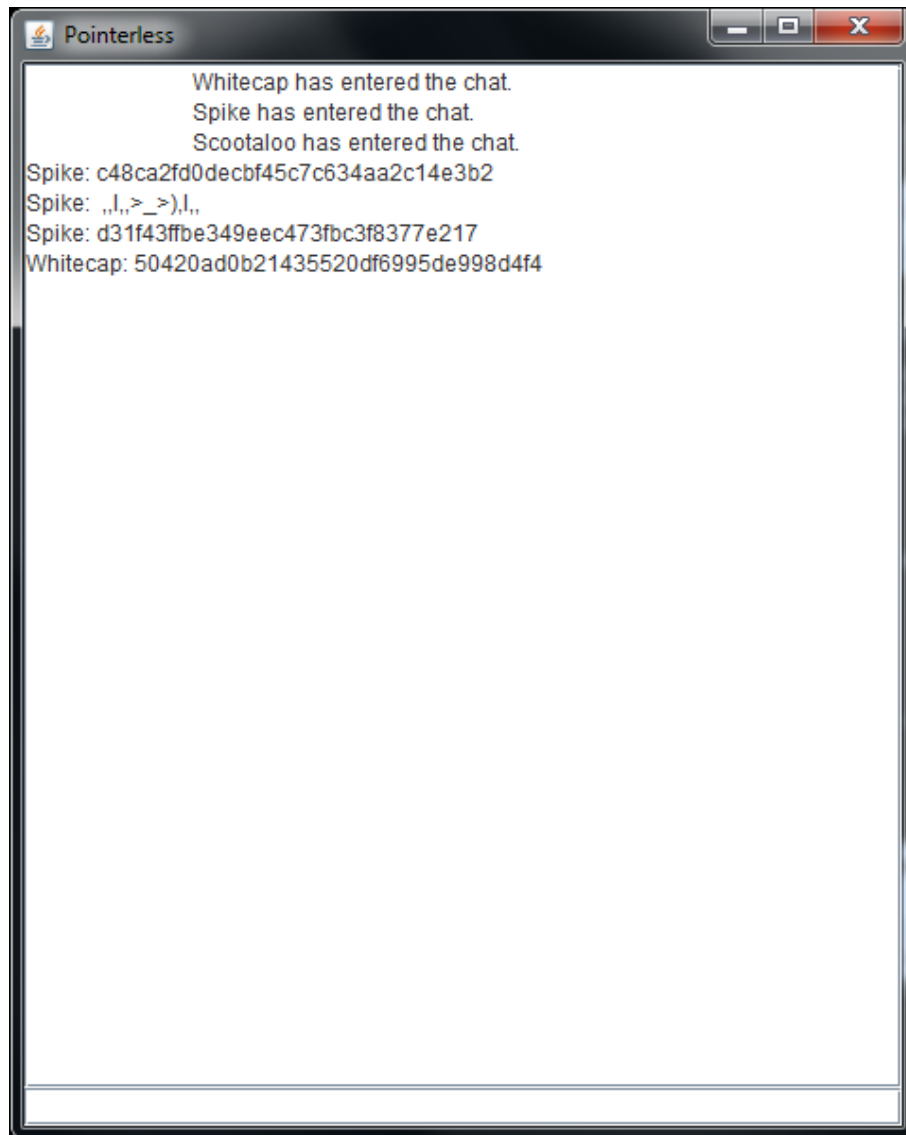


Pointerless



Innholdsfortegnelse

Abstract.....	2
Dokumentasjon av skjermflater med skjermdump	3
Hvorfor er programmet ubrukelig?	5
Hvorfor er programmet awesome?	5
Hvordan kjøres programmet?	5

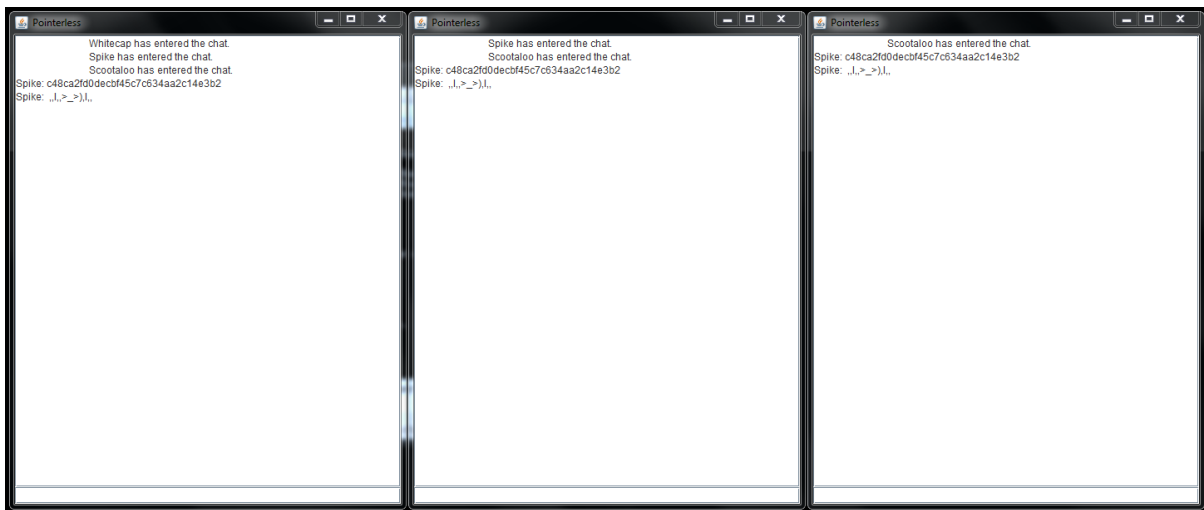
Abstract

Pointerless er, som navnet tilsier, både uten mening og pekere. Det er skrevet i Java. Som ikke har pointers. Geddit? Geddit?

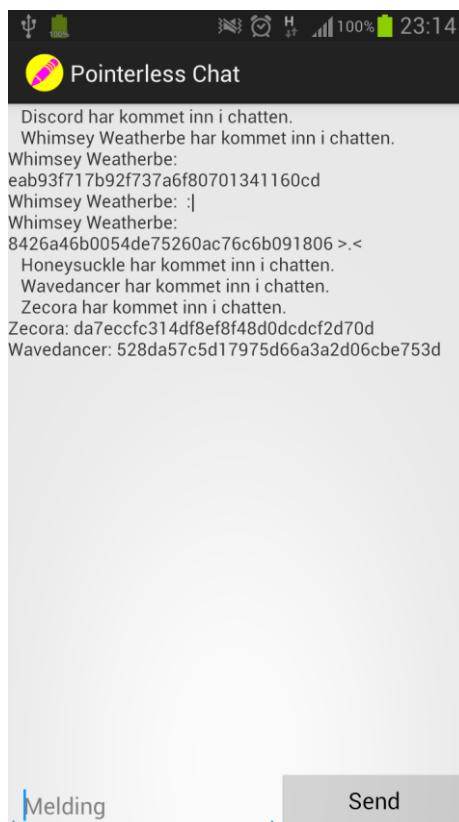
:D

Vi har skrevet en fullverdig chat server som kommuniserer med de tilkoblede klientene over TCP sockets, og fungerer mye på samme måte som en gruppesamtale i for eksempel Skype eller Windows Live Messenger. Vi har skrevet klienter til desktop og Android-enheter med støtte for alle API-nivåer fra og med 8 (versjon 2.2, alias Froyo). Dette dokumentet dokumenterer vårt bidrag til konkurransen Useless Utility i forbindelse med The Gathering 2013.

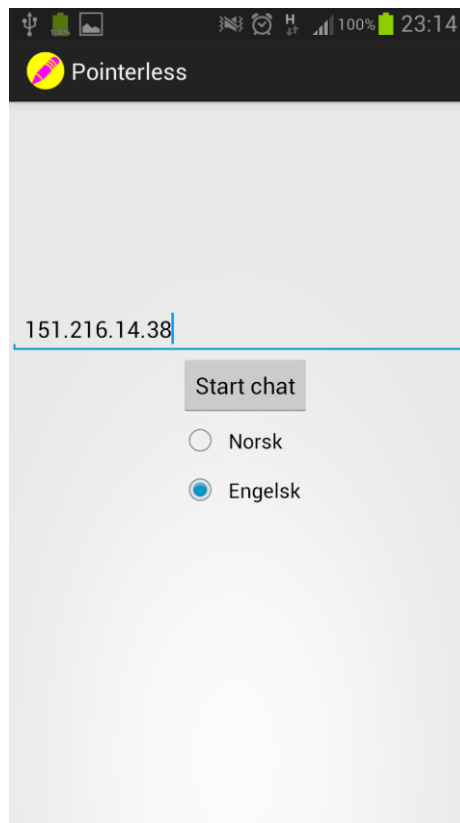
Dokumentasjon av skjermflater med skjermdump



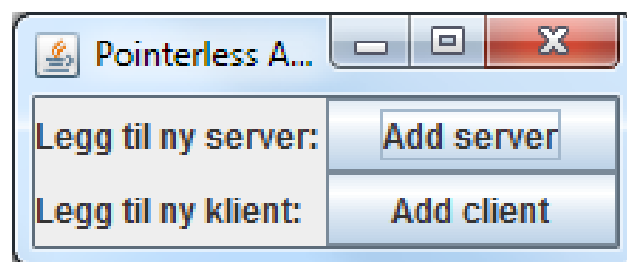
Chat-klienten er den eneste flaten i desktop-versjonen selve chat-deltakeren vil interagere med. Grensesnittet for desktop er simpelt: Det inneholder kun et stort tekstområde med støtte for manuell scrolling når og automatisk scrolling etter hvert som tekstfeltet fylles opp med tekst, og et lite input-felt der brukeren skriver inn sine meldinger (helt nederst) som kan sendes med Enter, og distribueres av serveren til alle de andre klientene.



Det enkle designet går igjen i Android-klienten, men dennes grensesnitt inneholder også en knapp for å sende meldinger.



Til Android har vi også skrevet en enkel «meny» som lar brukeren (les: tvinger brukeren til å) spesifisere serverens adresse, og bestemme språk. Hele desktop-applikasjonen er også internasjonalisert, men vi har ikke implementert støtte for å kjøre demoen med andre språk enn engelsk.



Vi har også skrevet et enkelt grensesnitt for å demonstrere programmet, ved å la brukeren legge til en ny server, og nye klienter ved trykk på henholdsvis «Add server» og «Add client».

Hvorfor er programmet ubrukelig?

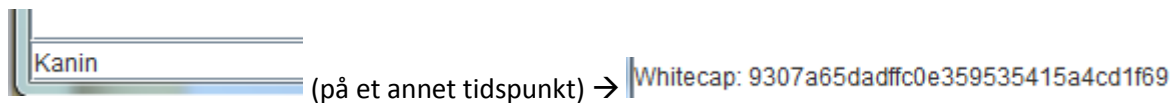
... og det er virkelig ubrukelig.

For det første: alle meldinger som sendes blir slått sammen med en pseudotilfeldig generert salt på 30 tegn og hashet med MD5, som (fordi hashing er en enveisfunksjon) gjør meldingene fullstendig umulig (les: unfeasible) å tolke, og ødelegger hele poenget med å ha et chatsystem.



A screenshot of a chat window. On the left, a text input field contains the name 'Kanin'. To the right of the input field, an arrow points to the text 'Enter/Return', which in turn points to a 'Whitecap:' label followed by a long, random alphanumeric string: '6f4283a872b9b6325fe4604182963717'.

... og for å demonstrere den tilfeldige salten:

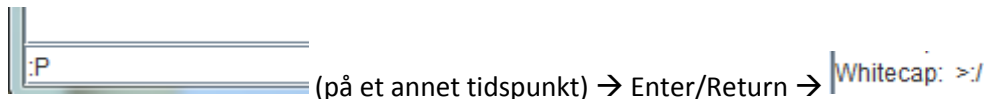


A screenshot of a chat window, similar to the one above. The input field contains 'Kanin'. An arrow points from the input area to the text '(på et annet tidspunkt) →', which then points to a 'Whitecap:' label followed by a different random alphanumeric string: '9307a65dadffc0e359535415a4cd1f69'.

For det andre: alle (med forbehold) emoticons blir byttet ut med et annet (pseudotilfeldig) emoticon. Hvis det legges inn flere emoticons i samme melding, vil teksten mellom hvert emoticon saltes og hashes separat, og hvert enkelt emoticon byttes ut.



A screenshot of a chat window. The input field contains the emoticon ':P'. An arrow points from the input area to the text 'Enter/Return →', which then points to a 'Whitecap:' label followed by the text '(o_o)'.



A screenshot of a chat window. The input field contains the text ':P'. An arrow points from the input area to the text '(på et annet tidspunkt) → Enter/Return →', which then points to a 'Whitecap:' label followed by the text '>/'.

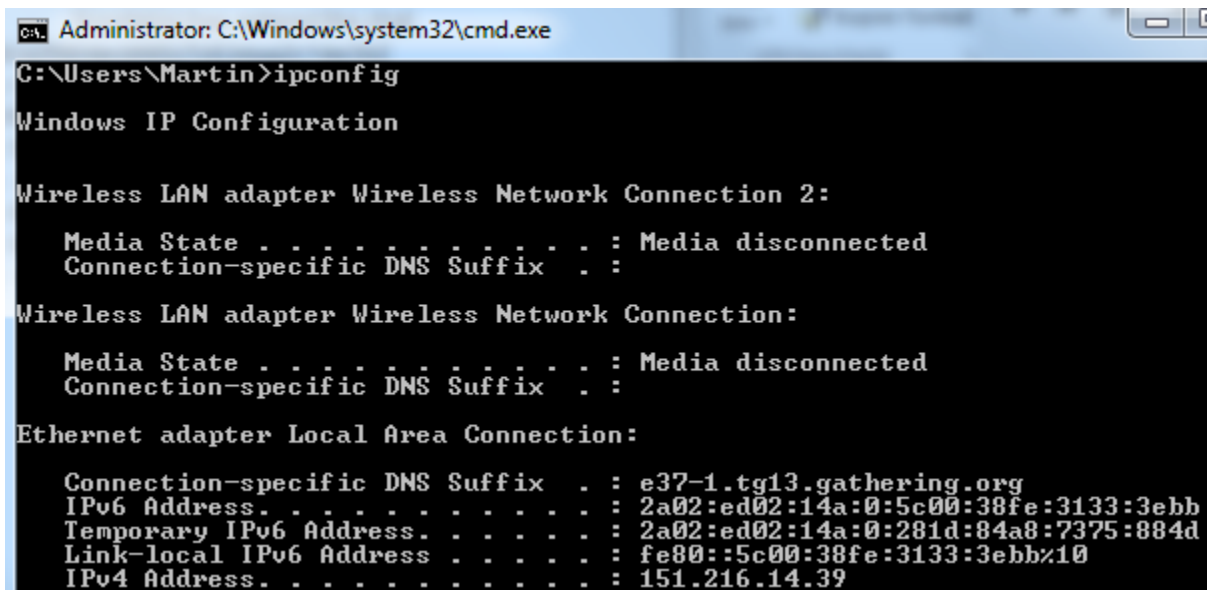
Hvorfor er programmet awesome?

- Systemet for navngiving av klienter. Klientene tildeles automatisk et navn fra My Little Pony når de kobler seg til serveren. :D
- [Det faktum at vi bygget hele løsningen på under 24 timer.](#)

Hvordan kjøres programmet?

Serveren kompilerer til å kjøre på localhost (IPv4: 127.0.0.1; IPv6: ::1), og desktop-klientene vil også kjøre her. Android-appen har derimot nødvendigvis en annen IP, og kan ikke koble seg til localhost. Derfor må brukeren av appen skrive inn IP-adressen til serveren, som kan finnes ved å kjøre

Kommandoen ipconfig (Windows) eller ifconfig (Linux/OSX) på maskinen som har serveren kjørende.



```
C:\Users\Martin>ipconfig

Windows IP Configuration

Wireless LAN adapter Wireless Network Connection 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : e37-1.tg13.gathering.org
    IPv6 Address. . . . . : 2a02:ed02:14a:0:5c00:38fe:3133:3ebb
    Temporary IPv6 Address. . . . . : 2a02:ed02:14a:0:281d:84a8:7375:884d
    Link-local IPv6 Address . . . . . : fe80::5c00:38fe:3133:3ebb%10
    IPv4 Address. . . . . : 151.216.14.39
```

Vi utnytter dette oppsettet, og har forberedt en .jar-fil som kjører i gang programmet uten mer om og men. Dersom dette skulle feile, har vi også skrevet en batch-fil som kjører selve klassefilene i stedet for å støtte seg på en jar-fil.

Det er per nå ikke mulig å stoppe serveren igjen uten å drepe prosessen, men det går fint å forsøke å starte den flere ganger, da bare den første launch-en vil være gjeldende.