

Trabajar en ramas:

[01] - Siempre verificar tener la ultima actualizacion de la rama MASTER:

□ - | git checkout master (Movernos a la rama master)

□ - | git pull origin master (Me traigo todos los cambios)

[02] - Creo una nueva rama (Nombre/Feature/Objetivo)

□ - | git checkout -b Patricio/Feature/Login

[03] - Realizo los cambios

[¡ATENCIÓN!] - Solo realizar ÚNICAMENTE cambios en los archivos donde los demás no estén trabajando, si dos o más personas trabajan sobre el mismo luego traería problemas.

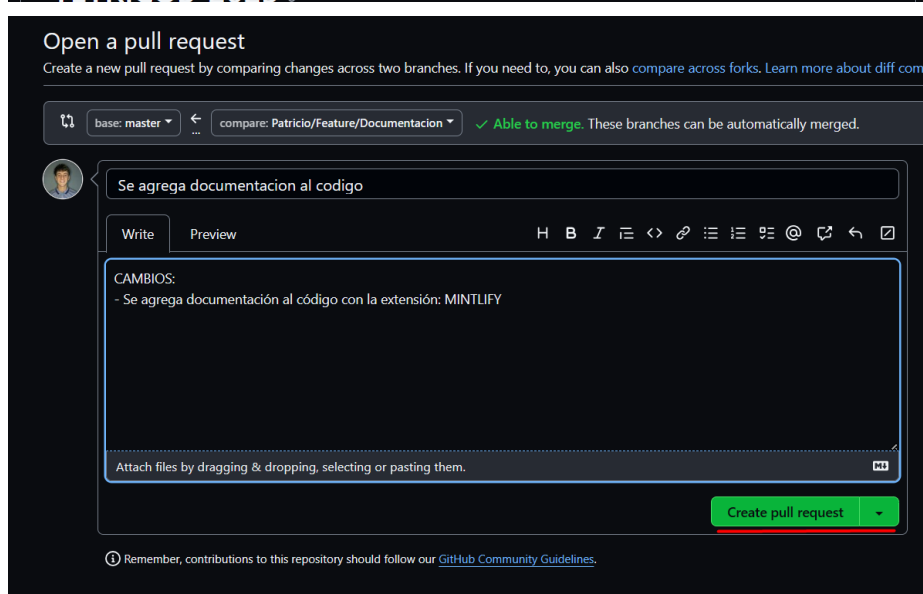
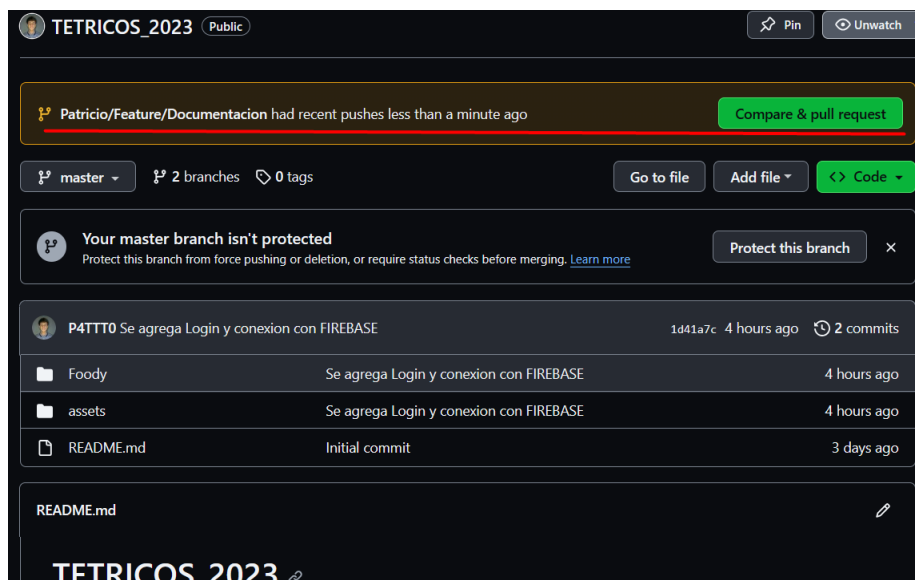
[04] - Envío los cambios

□ - | git add .

□ - | git commit -m "Descripción del commit"

□ - | git push origin <nombreDeLaRama>

[05] - Realizar PR



[06] - Esperar a que alguien revise el PR y realice el MERGE



¿Qué debo hacer si mi rama se ha trabajado con la MASTER desactualizada?

[01] - Vuelve a la rama master

`git checkout master`

[02] - Me actualizo la rama master

`git pull origin master`

[03] - Vuelves a tu rama

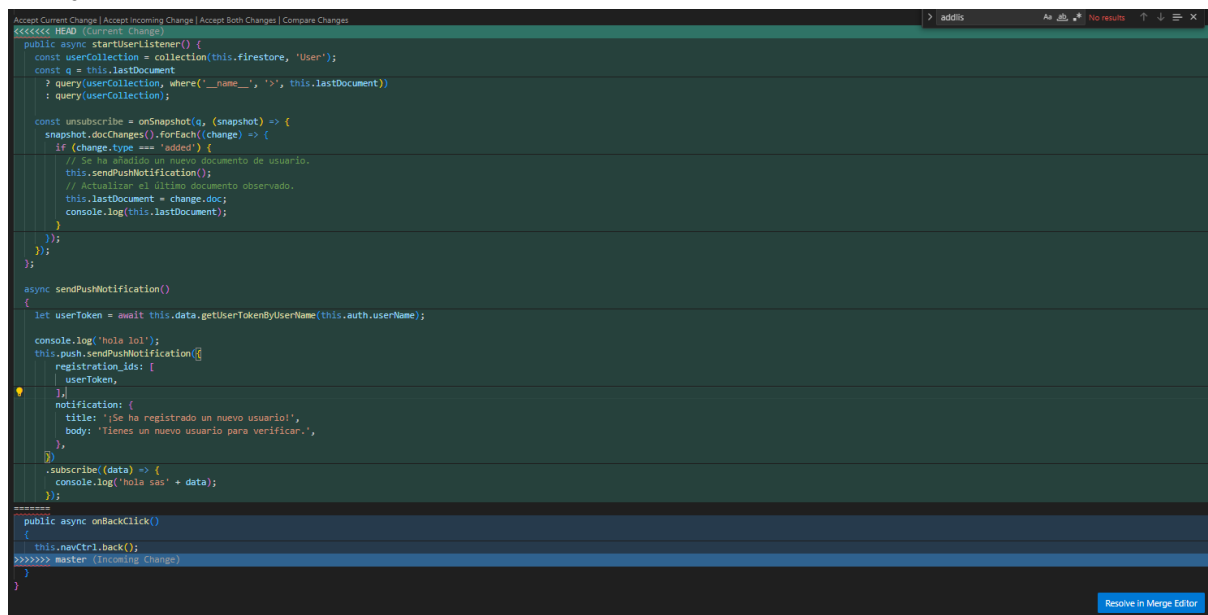
`git checkout rama`

[04] - Realizar el merge de tu rama con el master actualizado

`git merge master`

[!CUIDADO!] - Lo más probable es que al realizar el merge ocurran CONFLICTOS, pero estos no son errores, solo hay que especificar cómo deben quedar los archivos.

Por ejemplo, en este archivo:



Podemos observar lo verde que son mis cambios hechos en mi rama y lo azul los cambios traídos de la master, simplemente hay que especificar con que nos queremos quedar. Esto se puede hacer simplemente eliminando los textos agregados (Subrayados en amarillo):

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
public async startUserListener() {
  const userCollection = collection(this.firestore, 'User');
  const q = this.lastDocument
    ? query(userCollection, where('__name__', '>', this.lastDocument))
    : query(userCollection);

  const unsubscribe = onSnapshot(q, (snapshot) => {
    snapshot.docChanges().forEach((change) => {
      if (change.type === 'added') {
        // Se ha añadido un nuevo documento de usuario.
        this.sendPushNotification();
        // Actualizar el último documento observado.
        this.lastDocument = change.doc;
        console.log(this.lastDocument);
      }
    });
  });
};

async sendPushNotification()
{
  let userToken = await this.data.getUserTokenByUserName(this.auth.userName);

  console.log('hola lol');
  this.push.sendPushNotification({
    registration_ids: [
      userToken,
    ],
    notification: {
      title: '¡Se ha registrado un nuevo usuario!',
      body: 'Tienes un nuevo usuario para verificar.',
    },
  });

  .subscribe((data) => {
    console.log('hola sas' + data);
  });
}

=====
public async onBackClick()
{
  this.navCtrl.back();
}
>>>>>>> master (Incoming Change)
```

También se puede realizar desde el MERGE EDITOR. Del lado izquierdo veremos el archivo de la master, del lado derecho veremos el archivo de nuestra rama y por debajo el resultado del MERGE. En este caso yo quiero quedarme con ambas modificaciones, la de la master y la de mi rama, de esta forma hago click en “Accept Combination” y luego completo el merge. Esto debemos repetirlo con cada archivo que tenga conflictos.

MASTER

```
@Component({
  selector: 'app-admin-home',
  templateUrl: './admin-home.component.html',
  styleUrls: ['./admin-home.component.scss'],
})
export class AdminHomeComponent implements OnInit {

  constructor(private router : Router, private navCtrl : NavController) {}

  ngOnInit() {}

  public async onValidateUserClick() {
    {
      this.router.navigateByUrl('validar-usuario');
    }
  }
}
```

No Changes Accepted

CURRENT

```
import { ActivatedRoute } from '@angular/router';
import { AuthenticationService } from 'src/app/services/authentication.service';
import { DataService } from 'src/app/services/data.service';
import { PushNotificationService } from 'src/app/services/push-notifications.service';

@Component({
  selector: 'app-admin-home',
  templateUrl: './admin-home.component.html',
  styleUrls: ['./admin-home.component.scss'],
})
export class AdminHomeComponent implements OnInit {

  constructor(private router : Router, private navCtrl : NavController, private authService : AuthenticationService, private push : PushNotificationService, private dataService : DataService) {}

  private lastDocument : any;
  private notificationInvada : boolean = false;

  ngOnInit() {
    this.startUserListener();
  }

  public async onValidateUserClick() {
    {
      this.router.navigateByUrl('validar-usuario');
    }
  }
}
```

2 Conflicts Remaining

Complete Merge

RESULT

```
@Component({
  selector: 'app-admin-home',
  templateUrl: './admin-home.component.html',
  styleUrls: ['./admin-home.component.scss'],
})
export class AdminHomeComponent implements OnInit {

  constructor(private router : Router) {}

  ngOnInit() {
    this.startUserListener();
  }

  public async onValidateUserClick() {
    {
      this.router.navigateByUrl('validar-usuario');
    }
  }
}
```

No Changes Accepted