body: notification.body,
    };
    //Show the notification :)
    return self.registration.showNotification(
        notificationTitle,
        notificationOptions
    );
});
```

Push notifications

```
messaging
    .requestPermission()
    .then(() => {
        return messaging.getToken();
    })
    .then((token) => {
        console.log("token: " + token);
        const url = `http://localhost:3000/tokens/${token}`;
        fetch(url, {
            method: "POST",
        })
    })
    .catch((err) => {
        console.log('No permission to send push', err);
    });
```

Push notifications

```
async function pushNotifications(information) {
    const url = " https://fcm.googleapis.com/fcm/send";

    const headers = {
        "Content-Type": "application/json",
        Authorization: "key=AAAAWn8XdRU:APA91bG4TZQqE2ZYdgBNAoIIqReHTgq0PKvJ7tyHstbZLorVA57WrputDuuH4W6VqrjvKIRH4r7Kj
    };

    tokens = await db_token.getAllTokens();
    tokens.forEach((token) => {
        const data = {
            notification: {
                title: information.title,
                body: information.body,
                click_action: "http://127.0.0.1:8080/index.html",
                icon: "https://icons.veryicon.com/png/o/food--drinks/supermarket-supplies/banana-37.png",
            },
            to: token,
        };

        axios
            .post(url, data, { headers })
            .then((response) => {
                console.log("Response:", response.data);
            })
            .catch((error) => {
                console.error("Error:", error);
            });
    });
}

module.exports = pushNotifications;
```

Server

```
const express = require("express");
const app = express();
const port = 3000;
const cors = require("cors");
const morgan = require("morgan");
const bodyParser = require("body-parser");

const db_post = require("./queriesPost");
const db_token = require("./queriesToken");

app.use(morgan("dev"));
app.use(bodyParser.json());

app.use(
    cors({
        origin: "http://localhost:8080",
        methods: ["GET", "POST"],
        allowedHeaders: ["Content-Type", "Authorization"],
    })
);

app.get("/posts", db_post.getPosts);
app.get("/posts/:id", db_post.getPostById);
app.post("/posts", db_post.createPost);
app.put("/posts/:id", db_post.updatePost);
app.delete("/posts/:id", db_post.deletePost);

app.get("/tokens", db_token.getTokens);
app.post("/tokens/:token", db_token.createToken);

// Start server

app.listen(port, () => {
    console.log(`Servidor en ejecución en http://localhost:${port}`);
});
```

Server

```
const createPost = (request, response) => {
    const { title, body, img_url } = request.body;

    pool.query(
        "INSERT INTO posts (title, body, img_url) VALUES ($1, $2, $3)", [title, body, img_url],
        (error, results) => {
            if (error) {
                throw error;
            }
            response.status(201).send(`Post added with ID: ${results.insertId}`);
        }
    );
    setTimeout(() => {
        sendPushNotification({
            title: "Tu Publicación fue creada exitosamente!",
            body: "Eres todo un escritor! Wow!",
        });
    }, 5000);
};
```

CONCLUSIONES

Equipo 09