1. Många | använt Git | Sammarbeta | Hemifrån

För ett sånt här omfatade projekt och så här många programmerare har vi använt oss av versionhanteringsystemet Git, för att smidigt kunna sammarbeta och även jobba hemifrån.

2. Visar ändringar | Tillbaka version | Världens | Därför

Git visar alla ändringar som är gjord i programkoden under arbetsflödets gång. Det möjliggör även att gå tillbaka versioner samt ångra ändringar om det skulle krävas. Git är världens mest använda versionshanterare. Det var därför vi valde att arbeta med det, för att förbereda oss till arbetslivet.

3. Central server| Tillgängligt | Ladda upp | Ladda ned

Git använder sig av en central server för att centralisera projektet och göra det tillgängligt för flera användare, där varje individ som har tillgång till projektet kan ladda upp eller ladda hem ändringar.

4. Github| Gratis | Open-source | Månadskostnad

I vårat fall använde vi Github som central server. En fördel med Github är att det är gratis, men nackdelen är att projektet blir ”open source”, det vill säga att vem som helst har tillgång till källkoden. Detta kan man slippa om man betalar en månadskostnad.

5. Behaglig arbetsmiljö| Samma kod | Kollision | Två personer

Viktigast av allt var att Git skapade en behaglig arbetsmiljö för våran projektgrupp. Det underlättade väldigt mycket med att jobba med samma projekt. Git meddelar även om en kollision har uppstått, det vill säga att två personer har ändrat i samma kodrad.

6. Hur har vi använt det?

`git clone` - *Användes flitigt när fel uppstått iställer för git reset HEAD*

`git pull` - *Glömde ofta göra detta vilket resulterade i onödiga kollisioner vi git push*

`git status` - *Förstod ej skillnad på modified och untracked files, fick till med git clean -f*

`git add` - *Användes för ofta på alla filer vilket resulterade I att långa commit kommentarer*

`git commit` - *Dåliga kommentarer I början men blev succesivt bättre. Gjorde förbättring när vi upptäckte hur man backade versioner.*

`git push` - *Upptcäkte att det var bra att göra en pull innan push*

`git checkout` - *Gjordes i början på alla enskilda filer. För en vecka sedan uppäckte vi /**

`git clean -f` - *För att få bort oönskade untracked files*

7. Läskigt| Förstöra/Radera | Test.txt | Kollision

I början var det läskigt att använda alla komandon för man var rädd att man skulle förstöra saker i projektet eller att allt skulle försvinna av någon anledning. Började med ett testdokument i .txt format för att se hur kollisioner visades.

8. Kopia i början| Undivik kollisioner | Kommunikation | Messenger

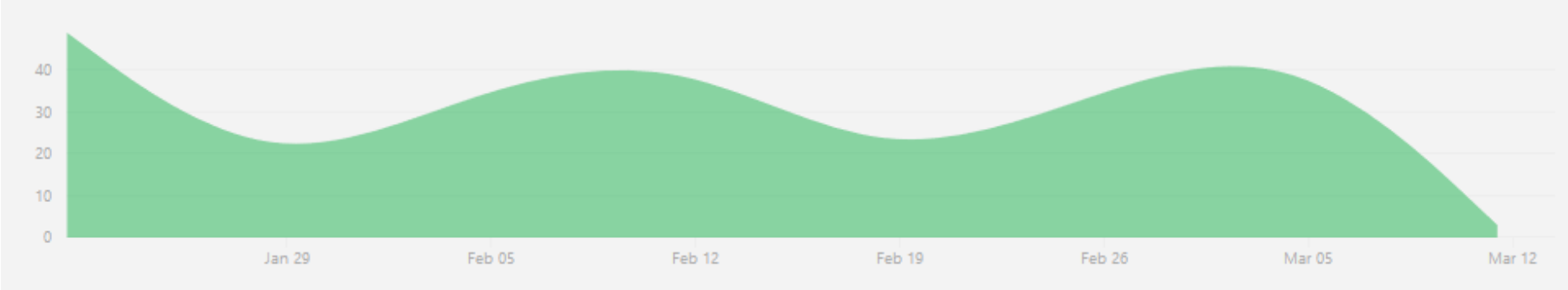
Vi gjorde även en kopia på hela projektet och expremimenterade med det innan vi lärde oss git. Kollisioner är något vi alltid försökt undvika genom att kommunicera med varandra då vi sitter bredvid varandra eller via facebook messenager.

9. Mycket att lära| Kollisioner | Forks | Tidigare versioner

Vi har fortfarande mycket att lära. Kollisioner är något vi fortfarande tycker är obehagligt. Vi vill bli bättre på att hantera kollisioner, börja använda forks osv gå tillbaka versioner.

10. Bild

Hur aktiv vi varit med commits. Dalar ner där vi fastnat men högsta peaken efter vi fick till CRUD till android med retrofit.



11. Slut

Gabriel ska gå genom retrofit.