N.E.S.T.

Release 0.1.0

Gruppo 2

30 mag 2021

Manuale utente

Benvenuto alla documentazione di N.E.S.T.!

CAPITOLO 1

Il progetto in breve

N.E.S.T. (Noi Estraiamo Statistiche Tweet) è un progetto realizzato nel 2021 per l'**esame di Progetto del Software** del corso di Informatica all'Unimore.

1.1 Suddivisione in moduli

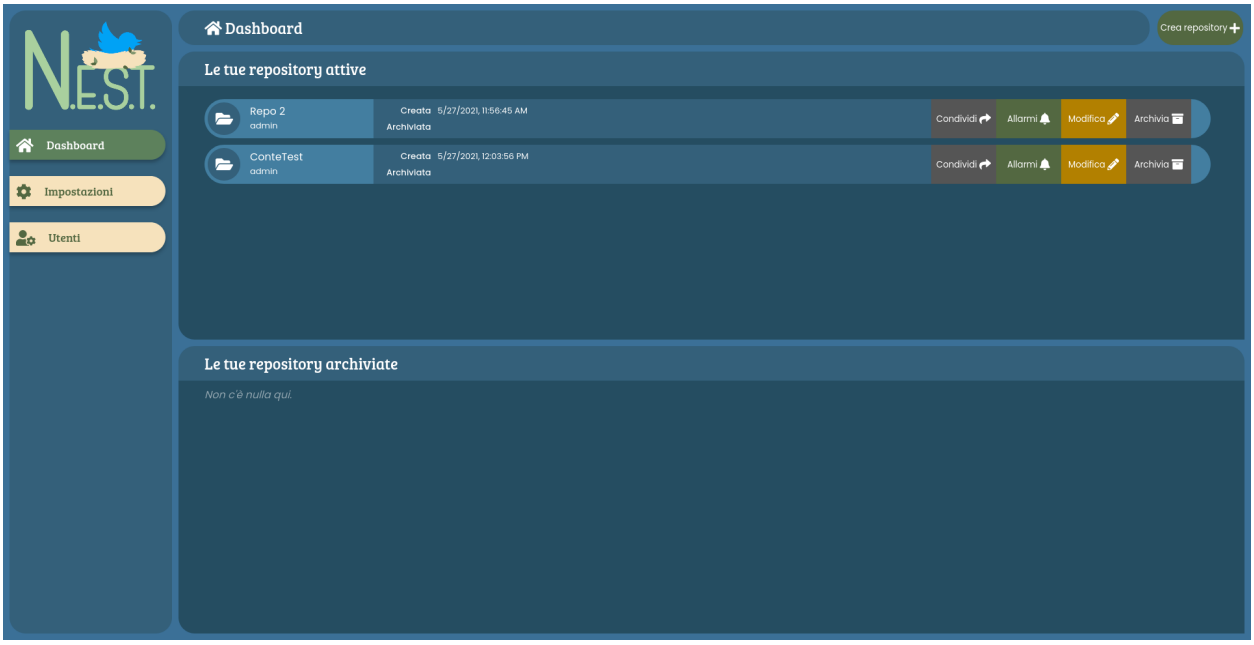
Il progetto è composto da tre parti:

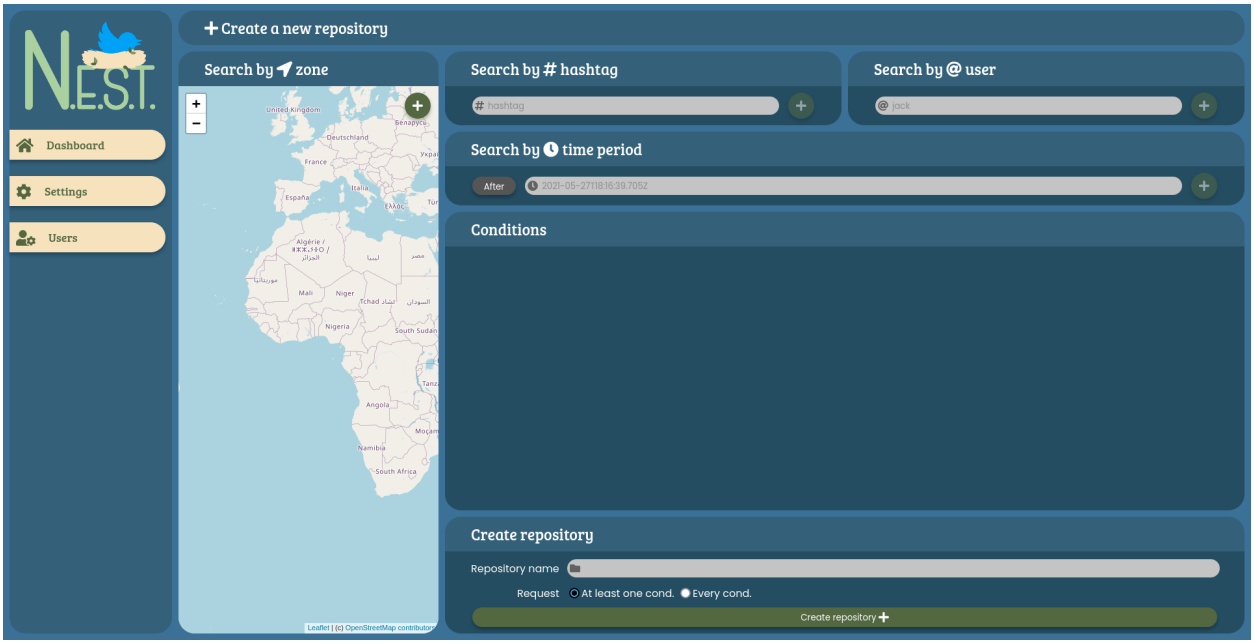
nest_crawler Un modulo scritto in Python usando `tweepy` che recupera tweet attraverso la Twitter API secondo le condizioni presenti all'interno del database e li salva per future elaborazioni.

nest_backend Un web server scritto in Python usando il framework `flask` che fornisce un'API HTTP per visualizzare e manipolare i dati dell'applicazione, gestendo autenticazione, autorizzazione e comunicazione con il database.

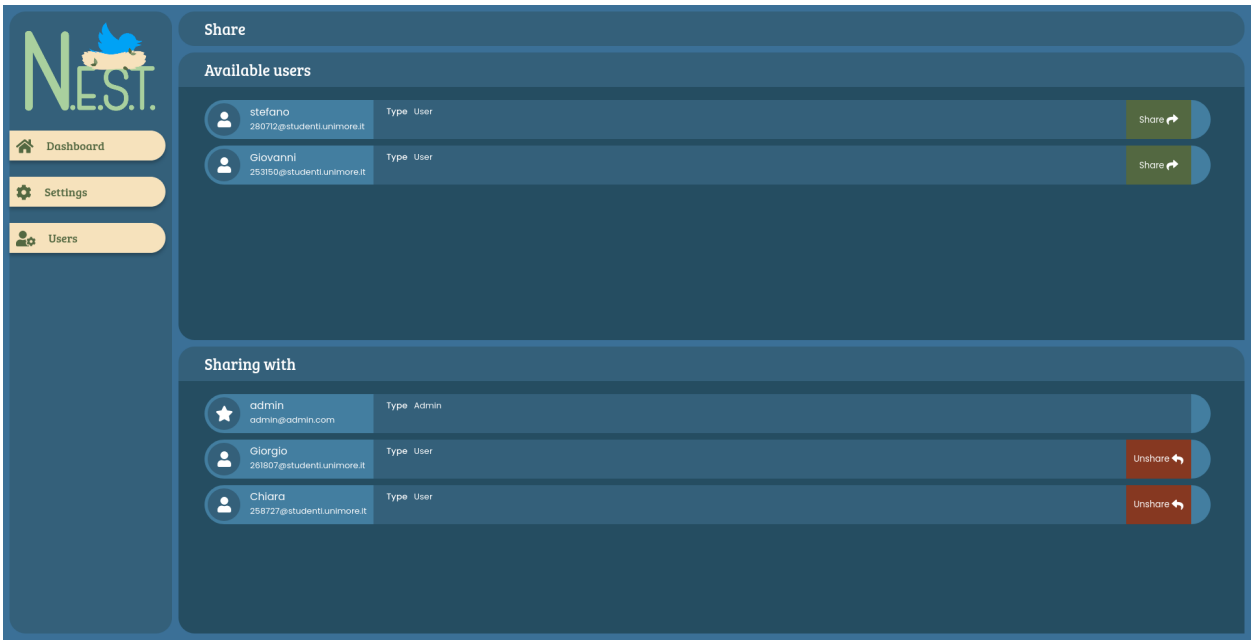
nest_frontend Una applicazione web scritta in JSX usando il framework `react` che comunica con il backend, mostrando all'utente i dati del backend in una formato immediatamente comprensibile.

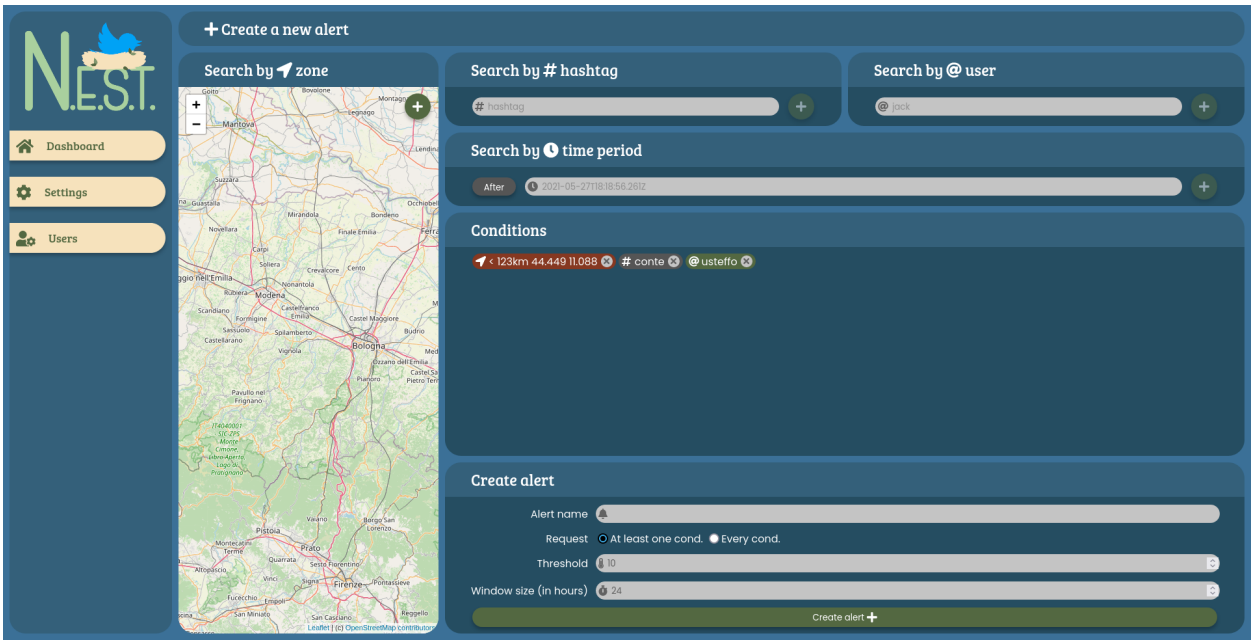
1.2 Screenshots






N.E.S.T.
Dashboard
Impostazioni
Utenti
ConteTest
Tweet
@yucd53250131
5/27/2021, 12:03:44 PM
RT @juvinsight: #Leonardo non va d'accordo con Mauricio #Pochettino, per questo Daniel #Levy vorrebbe riportare l'argentino al #Tottenham....
@Libero_official
5/27/2021, 12:03:40 PM
#Inter, "serve il cash". Dopo #Conte finiscono sul mercato i big: possono partire #Lautaro Martinez e Alessandro... https://t.co/rtjrrOxo0
@Sonoinnocentema
5/27/2021, 12:03:37 PM
RT @AlessandraOdri: La vedova inconsolabile #Travaglio, livoroso con tutti, tranne che con gli amici del M5S, ieri ha detto che Draghi si è...
@yutomanga
5/27/2021, 12:03:26 PM
RT @juvinsight: #Leonardo non va d'accordo con Mauricio #Pochettino, per questo Daniel #Levy vorrebbe riportare l'argentino al #Tottenham....
@poppy__seed_
5/27/2021, 12:03:26 PM
If Conte is gonna be Real Madrid manager...the assistant manager needs to equally good as Conte may be sent off for... https://t.co/306Kymnncg
@andal68
5/27/2021, 12:03:26 PM
Torna dopo 2 anni #Allegri. Ero infelice quando sostituì #Conte e sono infelice ora che sostituisce #Pirlo. Questio... https://t.co/800qAbDDKK
@___breeh
5/27/2021, 12:03:17 PM
With this situation, it's possible for #Conte to #RealMadrid , I think. Raul also on the list. https://t.co/sTYeMoAa5S
@ilnonnosimpson
5/27/2021, 12:03:16 PM
RT @capuanogio: ⚠️ Il #Psg tenta #Hakimi e l'offerta da 50 milioni può far vacillare l'#Inter. Possibile anche l'inserimento di #Paredes ab...
@xxx_diamante
5/27/2021, 12:03:15 PM
RT @FabrizioRomano: Official. Antonio Conte leaves Inter with immediate effect, official statement confirms. #Conte #Inter
Filtri
Wordcloud
infelice
daniel
riportare
mauricio
l'argentino
dè
@juvinsight:
#pochettino
d'accordo
#levy
conte
#leonardo
vorrebbe
va
dopo
#conte
#tottenham....
Ricerca per # hashtag
hashtag


N.E.S.T.
Dashboard
Settings
Users
Share
Available users
stefano
280712@studenti.unimore.it
Type User
Share
Giovanni
253150@studenti.unimore.it
Type User
Share
Sharing with
admin
admin@admin.com
Type Admin
Giorgio
261807@studenti.unimore.it
Type User
Unshare
Chiara
258727@studenti.unimore.it
Type User
Unshare


N.E.S.T.
Dashboard
Settings
Users
Create a new alert
Search by zone
Search by # hashtag
hashtag
Search by @ user
jack
Search by time period
After
2021-05-27T18:18:56.261Z
Conditions
< 123km 44.449 11.088
conte
@ usteffo
Create alert
Alert name
Request
At least one cond.
Every cond.
Threshold
10
Window size (in hours)
24
Create alert

CAPITOLO 2

Installazione

Questa guida illustra come installare interamente N.E.S.T. su un server Linux.

2.1 Prerequisiti

Per installare ed eseguire N.E.S.T., è necessario:

- Una connessione a Internet
- Un sistema operativo Linux-based (preferibilmente Arch Linux)
- SystemD ^248.2
- Apache HTTP Server ^2.4.46
- PostgreSQL ^13.2
- Git ^2.31.1
- Python ^3.8
- Poetry ^1.0
- NodeJS ^16.0
- npm ^7.13.0
- Un mail server (interno o esterno) che supporti l'SMTP

2.2 Creare un nuovo utente

Per motivi di sicurezza, si suggerisce di creare un nuovo utente con il quale eseguire il progetto:

Nota: È necessario essere amministratori di sistema per eseguire i seguenti comandi. Si veda il manuale di useradd per più dettagli.

```
root:~# mkdir --parents /srv/nest
root:~# useradd --home-dir /srv/nest --shell /bin/bash nest
root:~# chown --recursive nest: /srv/nest
```

2.3 Scaricare il codice sorgente

Per installare N.E.S.T., è necessario avere il codice sorgente disponibile sul server.

Si consiglia di scaricarlo tramite *Git*:

```
nest:~$ git clone https://gitlab.steffo.eu/nest/g2-progetto.git
```

Questo creerà una nuova cartella `g2-progetto` nella directory in cui è stato eseguito il comando.

Per proseguire, sarà necessario entrarvi:

```
nest:~$ cd g2-progetto
```

2.4 Creare il database

N.E.S.T. necessita di un database PostgreSQL in cui salvare i dati.

Per motivi di sicurezza, si suggerisce di creare un ruolo isolato dal resto del DBMS apposta per N.E.S.T.:

```
postgres:~$ createuser nest
```

Per creare il database PostgreSQL, si esegua:

```
postgres:~$ createdb --owner=nest nest
```

2.5 Creare un file di configurazione per il backend

Il backend usa un file di configurazione per impostare alcune variabili.

Si crei un nuovo file nella working directory del progetto denominato `config.py`:

```
nest:~/g2-progetto$ vim config.py
```

Il file dovrà avere i seguenti contenuti:

```
# Una stringa rappresentante il database da utilizzare
# Per maggiori informazioni sulla sintassi, si veda https://docs.sqlalchemy.org/en/14/
↪core/engines.html
SQLALCHEMY_DATABASE_URI = "postgresql://nest@/nest"

# Una stringa casuale utilizzata per generare i JSON Web Token (JWT)
# Va mantenuta segreta e costante per tutta l'operazione del backend!
# Si suggerisce di premere tasti casuali sulla tastiera finchè la riga non è piena.
SECRET_KEY =
↪"dsjiofgvinmodfiojvbnio3erfnoiweraqugu43ghjwrevniuwerng43iugnreuwignhritmj43i43nb8i42ug0wevkwovmwi
↪"
```

2.6 Installare le dipendenze Python

Le dipendenze Python sono gestite da *Poetry*, e possono essere installate con:

```
nest:~/g2-progetto$ poetry install
```

Poetry creerà automaticamente un venv e vi installerà all'interno tutti i pacchetti necessari all'esecuzione del backend e del crawler di N.E.S.T. .

Si suggerisce di ricordare il nome del venv creato da Poetry, in quanto sarà necessario per *Creare un servizio SystemD per il backend*:

```
Creating virtualenv nest-7C2fm2VD-py3.9 in /srv/nest/.cache/pypoetry/virtualenvs
```

2.7 Installare le dipendenze NodeJS

Le dipendenze NodeJS sono gestite da *npm*, e possono essere installate con:

```
nest:~/g2-progetto$ npm install
```

npm creerà automaticamente una cartella node_modules e vi installerà all'interno tutte le librerie necessarie all'esecuzione del frontend di N.E.S.T. .

2.8 Creare un servizio SystemD per il backend

Per fare in modo che il backend rimanga attivo in background, anche dopo un riavvio, si suggerisce di installarlo come servizio di sistema di *SystemD*:

```
root:~# systemctl edit --force --full nest-backend
```

Inserire all'interno del file le seguenti direttive:

```
[Unit]
Description=N.E.S.T. Backend
Wants=network-online.target postgresql.service
After=network-online.target nss-lookup.target postgresql.service

[Service]
```

(continues on next page)

(continua dalla pagina precedente)

```
Type=exec
User=nest
Group=nest
WorkingDirectory=/srv/nest/g2-progetto

# Si sostituisca a questo il percorso del virtualenv creato in precedenza da Poetry
#          ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
ExecStart=/srv/nest/.cache/pypoetry/virtualenvs/nest-7C2fm2VD-py3.9/bin/python -m␣
↪gunicorn -b 127.0.0.1:30040 --env="FLASK_CONFIG=../config.py" nest_backend.app:rp_
↪app

[Install]
WantedBy=multi-user.target
```

Ora, si verifichi che il servizio si avvii correttamente eseguendolo manualmente con:

```
root:~# systemctl start nest-backend
```

In caso di successo, l'API dovrebbe essere esposto sulla porta `30040` dell'indirizzo di loopback `127.0.0.1`:

```
root:~# curl 127.0.0.1:30040/doa
If you see this, the server is fine.
```

Si abiliti il servizio, in modo che venga automaticamente avviato al riavvio del sistema:

```
root:~# systemctl enable nest-backend
```

2.9 Compilare il frontend

Perchè sia possibile servire il frontend agli utenti, è necessario prima crearne una versione compilata ottimizzata.

È possibile farlo con il comando:

```
nest:~/g2-progetto$ npm run build
```

Verrà creata una cartella `build` con all'interno la versione compilata.

2.10 Creare un servizio SystemD per il frontend

Per rendere disponibile alla rete la copia locale del frontend, si suggerisce di avviare lo script npm `serve` integrato con N.E.S.T. come un servizio di sistema di *SystemD*:

```
root:~# systemctl edit --force --full nest-frontend
```

Inserire all'interno del file le seguenti direttive:

```
[Unit]
Description=N.E.S.T. Frontend
Wants=network-online.target nest-backend.service
After=network-online.target nss-lookup.target nest-backend.service

[Service]
```

(continues on next page)

(continua dalla pagina precedente)

```
Type=exec
Environment=NODE_ENV=production
User=nest
Group=nest
WorkingDirectory=/srv/nest/g2-progetto
ExecStart=/usr/bin/npm run serve

[Install]
WantedBy=multi-user.target
```

Avvertimento: Questo file non è stato testato, in quanto sul server demo è in uso una versione più complessa che usa nvm per gestire più versioni di NodeJS sullo stesso sistema.

La versione in uso sul server demo è:

```
[Unit]
Description=N.E.S.T. Frontend
Wants=network-online.target nest-backend.service
After=network-online.target nss-lookup.target nest-backend.service

[Service]
Type=exec
Environment=NODE_ENV=production
Environment=NODE_VERSION=16
User=nest
Group=nest
WorkingDirectory=/srv/nest/g2-progetto
ExecStart=/srv/nest/.nvm/nvm-exec npm run serve

[Install]
WantedBy=multi-user.target
```

Ora, si verifichi che il servizio si avvii correttamente eseguendolo manualmente con:

```
root:~# systemctl start nest-frontend
```

In caso di successo, il frontend dovrebbe essere esposto sulla porta `30041` dell'indirizzo di loopback `127.0.0.1`:

```
root:~# curl 127.0.0.1:30041
[...]
```

Si abiliti il servizio, in modo che venga automaticamente avviato al riavvio del sistema:

```
root:~# systemctl enable nest-frontend
```

2.11 Creare un servizio SystemD per il crawler

Perchè i repository vengano popolati di Tweet, è necessario configurare il crawler come servizio di *SystemD*:

```
root:~# systemctl edit --force --full nest-crawler
```

All'interno del file, inserire le seguenti direttive:

```
[Unit]
Description=N.E.S.T. Crawler
Wants=network-online.target nest-backend.service
After=network-online.target nss-lookup.target nest-backend.service

[Service]
Type=exec
Environment=FLASK_CONFIG=../config.py
User=nest
Group=nest
WorkingDirectory=/srv/nest/g2-progetto
# Si sostituisca a questo il percorso del virtualenv creato in precedenza da Poetry
#          ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
ExecStart=/srv/nest/.cache/pypoetry/virtualenvs/nest-7C2fm2VD-py3.9/bin/python -m␣
↪nest_crawler

[Install]
WantedBy=multi-user.target
```

2.12 Configurare il crawler

Nota: Per utilizzare gli API di Twitter, è necessario essere approvati dal supporto tecnico di Twitter.

È dunque necessario fare richiesta, e sarà possibile procedere con l'installazione solo una volta ricevute le credenziali per l'utilizzo.

Per impostare le variabili di ambiente richieste dal crawler, si suggerisce di creare un *file di override* di SystemD:

```
root:~# systemctl edit nest-crawler
```

All'interno del file, inserire le seguenti direttive:

```
[Service]

# Sostituire a questi caratteri la Consumer Key ricevuta da Twitter
#                ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Environment=C_K=00000000000000000000000


# Sostituire a questi caratteri il Consumer Secret ricevuto da Twitter
#                ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Environment=C_S=0000000000000000000000000000000000000000000000000000

# Sostituire a questi caratteri l'Access Token ricevuto da Twitter
#                ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
```

(continues on next page)

(continua dalla pagina precedente)

```
Environment=A_T=0000000000000000000000000000000000000000000000000

# Sostituire a questi caratteri l'Access Token Secret ricevuto da Twitter
#                 ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Environment=A_T_S=0000000000000000000000000000000000000000000000000

# Sostituire con l'indirizzo del proprio SMTP mail server
#                      ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Environment=SMTP_HOST=mail.gandi.net

# Sostituire con le proprie credenziali dell'SMTP mail server
#                           ↓↓↓↓↓↓↓↓↓↓↓↓
Environment=SMTP_USERNAME=bot@ryg.one
#                           ↓↓↓↓↓↓↓↓
Environment=SMTP_PASSWORD=password

# Sostituire con l'email da cui si desidera che vengano inviate le allerte
#                             ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
Environment=SMTP_FROM_EMAIL=nest-bot@ryg.one
```

Ora, si verifichi che il servizio si avvii correttamente eseguendolo manualmente con:

```
root:~# systemctl start nest-crawler
```

Nel log di sistema non dovrebbe comparire nessun errore:

```
root:~# journalctl nest-crawler
```

2.13 Creare un timer SystemD per il crawler

Per fare in modo che il crawler venga eseguito periodicamente, si suggerisce di configurare un timer SystemD:

```
root:~# systemctl edit --force --full nest-crawler.timer
```

Si inseriscano all'interno del file le seguenti direttive:

```
[Unit]
Description=Run nest-crawler every 60 minutes

[Timer]
OnBootSec=60min
OnUnitActiveSec=60min
Unit=nest-crawler.service

[Install]
WantedBy=timers.target
```

Ora, si verifichi che il timer si avvii correttamente eseguendolo manualmente con:

```
root:~# systemctl start nest-crawler.timer
```

Nello stato del timer non dovrebbe comparire nessun errore:

```
root:~# systemctl status nest-crawler.timer
```

Si abiliti il timer, in modo che venga automaticamente avviato al riavvio del sistema:

```
root:~# systemctl enable nest-crawler.timer
```

2.14 Configurare Apache come reverse proxy

Per rendere l'API e il frontend disponibili al pubblico, si suggerisce di configurare Apache HTTP Server come reverse proxy.

La configurazione di Apache varia molto da distribuzione a distribuzione Linux, e talvolta anche da server a server; pertanto, si fornisce solamente un file VirtualHost di esempio da adattare al proprio setup:

```
<VirtualHost *:80>
    ServerName "api.nest.steffo.eu"
    ServerName "prod.nest.steffo.eu"

    RewriteEngine On
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>

<VirtualHost *:443>
    ServerName "api.nest.steffo.eu"

    SSLEngine on
    SSLCertificateFile      "/root/.acme.sh/*.nest.steffo.eu/fullchain.cer"
    SSLCertificateKeyFile   "/root/.acme.sh/*.nest.steffo.eu/*.nest.steffo.eu.key"

    ProxyPass           "/" "http://127.0.0.1:30040/"
    ProxyPassReverse    "/" "http://127.0.0.1:30040/"
    RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}

    Protocols h2 http/1.1
    Header always set Strict-Transport-Security "max-age=63072000"
</VirtualHost>

<VirtualHost *:443>
    ServerName "prod.nest.steffo.eu"

    SSLEngine on
    SSLCertificateFile      "/root/.acme.sh/*.nest.steffo.eu/fullchain.cer"
    SSLCertificateKeyFile   "/root/.acme.sh/*.nest.steffo.eu/*.nest.steffo.eu.key"

    ProxyPass           "/" "http://127.0.0.1:30041/"
    ProxyPassReverse    "/" "http://127.0.0.1:30041/"
    RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}

    Protocols h2 http/1.1
    Header always set Strict-Transport-Security "max-age=63072000"
</VirtualHost>
```

CAPITOLO 3

Aggiornamento

Per scaricare gli aggiornamenti di N.E.S.T. si esegua il seguente comando nella directory del codice sorgente:

```
nest:g2-progetto$ git pull
```

Si aggiornino poi tutte le dipendenze:

```
nest:g2-progetto$ poetry install
nest:g2-progetto$ npm install
```

Si ricompili il frontend:

```
nest:g2-progetto$ npm build
```

In seguito, si riavviino tutti i servizi di N.E.S.T.:

```
root:~# systemctl restart nest-frontend nest-backend nest-crawler nest-crawler.timer
```

Si verifichi infine il corretto avvio di tutti i servizi:

```
root:~# systemctl status nest-frontend nest-backend nest-crawler nest-crawler.timer
```

CAPITOLO 4

Introduzione

4.1 Obiettivo

L'obiettivo del progetto è la creazione di un software per fornire l'aggregazione e l'analisi di Tweet, in modo da rilevare eventi *macroscopici*, *locali* o più semplicemente filtrarli in base a delle *keyword*.

Il prodotto sarà utilizzato dal cliente e da un piccolo gruppo di suoi dipendenti per effettuare ricerche statistiche.

Il software andrà ad integrarsi direttamente con Twitter, da cui verranno raccolti dati e su cui verranno pubblicate allerte su di essi.

4.2 Campo di applicazione

Il software trova utilizzo principalmente in **ambito statistico**, essendo il suo scopo quello di raccogliere dati e permettere di analizzarli tramite un'interfaccia grafica.

4.3 Caratteristiche degli utenti

Il software potrà essere utilizzato da utenti con una discreta esperienza nell'analisi di dati ma senza particolari conoscenze informatiche.

4.4 Glossario

Repository Raccolta di tweet che soddisfano determinate condizioni.

Condizione Predicato logico che deve essere soddisfatto da un tweet per essere raccolto in fase di raccolta dati, o per essere contato in fase di allertamento utente.

Filtro Predicato logico che deve essere soddisfatto da un tweet per essere visualizzato in fase di analisi dati.

Allarme Notifica inviata all'utente attraverso un mezzo telematico, come email oppure un tweet.

Utente Utilizzatore del software con un proprio account creato dall'amministratore della piattaforma.

In particolare, la piattaforma prevederà due tipologie di utenti:

Utente regolare Potranno eseguire attività di creazione, analisi, condivisione, archiviazione ed eliminazione dei propri repository.

Utente amministratore Potrà effettuare tutte le attività dell'utente regolare, e in aggiunta potrà creare ed eliminare nuovi utenti regolari.

4.5 Macro-funzionalità

Il software permetterà di selezionare **condizioni** con cui scegliere quali tweet raccogliere:

- in base ai loro hashtag
- in base al loro autore
- in base alla loro posizione geografica (ove presente)
- in base alla loro data di pubblicazione

Selezionate le condizioni, l'utente potrà creare una **repository**: una cartella in cui verranno raccolti i tweet soddisfacenti le condizioni richieste.

Una volta raccolti, i tweet di una repository potranno essere **analizzati** in qualsiasi momento: durante l'analisi, saranno mostrate statistiche e grafici relativi ai tweet.

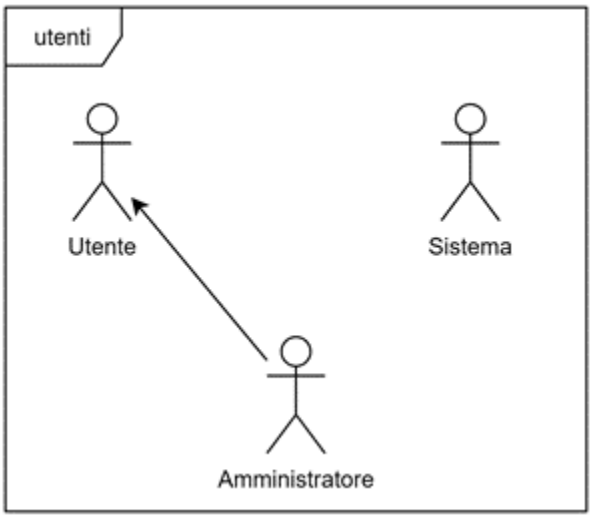
La raccolta potrà essere interrotta in qualsiasi momento **archiviando** il repository.

Sarà possibile **condividere** una repository con altri utenti della piattaforma, permettendo loro di analizzarla.

Infine, l'utente potrà configurare una repository in modo che gli invii una **allerta** qualora vengano raccolti un dato numero di tweet in una certa *finestra temporale*.

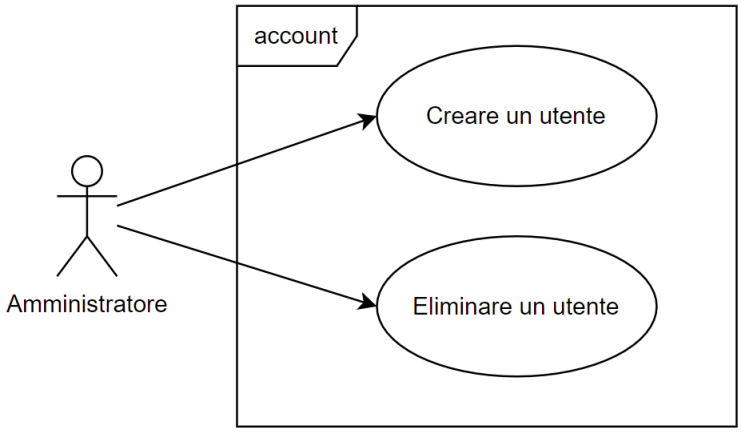
4.6 Casi d'uso

N.E.S.T. prevede tre tipologie di *agenti* ("utenti" UML): **utente**, **amministratore** e **sistema**.

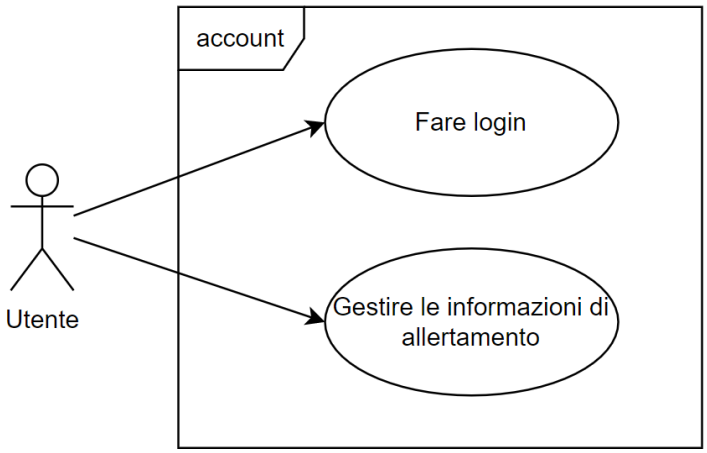


I principali casi d'uso individuati durante la progettazione di N.E.S.T. sono:

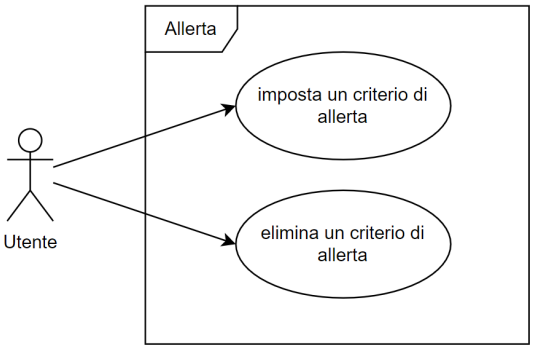
- La gestione degli utenti da parte di un Amministratore:

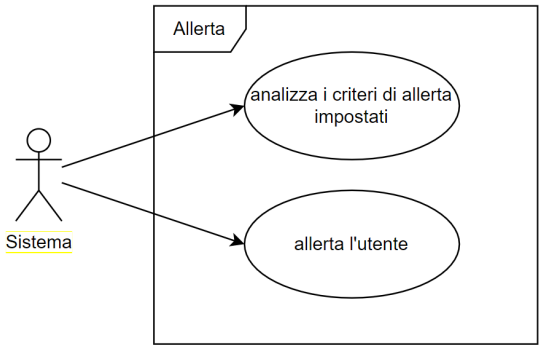


- La gestione del login da parte di un Utente:

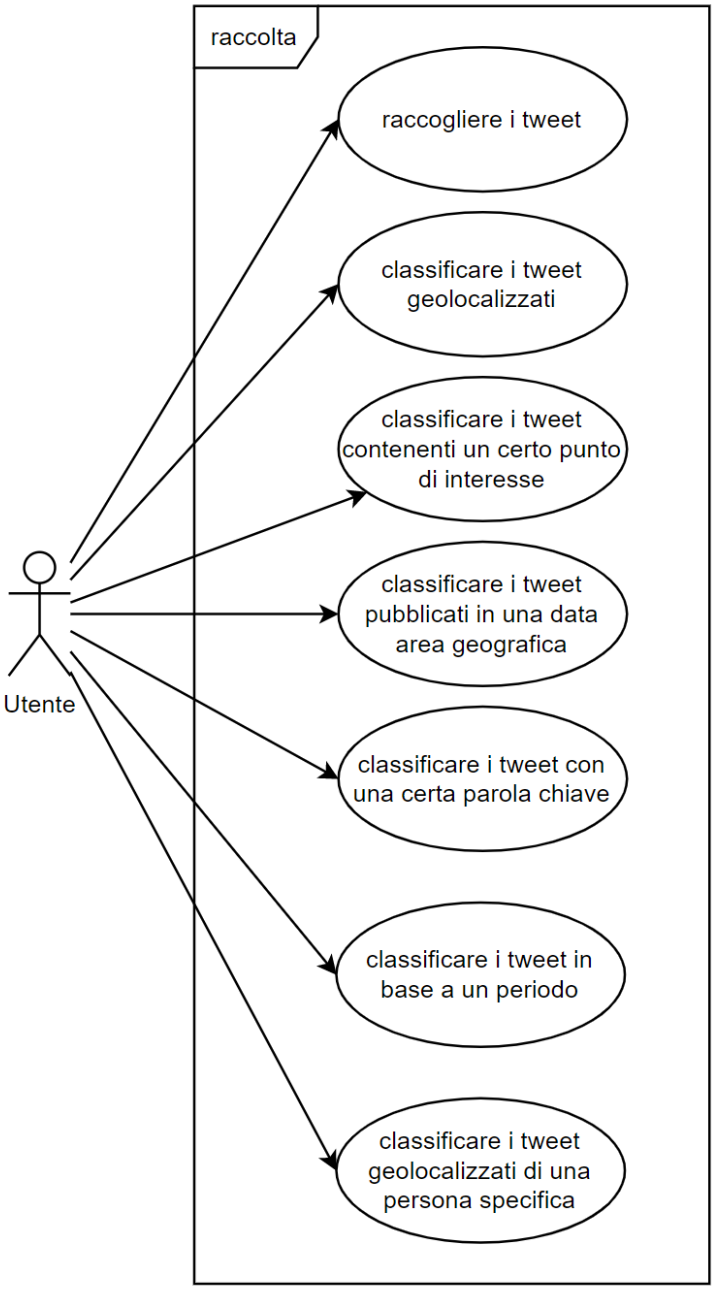


- La gestione delle Allerte sia dal punto di vista dell'Utente che del Sistema:

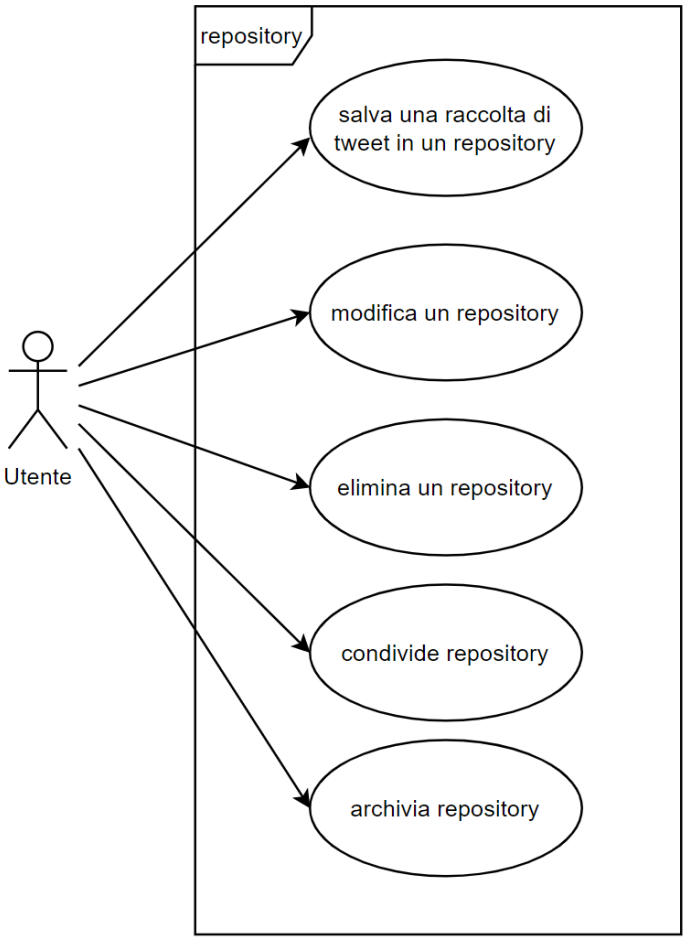




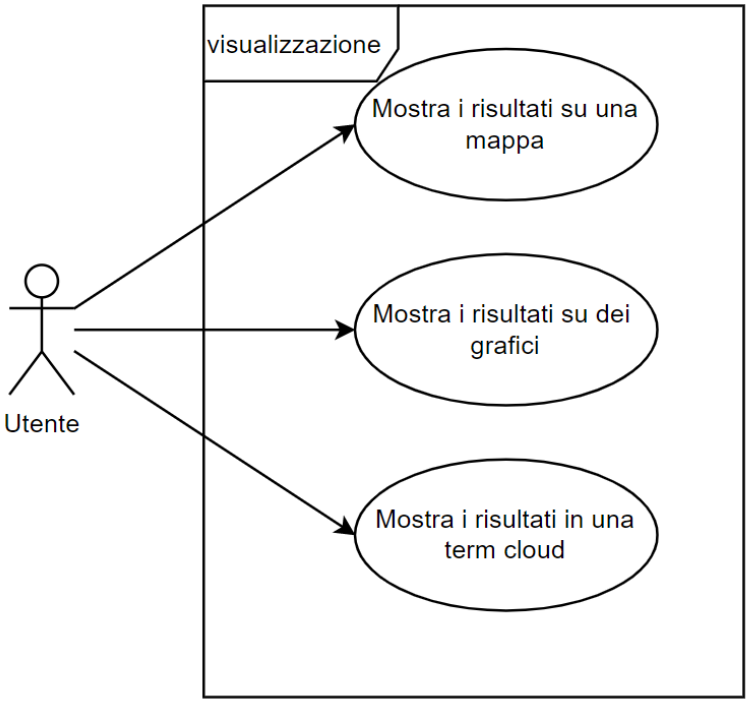
- La gestione della raccolta da parte dell'utente:



- La gestione di un repository da parte dell'utente:



- La visualizzazione di un repository:



4.7 Backlog generale

Si riporta qui di seguito il Backlog definito ad inizio progetto, prima dell'avvio dello sviluppo. Gli elementi dal bordo grigio sono le epiche:

Nota: Alcune user story sono state rimosse in seguito al feedback ricevuto durante il primo sprint!

#1 Filtrare per la raccolta

#2 Come utente voglio definire una parola chiave per creare un filtro

#3 Come utente voglio definire una zona geografica per creare un filtro

#4 Come utente voglio definire un punto di interesse per creare un filtro

#5 Come utente voglio definire un periodo temporale per creare un filtro

#6 Come utente voglio definire una immagine campione per creare un filtro

#7 Come utente voglio usare l'attributo di geolocalizzazione per creare un filtro

#8 Come utente voglio combinare più filtri per creare un filtro complesso

#9 Come utente voglio elencare i filtri creati

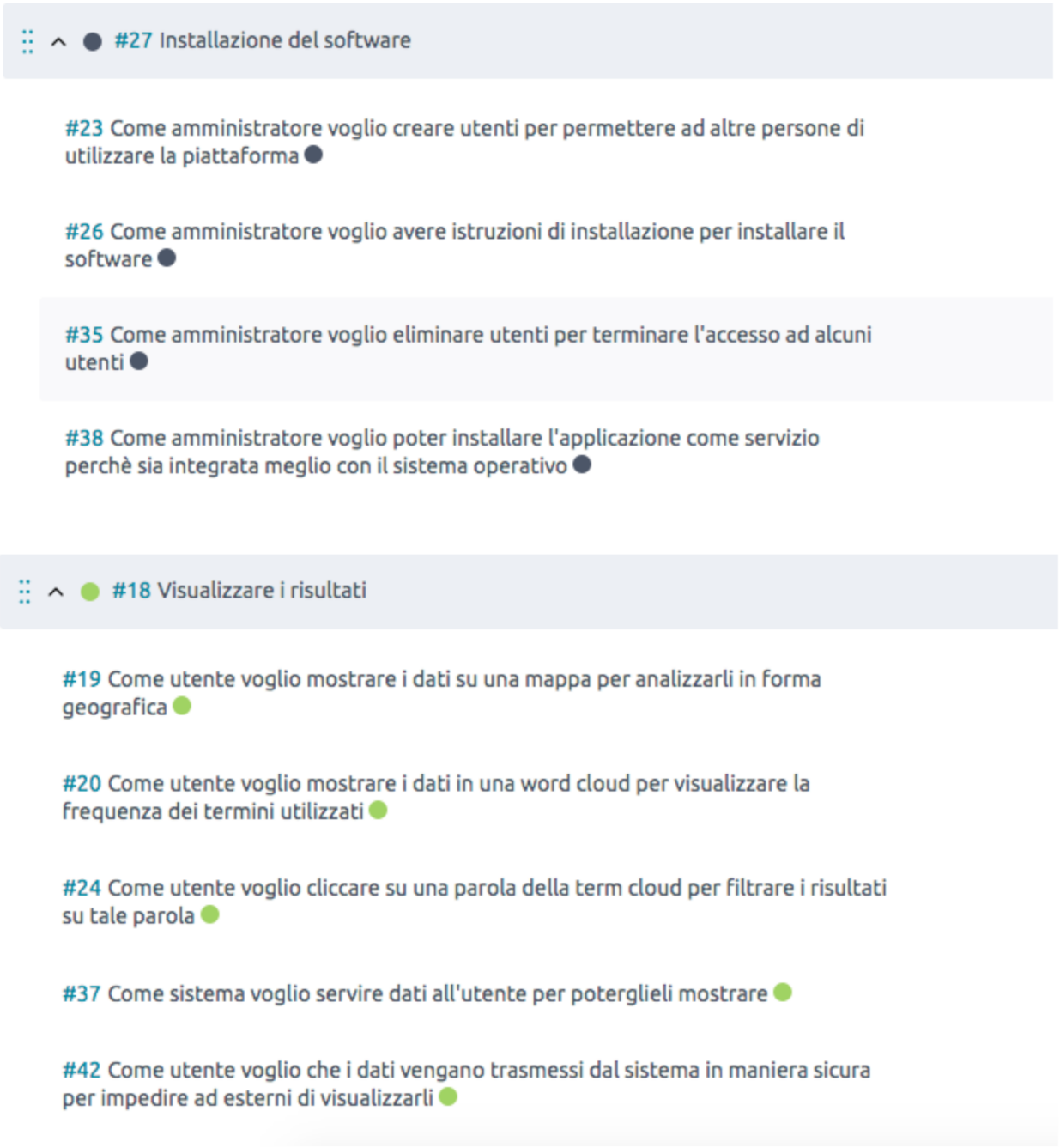
#10 Come utente voglio eliminare un filtro

#11 Come utente voglio attivare un filtro per iniziare la raccolta in tempo reale

#22 Come utente voglio attivare un filtro per applicarlo ad un repository salvato

- #12 Usare un repository
  - #13 Come utente voglio salvare una raccolta per creare un repository
  - #14 Come utente voglio elencare i repository salvati
  - #15 Come utente voglio eliminare un repository salvato
  - #16 Come utente voglio caricare un repository salvato per analizzarlo
  - #36 Come sistema voglio salvare i dati in un database per poterli mantenere in memoria
  - #51 Come utente voglio poter condividere i miei dati con altri utenti per collaborare

- #25 Allertare l'utente
  - #28 Come utente voglio impostare un criterio per essere allertato
  - #29 Come utente voglio eliminare un criterio di allerta per non essere più allertato
  - #30 Come utente voglio visualizzare un elenco dei criteri di allerta impostati
  - #31 Come sistema voglio analizzare tutti i criteri di allerta impostati dall'utente
  - #32 Come sistema voglio allertare l'utente con un avviso in-app se si attiva un criterio da lui definito
  - #33 Come sistema voglio allertare l'utente con un tweet se si attiva un criterio da lui definito
  - #34 Come sistema voglio allertare l'utente con una email se si attiva un criterio da lui definito


#27 Installazione del software
#23 Come amministratore voglio creare utenti per permettere ad altre persone di utilizzare la piattaforma
#26 Come amministratore voglio avere istruzioni di installazione per installare il software
#35 Come amministratore voglio eliminare utenti per terminare l'accesso ad alcuni utenti
#38 Come amministratore voglio poter installare l'applicazione come servizio perchè sia integrata meglio con il sistema operativo
#18 Visualizzare i risultati
#19 Come utente voglio mostrare i dati su una mappa per analizzarli in forma geografica
#20 Come utente voglio mostrare i dati in una word cloud per visualizzare la frequenza dei termini utilizzati
#24 Come utente voglio cliccare su una parola della term cloud per filtrare i risultati su tale parola
#37 Come sistema voglio servire dati all'utente per poterglieli mostrare
#42 Come utente voglio che i dati vengano trasmessi dal sistema in maniera sicura per impedire ad esterni di visualizzarli

CAPITOLO 5

Strumenti utilizzati

Nello sviluppo di N.E.S.T. sono stati usati i seguenti strumenti software:

IntelliJ IDEA Ultimate

: IDE multilinguaggio utilizzato per lo sviluppo di tutte le parti di N.E.S.T., per la scrittura della documentazione, per l'esecuzione del testing e per la visualizzazione del coverage, per il refactoring automatico e per l'analisi statica del codice.

Git

: Software di controllo versione utilizzato per tracciare tutte le modifiche al software.

GitLab

: Applicazione web self-hosted per hosting di repository Git, utilizzato per ospitare il repository del progetto.

  Suggerimento: È accessibile al seguente indirizzo: https://gitlab.steffo.eu/

Taiga

: Applicazione web self-hosted per il Project Management, utilizzata per tracciare lo stato del progetto giorno per giorno.

  Suggerimento: È accessibile al seguente indirizzo: https://taiga.steffo.eu/

SonarQube

: Applicazione web self-hosted per l'analisi statica e visualizzazione del coverage del software, utilizzata per il miglioramento della qualità del codice del progetto.

  Suggerimento: È accessibile al seguente indirizzo: https://sonarqube.steffo.eu/

Figma

Applicazione web centralizzata per la creazione collaborativa di interfacce grafiche moderne.

Discord

Piattaforma centralizzata di messaggistica istantanea e chiamate vocali utilizzata per la comunicazione sincrona tra i membri del team.

CAPITOLO 6

Processo di sviluppo

6.1 Ruoli

Per lavorare più efficientemente, si è assegnato uno o più "ruoli" a ogni membro del team in base alle proprie competenze.

Stefano Goldoni Product Owner, Tester

Flavia Cocca Scrum Master, UI Designer

Chiara Calzolari UI Designer, Translator

Stefano Pigozzi Frontend Developer, Sysadmin

Giovanni Anniballi Lead Tester

Giorgio Minoccari Analyst, Crawler Developer

Lorenzo Balugani Database Architect, Backend Developer

Questi ruoli sono stati usati in maniera **flessibile**; è capitato infatti moltissime volte che un membro con un ruolo ne aiutasse un altro con un ruolo diverso: ad esempio, è successo che i Developer aiutassero i Tester indicando loro particolarità specifiche dei linguaggi di programmazione utilizzati, oppure che gli UI Designer aiutassero nella scrittura della documentazione.

6.2 Sprint

Lo sviluppo si è svolto in **4 Sprint** dalla durata di **2 settimane ciascuno**.

6.2.1 Sprint 0: 04 Apr - 18 Apr

6.2.1.1 Consegna

La seguente documentazione è stata fornita dal cliente durante questo sprint:

- `0-initial.pdf`
- `0-sprint-requirements.pdf`

6.2.1.2 Definition of Ready

Il team ha definito lo stato di Ready di una User Story in base ai seguenti criteri:

- La User Story è stata compresa ed accettata da tutti i membri
- I tester hanno confermato la possibilità di poterla testare
- Il Product Owner ha la visione necessaria per definirne la priorità
- Il Team è in grado di stimarla
- La User Story è indipendente o dipendente da altre a priorità maggiore

6.2.1.3 Definition of Done

La definizione di Done è stata concordata da tutto il team con il Product Owner, ed è stata così definita:

- Sviluppo completo della funzionalità richiesta
- Definizione e superamento dei test
- Bozza della documentazione della funzionalità
- Merge dei sorgenti nel branch `main` del repository Git

6.2.1.4 Statistiche

Gitinspector

Questa statistica è stata generata dal prof. Marcello Missiroli con Gitinspector al termine dello Sprint.

- `0-stats.html`

6.2.1.5 Sprint Retrospective

La Sprint Retrospective è disponibile al seguente link:

- `0-retrospective.pdf`

6.2.1.6 Sprint Review

Il video di Sprint Review è disponibile al seguente link:

- https://drive.google.com/file/d/12worWEcx-uf2UP4_lnEOovHZpvR77MsG/view

6.2.1.7 Artefatti

In questo sprint è stato realizzato un documento con i risultati dell'analisi effettuata per realizzare il software:

- `0-result.pdf`

6.2.1.8 Registro attività

Riunioni collettive

Data	Ora	Durata	Attività
		10h 6m	Totale
2021-04-08	20:45	1h	Analisi documento di specifiche
2021-04-09	14:15	35m	Intervista con il cliente
2021-04-09	15:00	1h	Resoconto intervista
2021-04-11	15:03	2h 39m	Documentazione e planning poker
2021-04-15	18:15	52m	Documentazione
2021-04-17	15:05	4h	Documentazione e partita a Scrumble

Attività individuali

Suggerimento: Per vedere più in dettaglio il lavoro di sviluppo effettuato da ogni membro del gruppo, si suggerisce di visualizzare il log di Git:

```
$ git log
```

Stefano Goldoni

Data	Durata	Attività
	6h 30m	Totale
2021-04-08	1h	user stories
2021-04-09	1h	epic
2021-04-10	30m	use cases
2021-04-11	30m	use cases
2021-04-14	30m	documentazione SRS
2021-04-15	30m	documentazione SRS
2021-04-17	1h 30m	diagrammi di attività

Flavia Cocca

Data	Durata	Attività
	15h	Totale
2021-04-08	2h 30m	brainstorming Ui
2021-04-10	2h	creazione frame in figma con relativo css
2021-04-11	1h 30m	mockup UI con relativo css (creazione componenti)
2021-04-12	1h 30m	mockup UI con relativo css (creazione componenti)
2021-04-14	3h	mockup UI con relativo css (creazione pagine)
2021-04-15	2h	mockup UI con relativo css (creazioni pagine)
2021-04-16	2h	ultimi ritocchi UI
2021-04-17	30m	Stesura documentazione partita scrumble

Chiara Calzolari

Data	Durata	Attività
	12h 30m	Totale
2021-04-09	2.5h	Brainstorming UI
2021-04-10	1.5h	mockup UI con relativo css (dashboard ed elenco repositories)
2021-04-11	1.0h	mockup UI con relativo css (settings)
2021-04-11	0.5h	Prima versione del logo
2021-04-12	3.0h	mockup UI con relativo css (creazione componenti, light/dark mode)
2021-04-13	1.0h	mockup UI con relativo css (alerts)
2021-04-15	1.0h	Versione definitiva del logo
2021-04-16	0.5h	mockup UI con relativo css (Notifications e Share)
2021-04-17	1.0h	Documentazione Mockup
2021-04-17	0.5h	Doppiaggio video di presentazione

Stefano Pigozzi

Data	Durata	Attività
	23h 30m	Totale
2021-04-08	4h 30m	Configurazione Discord e GitLab
2021-04-09	3h 30m	Riunione con il cliente e configurazione Taiga
2021-04-10	4h	Configurazione Penpot, Taiga, UI Design, progettazione Database
2021-04-11	3h 30m	Riunione, configurazione GitLab e project management
2021-04-12	30m	Configurazione Twitter e UI Design
2021-04-15	2h 30m	Partita a Scrumble e project management
2021-04-16	1h 30m	Project management e configurazione SonarQube
2021-04-17	3h 30m	Riunione e partita a scrumble

Giovanni Anniballi

Nota: Giovanni ha iniziato a raccogliere dati sul tempo impiegato a partire dallo Sprint 1.

Giorgio Minoccari

Nota: Giorgio ha iniziato a raccogliere dati sul tempo impiegato a partire dallo Sprint 1.

Lorenzo Balugani

Nota: Lorenzo ha iniziato a raccogliere dati sul tempo impiegato a partire dallo Sprint 1.

6.2.1.9 Risultati della partita di Scrumble

1. **Goal** Learn

   Question Do team members understand the Scrum roles?

   Metric Knowledge of Scrum roles by questions

   Evaluation

   1 no idea of the Scrum roles

   5 perfect knowledge of the roles and their jobs

   Chiara 4

   Giorgio 4

   Giovanni 4

   Stefano P. 4

   Lorenzo 5

   Stefano G. 5

   Flavia 4

2. **Goal** Learn

   Question Do team members feel they learned the process?

   Metric Opinions from the participants

   Evaluation

   1 couldn't repeat the game

   5 could play the game as a Scrum Master by himself

   Chiara 3

   Giorgio 3

Giovanni 4

Stefano P. 5

Lorenzo 3

Stefano G. 3

Flavia 3

3. **Goal** Learn

   Question Does everyone keep up with the other players?

   Metric Check during every sprint retrospective if every one is on point

   Evaluation

   1 totally lost

   5 leads the game driving the other players

   Chiara 3

   Giorgio 4

   Giovanni 4

   Stefano P. 4

   Lorenzo 5

   Stefano G. 5

   Flavia 5

4. **Goal** Practice

   Question Are the game mechanics linear and repeatable?

   Metric Opinions from the participants

   Evaluation

   1 feels the game is unrepeatable

   5 feels the game could be played in any situation

   Chiara 1

   Giorgio 2

   Giovanni 1

   Stefano P. 1

   Lorenzo 1

   Stefano G. 2

   Flavia 1

5. **Goal** Practice

   Question Do team success in completing the game?

   Metric Number of User Stories completed

   Evaluation

   1 0 to 3 stories

2 4 to 6

3 7 to 9

4 10 to 12

5 13 to 15

Chiara 5

Giorgio 5

Giovanni 5

Stefano P. 5

Lorenzo 5

Stefano G. 5

Flavia 5

6. **Goal** Practice

   Question Do team members efficiently estimate during sprint planning?

   Metric Uniformity in evaluating the size and the priority of user stories

   Evaluation

   1 abnormal difference from the other players

   5 coherent and uniform with the group most of the time

   Chiara 5

   Giorgio 4

   Giovanni 5

   Stefano P. 4

   Lorenzo 5

7. **Goal** Cooperation

   Question Do team members know each other better?

   Metric Level of players' serenity throughout the game

   Evaluation

   1 never speaks with the other players

   5 talks friendly to anyone in every situation

   Chiara 4

   Giorgio 5

   Giovanni 5

   Stefano P. 5

   Lorenzo 5

   Stefano G. 5

   Flavia 4

8. **Goal** Cooperation

Question Does the game let all players cooperate?

Metric Contribution of every player during the game

Evaluation

1 never puts effort in doing something

5 every time is willing to understand what is going on

Chiara 4

Giorgio 3

Giovanni 3

Stefano P. 2

Lorenzo 3

Stefano G. 4

Flavia 3

9. **Goal** Cooperation

Question Do team member consult each other about a topic?

Metric Sharing of ideas

Evaluation

1 never asks for an opinion

5 wants to discuss about every topic

Chiara 5

Giorgio 5

Giovanni 5

Stefano P. 3

Lorenzo 5

Stefano G. 4

Flavia 5

10. **Goal** Motivation

Question Do team members encourage collegues in need?

Metric Players explain something other players don't understand

Evaluation

1 not involved by the game

5 always makes sure everyone is on point

Chiara 3

Giorgio 5

Giovanni 5

Stefano P. 4

Lorenzo 5

Stefano G. 4

Flavia 4

11. **Goal** Motivation

    Question Does PO help the team?

    Metric Quality of PO's advices to get better in the next sprints

    Evaluation

    1 poor/absent advices

    5 wise and helpful suggestions when is required

    Stefano G. 4

12. **Goal** Motivation

    Question Does the team come up with good ideas?

    Metric Effectiveness of sprint retrospective

    Evaluation

    1 doesn't express opinions during retrospective

    5 feels the retrospective fundamental to express opinions

    Chiara 4

    Giorgio 5

    Giovanni 5

    Stefano P. 5

    Lorenzo 5

    Stefano G. 5

    Flavia 5

13. **Goal** Problem Solving

    Question Do team members behave well when facing a problem?

    Metric Level of the technical debt at the end of the game

    Evaluation On the game board, if the debt pawn is on the lowest stage,the evaluation is 5, for every higher stage it decreases by 1

    Chiara 5

    Giorgio 5

    Giovanni 5

    Stefano P. 5

    Lorenzo 5

    Stefano G. 5

    Flavia 5

14. **Goal** Problem Solving

    Question Does team organize their tasks properly?

Metric Average of tasks left at the end of each sprint

Evaluation

1 21+ average tasks left

2 16-20 average tasks left

3 11-15 average tasks left

4 6-10 average tasks left

5 0-5 average tasks left

Chiara 5

Giorgio 5

Giovanni 5

Stefano P. 5

Lorenzo 5

15. **Goal** Problem Solving

Question Does PO plan efficiently the Sprint Backlog?

Metric Average of tasks left at the end of each sprint

Evaluation

1 21+ average tasks left

2 16-20 average tasks left

3 11-15 average tasks left

4 6-10 average tasks left

5 0-5 average tasks left

Stefano G. 5

6.2.2 Sprint 1: 19 Apr - 02 Mag

6.2.2.1 Consegna

La seguente documentazione è stata fornita dal cliente durante questo Sprint:

- `1-sprint-requirements.pdf`

6.2.2.2 Goal

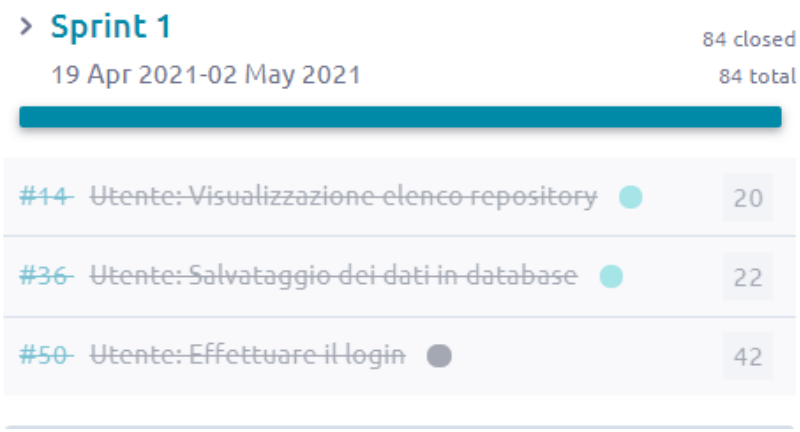
Il **goal** per questo Sprint è stato costruire una codebase facilmente mantenibile e modulare, che potesse accomodare grossi cambiamenti nei sottomoduli senza impattare gli altri:

- Dal lato **backend**: l'obiettivo è stato predisporre la struttura del database e creare le API calls per il login degli utenti e la creazione dei repository
- Dal lato **testing**: creare un piccolo framework per il testing automatizzato del backend
- Dal lato **frontend**: creare la struttura con le componenti importate da Figma e implementare le schermate principali

Backlog


Sprint 1
19 Apr 2021-02 May 2021
0 closed
405 total
#2 Come utente voglio definire una parola chiave per creare un filtro 29
#3 Come utente voglio definire una zona geografica per creare un filtro 63
#4 Come utente voglio definire un punto di interesse per creare un filtro 52
#5 Come utente voglio definire un periodo temporale per creare un filtro 41
#7 Come utente voglio usare l'attributo di geolocalizzazione per creare un filtro 30
#11 Come utente voglio attivare un filtro per iniziare la raccolta in tempo reale 22
#13 Come utente voglio salvare una raccolta per creare un repository 24
#14 Come utente voglio elencare i repository salvati 20
#22 Come utente voglio attivare un filtro per applicarlo ad un repository salvato 27
#36 Come sistema voglio salvare i dati in un database per poterli mantenere in memoria 22
#42 Come utente voglio che i dati vengano trasmessi dal sistema in maniera sicura per impedire ad esterni di visualizzarli 33
#50 Come utente voglio effettuare il login per vedere i miei dati 42

Task completati



6.2.2.3 Definition of Ready

Il team ha definito lo stato di Ready di una User Story in base ai seguenti criteri:

- La User Story è stata compresa ed accettata da tutti i membri
- I tester hanno confermato la possibilità di poterla testare
- Il Product Owner ha la visione necessaria per definirne la priorità
- Il Team è in grado di stimarla
- La User Story è indipendente o dipendente da altre a priorità maggiore

6.2.2.4 Definition of Done

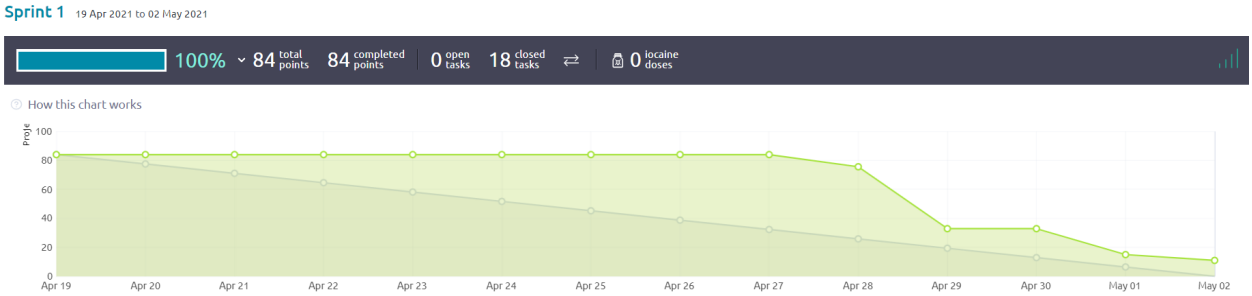
La definizione di Done è stata concordata da tutto il team con il Product Owner, ed è stata così definita:

- Sviluppo completo della funzionalità richiesta
- Definizione e superamento dei test
- Bozza della documentazione della funzionalità
- Merge dei sorgenti nel branch `main` di GitLab

6.2.2.5 Statistiche

Burndown chart

Questa è la burndown chart relativa allo Sprint 1:



Gitinspector

Questa statistica è stata generata dal prof. Marcello Missiroli con Gitinspector al termine dello Sprint.

- `1-stats.html`

6.2.2.6 Sprint Retrospective

La Sprint Retrospective è disponibile al seguente link:

- `1-retrospective.pdf`

6.2.2.7 Sprint Review

Il video di Sprint Review è disponibile al seguente link:

- https://drive.google.com/drive/folders/1dsis_cGCRnVgZAkZjEVIZKt4NndkycaF

6.2.2.8 Valutazione sul debito tecnico

Durante questo sprint è stata prodotta dallo Scrum Master la seguente valutazione sul debito tecnico:

> La valutazione del debito tecnico effettuata da SonarQube è relativa principalmente al backend, poiché l'analisi del progetto sarebbe stata troppo dispendiosa a causa dell'elevato numero di file presente nella cartella `/frontend`.
>
> Per quanto riguarda il backend invece il debito tecnico riscontrato è minore del 5% come si può verificare dal grado A attestato da SonarQube.
>
> Un punto chiave nell'implementazione ha permesso al team di diminuire il debito tecnico, questo è avvenuto grazie all'adozione del modello architetturale REST che ha migliorato esponenzialmente il tempo di risposta delle richieste e la leggibilità del codice.
>
> —Flavia Cocca, Scrum Master

6.2.2.9 Valutazione sulle User Stories

Durante questo sprint è stata prodotta dal Product Owner la seguente valutazione sulle User Stories:

> Durante questo sprint non sono state ristimate le US in quanto non sono emerse al termine dello Sprint 0 valutazioni che lo rendessero necessario, di conseguenza non sono state neanche rivalutate le priorità.
>
> Come Product Owner ho individuato come criterio di accettazione l'esito positivo dei test, anche per il fatto che i test sono presenti in ogni US.
>
> Durante lo sviluppo di questo primo sprint sono ovviamente emerse delle issue che sono state tracciate su Taiga. Una di queste, la #101 "L'API non è interamente REST" ha portato ad un refactoring di una parte del codice del backend, del frontend e naturalmente anche delle procedure di test, che erano già state abbozzate. La modifica comunque si è rilevata essere necessaria per rispondere appieno alle struttura tipica dei metodi REST.
>
> Altre issue sono state inserite ma possono essere sistemate nello sprint successivo in quanto non influiscono sul funzionamento atteso in questo primo sprint.
>
> Un leggero ritardo sullo sviluppo del frontend non consente di chiudere alcune US, che verranno chiuse e testate nel prossimo Sprint.
>
> —Stefano Goldoni, Product Owner

6.2.2.10 Registro attività

Riunioni collettive

Data	Ora	Durata	Attività
		2h	Totale
2021-05-01	15:30	2h	backlog grooming session

Attività individuali

Suggerimento: Per vedere più in dettaglio il lavoro di sviluppo effettuato da ogni membro del gruppo, si suggerisce di visualizzare il log di Git:

```
$ git log
```

Stefano Goldoni

Data	Durata	Attività
	23h 30m	Totale
2021-04-20	30m	analisi test
2021-04-20	1h 30m	scrittura test cases
2021-04-26	30m	predisposizione progetto in locale per il test
2021-04-26	2h	metodi di test login e creazione utente
2021-04-28	2h	metodi ti test
2021-04-29	2h	metodi di test
2021-04-30	2h	metodi di test backend
2021-05-01	1h	test frontend
2021-05-01	1h	test backend in pair programming
2021-05-02	1h	documentazione
2021-05-02	2h 30m	test backend

Flavia Cocca

Data	Durata	Attività
	8h	Totale
2021-04-27	1h	modifica mockup UI
2021-04-28	1h	Organizzazione Sprint retrospective
2021-04-30	1h	modifica mockup UI
2021-05-01	2h	Riunione con il team per Sprint review e Sprint retrospective
2021-05-02	3h	Stesura documenti Sprint 1 (debito tecnico e relazione Sprint retrospective), realizzazione video Sprint1 review

Chiara Calzolari

Data	Durata	Attività
	6h 30m	Totale
2021-04-26	3h	Configurazione ambiente di sviluppo
2021-04-27	1h	modifica mockup UI (adeguamento al JS)
2021-04-29	2h	modifica mockup UI (Manage users)
2021-05-01	30m	Ultimi ritocchi al mockup UI

Stefano Pigozzi

Data	Durata	Attività
	29h	Totale
2021-04-19	18m	Configurazione IntelliJ IDEA
2021-04-19	10m	Configurazione IntelliJ IDEA
2021-04-20	1h 26m	Sviluppo
2021-04-21	1h 3m	Sviluppo
2021-04-21	3m	Configurazione IntelliJ IDEA
2021-04-21	58m	Sviluppo
2021-04-21	13m	Sviluppo
2021-04-21	7m	Sviluppo
2021-04-21	6m	Project management
2021-04-21	15m	Sviluppo
2021-04-21	32m	Sviluppo
2021-04-21	10m	Sviluppo
2021-04-22	5m	Sviluppo
2021-04-22	16m	Presentazione
2021-04-22	36m	Riunione
2021-04-22	1h 23m	Sviluppo
2021-04-23	18m	Studio
2021-04-23	42m	Sviluppo
2021-04-23	10m	Sviluppo
2021-04-23	1h 9m	Sviluppo
2021-04-23	3m	Sviluppo
2021-04-24	19m	Sviluppo
2021-04-24	19m	Sviluppo
2021-04-25	2h 59m	Sviluppo
2021-04-25	10m	Documentazione
2021-04-26	13m	Configurazione SonarQube
2021-04-26	12m	Documentazione
2021-04-26	30m	Documentazione
2021-04-26	4m	Riunione
2021-04-26	2h 58m	Sviluppo
2021-04-26	1h 4m	Sviluppo
2021-04-26	56m	Sviluppo
2021-04-26	6m	Project management
2021-04-27	7m	Sviluppo
2021-04-27	15m	Configurazione Taiga
2021-04-27	29m	Sviluppo
2021-04-27	35m	Collaborazione
2021-04-28	34m	Collaborazione
2021-04-29	1h 42m	Sviluppo
2021-04-29	1h 26m	Sviluppo
2021-04-29	48m	Project management
2021-04-30	56m	Sviluppo
2021-04-30	46m	Sviluppo
2021-05-01	12m	Sviluppo
2021-05-02	14m	Documentazione
2021-05-02	28m	Bugfixing

continues on next page

Tabella 1 – continua dalla pagina precedente

Data	Durata	Attività
	29h	Totale
2021-05-02	15m	Collaborazione
2021-05-02	3m	Bugfixing

Giovanni Anniballi

Per il lavoro di testing è stata utilizzata anche la tecnica del Pair Programming, grazie ad un plugin dell'IDE utilizzato.

Data	Durata	Attività
	16h 30m	Totale
2021-04-22	30m	Riunione con il team
2021-04-23	1h	Studio struttura del backend
2021-04-24	1h 30m	Studio del backend, volto a capire il funzionamento delle varie API
2021-04-26	2h	Testing login e retrieval dei dati inerenti a tutti gli utenti registrati
2021-04-27	1h	Testing creazione user
2021-04-28	2h	Test cancellazione utente e ritorno dei dati inerenti a quell'utente
2021-04-29	1h 30m	Test modica utenti e creazione della prima repository
2021-04-30	2h	Testing ritorno di tutte le repository di proprietà dell'utente loggato, ritorno delle info inerenti alla repository specificata
2021-05-01	2h	Riunione con il team per Sprint review e Sprint retrospective
2021-05-01	1h	Testing modifica ed eliminazione di una repository (nome, stato)
2021-05-02	2h	Testing del frontend tramite UI, controllo generale dei test già effettuati.

Giorgio Minoccari

Durante lo Sprint ho principalmente svolto sviluppo e test riguardo alla API di twitter, in modo da poter effettuare chiamate efficenti e non venire limitati dal sito riguardo alle richieste effettuate.

Data	Durata	Attività
	10h	Totale
2021-04-21	1h	Primi test riguardo autenticazione Oauth per API twitter
2021-04-23	1h 30m	Autenticazione funzionante
2021-04-24	30m	Test su ricerche generali con parole chiave
2021-04-26	1h	Test su ricerche geolocalizzate
2021-04-27	1h	Definizione di entita' di esempio nei database per provare funzioni legate ad esse
2021-04-28	2h	Definizione di funzioni per l'aggregazione di condizioni di diverso tipo per la ricerca di tweet tramite API
2021-04-29	1h	Refactor codice scritto fino a quel momento, eliminazione di dati inutili
2021-05-01	1h	Implementazioni di analisi su termini e hashtag restituiti dall'API
2021-05-02	1h	Test di chiamata alle funzioni di ricerca a partire da repository con condizioni complesse aggregate

Lorenzo Balugani

Data	Durata	Attività
	13h 30m	Totale
2021-04-21	2h	Implementazione Base di Dati
2021-04-22	2h	Login, creazione utenti
2021-04-25	1h	Rimozione utenti, CORS, Creazione repo
2021-04-25	3h	Altre funzioni API
2021-04-26	30m	Standardizzazione output json
2021-04-26	1h	Documentazione
2021-04-27	30m	Aggiunto supporto al modulo explorer, fix
2021-04-28	30m	Bugfixing
2021-04-29	1h	Gestione migliorata errori
2021-04-29	1h	Refactoring
2021-05-02	1h	Bugfixing

6.2.3 Sprint 2: 03 Mag - 16 Mag

6.2.3.1 Consegna

La seguente documentazione è stata fornita dal cliente durante questo sprint:

- `2-sprint-requirements.pdf`

6.2.3.2 Goal

Il **goal** del secondo Sprint è stato la creazione, cancellazione e modifica dei repository, ovvero permettere agli utenti di creare repository partendo da una ricerca che rispetta determinate condizioni.

Backlog


Sprint 2
03 May 2021-16 May 2021
402 closed
643 total
#2 Utente: Creazione di Conditions basate su Hashtag 29
#3 Utente: Creazione di Conditions basate su zona geografica 63
#5 Utente: Creazione di Conditions basate su periodi di tempo 41
#7 Utente: Creazione di Condition che raccolga tutti i tweet aventi il tag di geolocalizzazione 30
#11 Utente: Creazione di repository 22
#13 Utente: Archiviazione di un repository 24
#22 Utente: Aggiungere filtri a repository già esistente 27
#23 Amministratore: Creazione ed eliminazione utenti 16
#8 Utente: usa BoolOperations tra Conditions per le allerte 70
#9 Utente: visualizzazione delle Conditions di un repository 16
#10 Utente: Eliminazione di una Condition 30
#15 Utente: Eliminazione di un repository 19
#16 Utente: Effettua analisi su repository 21
#42 Server: Utilizza HTTPS 33
#26 Amministratore: Installa il software 32
#20 Utente: Visualizzazione con wordcloud 51
#28 Utente: Imposta un'Alert 53
#29 Utente: Elimina un'Alert 23
#30 Utente: Visualizza Alert 23
#38 Amministratore: Configura un servizio di Systemd per rendere persistente il software 20
#219 Utente: Traduzione in italiano
SPRINT TASKBOARD

Task completati

6.2.3.3 Definition of Ready

Il team ha definito lo stato di Ready di una User Story in base ai seguenti criteri:

- La User Story è stata compresa ed accettata da tutti i membri
- I tester hanno confermato la possibilità di poterla testare
- Il Product Owner ha la visione necessaria per definirne la priorità
- Il Team è in grado di stimarla
- La User Story è indipendente o dipendente da altre a priorità maggiore

6.2.3.4 Definition of Done

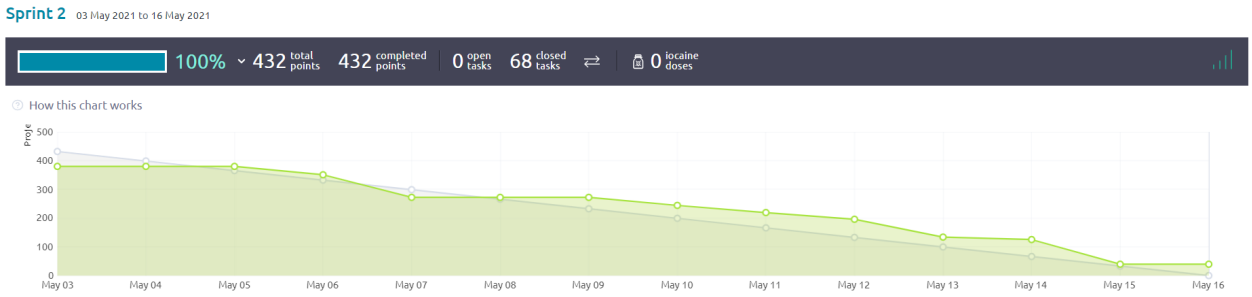
La definizione di Done è stata concordata da tutto il team con il Product Owner, ed è stata così definita:

- Sviluppo completo della funzionalità richiesta
- Definizione e superamento dei test
- Bozza della documentazione della funzionalità
- Merge dei sorgenti nel branch `main` di GitLab

6.2.3.5 Statistiche

Burndown chart

Questa è la burndown chart relativa allo Sprint 2:



Gitinspector

Questa statistica è stata generata dal prof. Marcello Missiroli con Gitinspector al termine dello Sprint.

- `2-stats.html` (relativa al codice)
- `2D-stats.html` (relativa a parte della documentazione)

6.2.3.6 Sprint Retrospective

La Sprint Retrospective è disponibile al seguente link:

- `2-retrospective.pdf`

6.2.3.7 Sprint Review

Il video di Sprint Review è disponibile al seguente link:

- https://drive.google.com/file/d/1x1kub-bpVJrwmGrn5LLU8ecqcbxFaoKg/view

6.2.3.8 Registro attività

Riunioni collettive

Data	Ora	Durata	Attività
		1h	Totale
2021-05-14	10:00	1h	Sprint Retrospective session

Attività individuali

Suggerimento: Per vedere più in dettaglio il lavoro di sviluppo effettuato da ogni membro del gruppo, si suggerisce di visualizzare il log di Git:

```
$ git log
```

Stefano Goldoni

Data	Durata	Attività
	16h	Totale
2021-05-04	1h	preparazione backlog sprint 2
2021-05-10	3h	test sprint 2
2021-05-11	2h 30m	test utenti e repository
2021-05-12	2h	test repository
2021-05-13	2h 30m	test repository
2021-05-14	3h	test conditions
2021-05-15	2h	test conditions + sonarqube scanner

Flavia Cocca

Data	Durata	Attività
	16h	Totale
2021-05-04	30m	Riunione per con PO, nuovo backlog
2021-05-05	1h	Riunione
2021-05-07	3h	Refactoring test
2021-05-08	2h	Refactoring test
2021-05-09	3h	Nuovi test
2021-05-11	2h 30m	Discussione test e risoluzione di alcune issue
2021-05-13	1h	Bugfixing
2021-05-14	1h	Sprint Retrospective
2021-05-15	2h	Documentazione

Stefano Pigozzi

Data	Durata	Attività
	27h 30m	Totale
2021-05-05	40m	Riunione
2021-05-05	21m	Documentazione
2021-05-06	9m	Sviluppo
2021-05-06	3m	Configurazione Taiga
2021-05-06	11m	Documentazione
2021-05-06	9m	Riunione
2021-05-06	20m	Documentazione
2021-05-06	39m	Sviluppo
2021-05-06	15m	Riunione
2021-05-06	1h 4m	Riunione
2021-05-07	1h 47m	Sviluppo
2021-05-07	17m	Sviluppo
2021-05-07	34m	Sviluppo
2021-05-07	34m	Project management
2021-05-07	2h 44m	Sviluppo
2021-05-08	13m	Sviluppo
2021-05-08	29m	Sviluppo
2021-05-08	18m	Sviluppo
2021-05-10	1h 00m	Sviluppo
2021-05-10	1h 15m	Configurazione GitLab
2021-05-10	25m	Sviluppo
2021-05-10	39m	Sviluppo
2021-05-11	25m	Bugfixing
2021-05-11	3h 57m	Sviluppo
2021-05-11	1h 45m	Sviluppo
2021-05-12	10m	Sviluppo
2021-05-12	1h 51m	Sviluppo
2021-05-13	1h 00m	Sviluppo
2021-05-13	1h 20m	Sviluppo
2021-05-14	20m	Sviluppo

continues on next page

Tabella 2 – continua dalla pagina precedente

Data	Durata	Attività
	27h 30m	Totale
2021-05-14	3h 17m	Documentazione
2021-05-15	1h 8m	Porting dipendenze

Chiara Calzolari

Data	Durata	Attività
	11h	Totale
2021-05-03	1h	Testing frontend e segnalazione issues
2021-05-03	1h	Modifica mockup UI (Dashboard repository)
2021-05-04	1h	Modifica mockup UI (adeguamento al JS)
2021-05-10	3h	Configurazione ambiente di sviluppo
2021-05-11	1h 30m	Modifica UI (traduzione in Italiano)
2021-05-12	30m	Modifica UI (traduzione in Italiano)
2021-05-14	3h	Realizzazione video demo sprint 2

Giovanni Anniballi

Data	Durata	Attività
	17h	Totale
2021-05-04	30m	Incontro con SM e PO per valutare quali US andranno nello sprint2
2021-05-05	1h	Incontro con il team
2021-05-07	2h	Refactoring tests sugli user e primi utilizzi delle fixtures
2021-05-08	2h	Completamento test users
2021-05-10	1h	Ulteriori approcci (fallimentari) al coverage di SonarQube
2021-05-11	2h	Discussione sui test e correzioni bug
2021-05-12	1h	Test repository
2021-05-13	1h	Generazione coverage pytest e setting sonarqube (riuscito!)
2021-05-14	4h	Sprint review e retrospettiva, conclusione test repository
2021-05-15	2h	generazione nuovo coverage e fix a sonarqube
2021-05-15	30m	fix piccoli bug segnalati da sonarqube

Giorgio Minoccari

Data	Durata	Attività
	10h	Totale
2021-05-04	30m	Aggiunta della condizione sull'utente
2021-05-10	2h	Tentativo di inserimento delle query con place_id
2021-05-11	1h 30m	Test per passare alla versione 2.0 delle API
2021-05-12	1h	Ritorno alla versione 1.1 perche' lascia disponibili le query sulla geolocalizzazione gratuitamente
2021-05-14	5h	Integrazione e inserimento dei tweet catturati nel database

Lorenzo Balugani

Data	Durata	Attività
	19h	Totale
2021-05-05	4h	Setup di swagger, documentazione
2021-05-06	4h	Documentazione, sviluppo
2021-05-07	3h	Alert, documentazione
2021-05-10	2h	Alert Put, bugfixing
2021-05-11	3h	Bugfixing, sviluppo
2021-05-12	3h	Risolti issue pubblicati su Taiga
2021-05-13	1h	Bugfixing
2021-05-14	1h	Bugfixing

6.2.4 Sprint 3: 17 Mag - 30 Mag

6.2.4.1 Consegna

La seguente documentazione è stata fornita dal cliente durante questo sprint:

- `3-sprint-requirements.pdf`
- `3-report.pdf`

6.2.4.2 Goal

Il **goal** del terzo Sprint è stato far funzionare il crawler, gli alert e completare tutte i task rimanenti.

Backlog


Sprint 3
17 May 2021-30 May 2021
365 closed
488 total
#229 Utente: Traduzione del software 20
#16 Utente: Effettua analisi su repository 21
#238 Utente: Visualizzazione con istogramma temporale 15
#19 Utente: Mappa dei tweet 82
#20 Utente: Visualizzazione con wordcloud 51
#24 Utente: Wordcloud cliccabile e filtrabile 77
#51 Utente: Condivisione repository tra più utenti 55
#31 Sistema: Analizza gli Alert e notifica l'utente in caso uno di essi si verifichi 39
#33 Sistema: Alert con un tweet 29
#30 Utente: Visualizza Alert 23
#28 Utente: Imposta un Alert 53
#29 Utente: Elimina un'Alert 23

Task completati

6.2.4.3 Grooming session

Sono state definite le nuove **User Stories** da inserire nel progetto sulla base delle nuove richieste pervenute dal cliente:

- analisi statistica più dettagliata
- postare su Twitter
- traduzione dell'interfaccia in inglese.

La richiesta relativa alle *ricerche basate sulla geolocalizzazione*, come già comunicato al cliente, non è stata completata interamente per motivi tecnici legati a limitazioni sulle features delle **API 1.1 di Twitter** che non permettono di eseguire query su campi di posizione geografica.

Le nuove User Stories sono state valutate tramite Scrum Poker, durante il quale ogni membro ha espresso la sua valutazione.

Tutte le nuove richieste sono state accettate dal Product Owner e sono pronte ad essere inserite nello sprint di sviluppo in partenza.

6.2.4.4 Definition of Ready

Il team ha definito lo stato di Ready di una User Story in base ai seguenti criteri:

- La User Story è stata compresa ed accettata da tutti i membri
- I tester hanno confermato la possibilità di poterla testare
- Il Product Owner ha la visione necessaria per definirne la priorità
- Il Team è in grado di stimarla
- La User Story è indipendente o dipendente da altre a priorità maggiore

6.2.4.5 Definition of Done

La definizione di Done è stata concordata da tutto il team con il Product Owner, ed è stata così definita:

- Sviluppo completo della funzionalità richiesta
- Definizione e superamento dei test
- Bozza della documentazione della funzionalità
- Merge dei sorgenti nel branch `main` di GitLab

6.2.4.6 Statistiche

Gitinspector

Nota: La statistica dello sprint 3 non è ancora stata generata dal prof. Marcello Missiroli.

Schermata finale di SonarQube

La schermata finale di SonarQube è visibile a questo link:

- `3-sonarqube.pdf`

6.2.4.7 Final Retrospective

- `Retrospettiva finale`

6.2.4.8 Demo

Il video di Demo è disponibile al seguente link:

- https://drive.google.com/file/d/15o70Ffe51CNj8LTKHC9dGiqRVnbv9UpZ/view

6.2.4.9 Registro attività

Riunioni collettive

Data	Durata	Attività
	3.5h	Totale
17/05	2.0 h	Grooming session nuove richieste cliente
29/05	1.5 h	sprint retrospective

Attività individuali

Stefano Goldoni

Data	Durata	Attività
	14h	Totale
21/05	2.0h	Analisi strumenti di test frontend
24/05	3.0h	Inizio test alerts
25/05	2.5h	Test alerts
26/05	3.0h	Test
28/05	3.5h	Test, refactory in base a Sonarqube

Flavia Cocca

Data	Durata	Attività
	14h	Totale
20/05	2.0h	Trasferimento documenti in nuova documentazione
21/05	1.0h	Studio Sphinx
23/05	1.0h	Studio sintassi rST
24/05	1.0h	Documentazione
25/05	1.0h	Documentazione
26/05	1.0h	Documentazione
27/05	1.0h	Documentazione
28/05	3.0h	Documentazione
29/05	3.0h	Documentazione

Chiara Calzolari

Data	Durata	Attività
	17h 30m	Totale
17/05	3.0h	Traduzione UI
17/05	1.5h	Traduzione UI
18/05	1.5h	Traduzione UI
20/05	1.0h	Traduzione UI
22/05	1.0h	Traduzione UI
24/05	2.0h	Traduzione UI
24/05	1.0h	Traduzione UI
25/05	1.0h	Traduzione UI
27/05	0.5h	Traduzione UI
28/05	2.0h	Configurazione ambiente di sviluppo
28/05	3.0h	Creazione video-demo

Stefano Pigozzi

Data	Durata	Attività
	50h 41m	Totale
2021-05-17	25m	Riunione
2021-05-17	19m	Riunione
2021-05-17	2h	Sviluppo
2021-05-17	1h 7m	Bugfixing
2021-05-17	7m	User Interface
2021-05-18	55m	Sviluppo
2021-05-18	14m	Configurazione GitLab
2021-05-18	1h 52m	Documentazione
2021-05-18	22m	Sviluppo
2021-05-18	21m	User Interface
2021-05-18	34m	Sviluppo
2021-05-18	40m	Sviluppo
2021-05-18	29m	Sviluppo
2021-05-18	1h 8m	Sviluppo
2021-05-19	36m	Sviluppo
2021-05-19	2h 40m	Sviluppo
2021-05-19	44m	Sviluppo
2021-05-19	19m	Sviluppo
2021-05-20	1h 26m	Sviluppo
2021-05-20	2h 59m	Sviluppo
2021-05-20	53m	Sviluppo
2021-05-20	18m	Sviluppo
2021-05-21	4h 32m	Sviluppo
2021-05-22	2h 28m	Sviluppo
2021-05-23	1h 12m	Documentazione
2021-05-23	1h 2m	Sviluppo
2021-05-23	1h 13m	Sviluppo
2021-05-23	16m	Code review

continues on next page

Tabella 3 – continua dalla pagina precedente

Data	Durata	Attività
	50h 41m	Totale
2021-05-24	10m	Sviluppo
2021-05-24	1h 46m	Sviluppo
2021-05-24	5m	Project Management
2021-05-24	3m	Sviluppo
2021-05-24	54m	Sviluppo
2021-05-25	2h 13m	Sviluppo
2021-05-25	2h 12m	Sviluppo
2021-05-27	19m	Sviluppo
2021-05-27	34m	Documentazione
2021-05-27	1h 46m	Documentazione
2021-05-27	33m	Documentazione
2021-05-27	1h 34m	Documentazione
2021-05-28	1h	Documentazione
2021-05-28	1h 30m	Documentazione
2021-05-28	7m	Code review
2021-05-29	1h 38m	Documentazione
2021-05-29	2h 39m	Documentazione
2021-05-29	27m	Deployment

Giovanni Anniballi

Data	Durata	Attività
	18h	Totale
17/05	1h	Ricontrollo generale codice, fix piccoli typo
18/05	1h	Aggiornati files di log e refactoring documentazione
21/05	2h	Studio Jest, valutazione di test sul frontend
24/05	3h	Primi test sulle autorizzazioni
26/05	2h	Fix ai test
27/05	2h	Test autorizzazioni completati, fix
28/05	4h	Fix test malfunzionanti, refactoring test
29/05	3h	Aggiornamento documentazione, aggiunti ulteriori test seguendo i suggerimenti di SQ

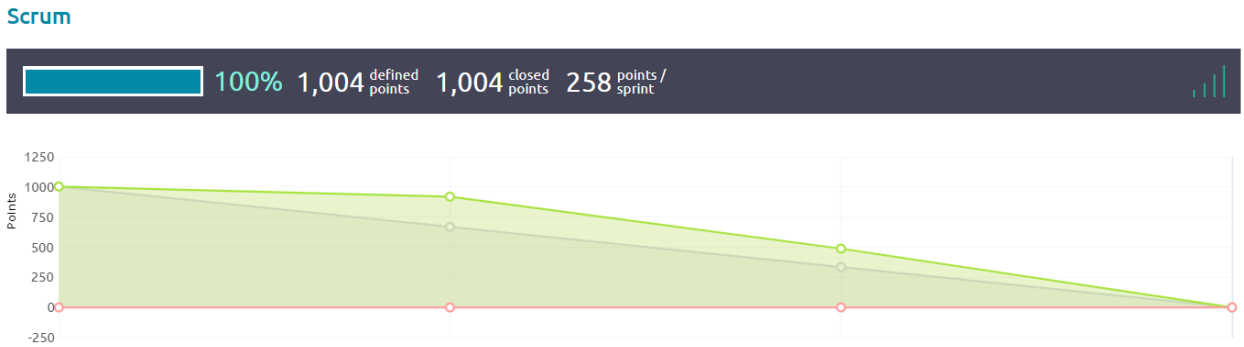
Giorgio Minoccari

Data	Durata	Attività
	22h	Totale
17/05	4h	Aggiunta associazione tra singoli tweet e condizioni che ne hanno scaturito il download
18/05	3h	Aggiunto salvataggio delle immagini presenti nei tweet e data in cui sono stati postati
20/05	3h	Ristrutturazione del crawler in uno script lanciabile separatamente su tutte le repository
21/05	2h	Aggiunta degli alert e delle azioni che vengono svolte quando scatta il trigger
24/05	2h	Bugfixing
25/05	4h	Refactoring di sicurezza
27/05	3h	Refactor struttura per poter associare le condizioni degli allarmi ai singoli tweet
28/05	6h	Completamento generale e bugfixing

Lorenzo Balugani

Data	Durata	Attività
	22h	Totale
17/05	4h	Bugfixing, supporto alla localizzazione degli errori
18/05	3h	Bugfixing
20/05	3h	API autorizzazioni, refactoring
21/05	2h	Gestione tweet, rappresentazione tweet
24/05	2h	Bugfixing
25/05	4h	Docs, refactoring
27/05	3h	Bugfixing
28/05	6h	Bugfixing

6.3 Burndown chart complessiva



CAPITOLO 7

Artefatti

Gran parte degli artefatti sono stati elencati nei singoli sprint.

Vengono di seguito elencati alcuni artefatti che non sono stati precedentemente elencati:

7.1 Diagrammi UML

Per la realizzazione dei diagrammi UML, utili nelle fasi iniziali di progettazione di N.E.S.T., è stato utilizzato DrawIO (oggi diagrams.net). Si allega il file con il progetto UML.

- `Diagrammi UML`

Dopo aver scaricato il file, recarsi su https://app.diagrams.net/ e tramite il menù `File` → `Open from` → `Device...` caricare il file appena scaricato.

7.2 Demo

7.2.1 Video

Il video di demo è disponibile al seguente URL:

- https://drive.google.com/file/d/15o70Ffe51CNj8LTKHC9dGiqRVnbv9UpZ/view

7.2.2 Interattiva

Una demo interattiva è disponibile ai seguenti URL:

- https://api.nest.steffo.eu/ (backend)
- https://prod.nest.steffo.eu/ (frontend pre-popolato)
- https://docs.nest.steffo.eu/ (documentazione e relazione)

CAPITOLO 8

Conclusioni

- Il team dimostra di aver **imparato numerose nuove nozioni**, acquisendo dimestichezza con vari linguaggi e software, come:
  - Python
  - NodeJS
  - reStructuredText
  - IntelliJ IDEA
  - Taiga
  - Git
  - GitLab
  - SystemD
  - *Twitter*
- Il team ritiene che l'**organizzazione e collaborazione** tra tutti i membri del gruppo è stata eccellente:
  - Ciascuno aveva un suo compito e non sono sorti conflitti.
  - L'organizzazione strutturata ci ha permesso di procedere a ritmo sostenuto dall'inizio alla fine, realizzando un prodotto efficace e modulare.
- Parte del team ritiene che **alcune attività**, come le varie riunioni, **sarebbero state più edificanti e veloci** se fossero state effettuate in presenza, purtroppo la pandemia e la distanza non lo hanno reso possibile.
- Parte del team ritiene invece che il **lavoro a distanza** ci ha permesso di adottare metodi migliori di **comunicazione asincrona**, permettendo ai membri di lavorare indipendentemente uno dall'altro in base alle proprie disponibilità di tempo.

CAPITOLO 9

Suggerimenti relativi al corso

- Il team concorda sull'**utilità del progetto** poiché permette di utilizzare le nozioni viste a lezione nella pratica, facendo quindi esperienza sulle varie metodologie da adottare e le varie problematiche in cui si può incorrere.
- Il team concorda nel dire che **il progetto è eccessivamente corposo**:
  - Tecnicamente a 6 CFU dovrebbero corrispondere circa 125 ore di lavoro, mentre questo progetto ne ha richieste molte di più.
  - Suggeriamo quindi di **alleggerire significativamente il carico di lavoro**, e renderlo più proporzionato ai crediti previsti.
- Il team suggerisce di lasciare agli studenti **scelta completamente libera sugli strumenti da adottare**, in quanto si sono verificati svariati problemi con quelli previsti dal corso:
  - Le istanze di Taiga, GitLab e SonarQube ospitate dall'Università si sono dimostrate inaffidabili, portando il team a dover ospitare le proprie e quindi a perderci una fetta significativa del proprio tempo.
    - CAS è supportato solo su Debian, e lo script di installazione esegue operazioni con effetti collaterali sul Docker del sistema host.
    - L'installazione di Taiga tramite Docker Compose ha richiesto 5 ore, in quanto una race condition presente nel file `docker-compose.yml` impediva al software di raggiungere il database.
    - L'installazione e manutenzione di GitLab ha richiesto 5 ore, e richiede competenze di amministrazione di sistema avanzate che sono oltre il livello dello studente medio del terzo anno di Informatica.
  - Taiga, per quanto funzionale, è un po' acerbo a livello di User Experience, il che ha portato a svariati grattacapi durante il suo utilizzo da parte del team.
    - Inoltre, è in parte superfluo: quasi tutte le funzionalità che fornisce sono già implementate sui sistemi di issues di GitHub e GitLab.
  - Utilizzare GitLab invece che il più popolare GitHub impedisce di sfruttare la licenza gratuita o universitaria di numerosi strumenti di Continuous Integration e Deployment, quali GitHub Actions, Read the Docs e Render.
  - SonarQube, per quanto semplice da installare, è molto complesso da utilizzare: il team ha necessitato di parecchie ore per capirne il funzionamento.

    * È forse anche superfluo: tutte le issues che ha segnalato erano già state segnalate in precedenza dal sistema di linting di IntelliJ IDEA Ultimate.

- Il team concorda nel dire che sarebbe molto utile la **partecipazione diretta del prof** durante le prime fasi di progetto, all'inizio del lavoro, per instradare il gruppo nella giusta direzione ed evitare l'effetto "salto nel buio":
  - Sarebbe stato piacevole avere una spiegazione più dettagliata delle regole dello Scrumble, e soprattutto sul come giocarlo a distanza.
  - Sarebbero state utili lezioni più specifica sui tool da utilizzare, come SonarQube.

CAPITOLO 10

Meta-documentazione

La documentazione è scritta in reStructuredText, ed è generata con Sphinx.

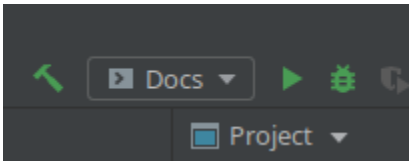
I dati relativi alla documentazione si trovano nella directory standard `/docs`; in particolare, le sorgenti reStructuredText si trovano in `/docs/source`, mentre l'ultima versione compilata si trova in `/docs/build`.

Questa struttura permetterebbe l'integrazione con lo strumento di "Continuous Documentation" ReadTheDocs, qualora il progetto fosse ospitato su GitHub o disponessimo di una licenza almeno Basic.

10.1 Compilazione con IntelliJ IDEA

È possibile compilare manualmente la documentazione con IntelliJ IDEA.

Su sistemi Linux, è disponibile la Run Configuration `Docs`, che compilerà la versione HTML della documentazione se eseguita.



10.2 Compilazione con GNU Make

È possibile compilare manualmente la documentazione con GNU Make.

Come prima cosa, è necessario *Installare le dipendenze Python* del progetto:

```
user:g2-progetto$ poetry install
```

Poi, si entri all'interno del virtual environment:

```
user:g2-progetto$ poetry shell
```

Una volta all'interno del venv, si entri nella cartella della documentazione:

```
user:g2-progetto$ cd docs
```

Infine, si esegua il target `html` del Makefile:

```
user:g2-progetto/docs$ make html
```

10.3 Compilazione con Windows Powershell

È possibile compilare manualmente la documentazione con Windows Powershell.

Come prima cosa, è necessario *Installare le dipendenze Python* del progetto:

```
g2-progetto> poetry install
```

Poi, si entri all'interno del virtual environment:

```
g2-progetto> poetry shell
```

Una volta all'interno del venv, si entri nella cartella della documentazione:

```
g2-progetto> cd docs
```

Infine, si esegua lo script `make.bat` con il parametro `html`:

```
g2-progetto/docs> make html
```

CAPITOLO 11

Struttura del database

`class Alert`

Un alert è un allarme impostato da un utente che si "attiva" quando un numero di tweet che rispetta certe condizioni (poste in and oppure or) supera una certa soglia, indicata dall'utente.

Ogni volta che l'alert si attiva, viene creata una "notifica", ovvero una entry nella tabella Notifications. Questo permette di tenere conto del numero di volte in cui l'alert viene triggerato.

Gli alert sono legati al repository di appartenenza, e quando uno di essi viene allertato viene inviata una mail all'admin e pubblicato un tweet sull'account Twitter usato per le analisi.

La tabella alert contiene le seguenti colonne:

	Definizioni
id (INTEGER, PK)	l'identificativo dell'alert
name (VARCHAR, NOT NULL)	il nome dell'alert
limit (INTEGER, NOT NULL)	il numero di tweet che innescano l'alert
window_size (INTEGER, NOT NULL)	numero di ore in cui il limit può venire superato
evaluation_mode (ENUM/SMALLINT, NOT NULL)	può essere posto a all_or oppure all_not
repository_id (INTEGER, FK, NOT NULL)	

`class Authorization`

Una autorizzazione è un'entità che rappresenta il permesso, concesso dal creatore del repository ad un altro utente, di ispezionare il contenuto di un repo e di eseguire analisi su di esso.

La tabella authorization contiene le seguenti colonne:

	Definizioni
rid (INTEGER, PK, FK)	id del repository
email (VARCHAR, PK, FK)	email dell'utente

`class Composed`

Composed è una tabella le cui righe indicano l'appartenenza di un Tweet ad un certo repository.

La tabella Composed contiene le seguenti colonne:

	Definizioni
rid (INTEGER, PK, FK)	id del repository
snowflake (VARCHAR, PK, FK)	id del tweet

class Condition

Una condizione è un elemento che viene usato da repository e alert per cercare e classificare i tweet.

Le condizioni possono essere di diversi tipi:

- **hashtag**: valore `0`, richiede che il tweet contenga un dato hashtag
- **time**: valore `2`, richiede che il tweet sia stato pubblicato prima o dopo una certa data
- **coordinates**: valore `3`, richiede che il tweet sia stato pubblicato entro un certo raggio da delle coordinate
- **user**: valore `5`, richiede che il tweet sia stato pubblicato da un dato utente

La tabella condition contiene le seguenti colonne:

	Definizioni
id (INTEGER, PK)	id della condition
type (ENUM/SMALLINT, NOT NULL)	tipo del contenuto
content (VARCHAR, NOT NULL)	contenuto della condition
repository_id (INTEGER, FK, NOT NULL)	

class Contains

Contains è una tabella le cui righe indicano la presenza di una certa condition rispetto ad un certo tweet.

La tabella contains contiene le seguenti colonne:

	Definizioni
cid (INTEGER, PK, FK)	id della condition
snowflake (VARCHAR, PK, FK)	id del tweet

class MadeOf

MadeOf è una tabella le cui righe indicano il legame tra un alert e una certa condition.

La tabella madeof contiene le seguenti colonne:

	Definizioni
aid (INTEGER, PK, FK)	id dell'alert
cid (INTEGER, PK, FK)	id della condition

class Notification

Una notification è un'entità che consente di tenere traccia del momento in cui un certo alert si è attivato per l'ultima volta.

La tabella notification contiene le seguenti colonne:

	Definizioni
id (INTEGER, PK)	id della notifica
ora (TIMESTAMP, NOT NULL)	timestamp di attivazione
alert_id (INTEGER, FK, NOT NULL)	

class Repository

Un repository è un "contenitore" di tweet, a cui sono legati alert, autorizzazioni di lettura e condizioni.

Le condizioni possono essere messe in and oppure or, inoltre un repository può venire archiviato prima divenire eliminato. Quando un repository non è archiviato, questo viene riempito di tweet su base oraria, cosa che non accade se viene archiviato.

La tabella repository contiene le seguenti colonne:

	Definizioni
id (INTEGER, PK)	id del repository
name (VARCHAR, NOT NULL)	nome del repository
start (TIMESTAMP)	timestamp di partenza del repository
end (TIMESTAMP)	timestamp di chiusura del repository
is_active (BOOLEAN, NOT NULL)	flag per segnalare se il repo è aperto o meno
evaluation_mode (ENUM/SMALLINT, NOT NULL)	può essere posto a all_or oppure all_not
owner_id (VARCHAR, FK, NOT NULL)	email del proprietario
is_deleted (BOOLEAN, NOT NULL)	flag per segnalare se l'oggetto è eliminato o meno

class Tweet

Un tweet è un'entità che viene raccolta dal componente crawler, e quando viene inserita nella base di dati viene legata ad un repository e alle condition che contiene. Un tweet contiene informazioni relativamente a chi l'ha creato, eventuali immagini, il tempo di creazione, il tempo di inserimento nel db e l'opzionale posizione legata al tweet.

La tabella tweet contiene le seguenti colonne:

	Definizioni
snowflake (VARCHAR, PK)	id univoco del tweet
content (VARCHAR)	contenuto del tweet
location (VARCHAR)	stringa contenente informazioni sulla posizione
place (VARCHAR)	riservato per sviluppi futuri
poster (VARCHAR)	informazioni sull'utente che ha creato il tweet
insert_time (TIMESTAMP, NOT NULL)	timestamp dell'inserimento del tweet
image_url (VARCHAR)	link alle immagini, se presenti
post_time (TIMESTAMP)	timestamp relativo all'invio del tweet

class User

Uno user è l'utilizzatore della piattaforma.

E' presente di default un utente admin, il quale può creare nuovi utenti.

La tabella user contiene le seguenti colonne:

	Definizioni
email (VARCHAR, PK)	email dell'utente
username (VARCHAR, NOT NULL)	username dell'utente
password (BYTEARRAY, NOT NULL)	sale della password, codificata usando l'algoritmo bcrypt
isAdmin (BOOLEAN, NOT NULL)	true se l'utente è admin

CAPITOLO 12

nest_backend - Web API in Python

12.1 .gestione - Metodi di utility

Gestione adds many fancy thingamajigs to the flask application, such as a login system and such.

nest_backend.gestione.**authenticate**(*username*, *password*)
Authentication method. It checks if the combination of username+password is a valid match. If not, it returns None. :param username: the user's email :param password: the user's password :return: if the credentials are correct, it returns the user. Else, it returns None.

nest_backend.gestione.**identity**(*payload*)
Authentication verification method. It checks if the user is in fact registered on the server. It is required by Flask-JWT, and shouldnt be used alone. :param payload: the reqest payload. :return: an User or None. It depends whether the user is actually registered on the platform.

nest_backend.gestione.**gen_password**(*password*)
It generates an hashed password. :param password: the password that needs to be hashed. :return: the password's hash.

nest_backend.gestione.**find_user**(*email*)

nest_backend.gestione.**admin_or_403**(*f*)

nest_backend.gestione.**repository_auth**(*f*)

nest_backend.gestione.**json_request_authorizer**(*json*, *serializable*)

nest_backend.gestione.**json_error**(*msg*, *code='errorUnknownError'*)
Returns an error in json format :param code: the code of the error according to the spec. :param msg: the error message. :return: a json formatted string.

nest_backend.gestione.**json_success**(*data*)
An happy little function. Its happy because the operation was successful. :param data: the thing you want to be returned :return: a json formatted string

nest_backend.gestione.**error_handler**(*e*)

nest_backend.gestione.**hashtag_validator**(*hashtag*)

12.2 .database - Database

This module imports all the tables and the declarative base

12.2.1 .base - Estensione flask

12.2.2 .tables - Tabelle

This module contains all database classes.

class nest_backend.database.tables.**Alert**(***kwargs*)

Basi: sqlalchemy.orm.decl_api.Model

id

name

limit

window_size

evaluation_mode

repository_id

repository

notifications

conditions

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Authorization**(***kwargs*)

Basi: sqlalchemy.orm.decl_api.Model

rid

email

repository

user

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Composed**(***kwargs*)

Basi: sqlalchemy.orm.decl_api.Model

rid

snowflake

repository

tweet

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Condition**(***kwargs*)

Basi: sqlalchemy.orm.decl_api.Model

id

type

content

repository_id

repository

tweets

alerts

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Contains**(***kwargs*)

Basi: sqlalchemy.orm.decl_api.Model

cid

snowflake

condition

tweet

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Notification**(***kwargs*)

Basi: sqlalchemy.orm.decl_api.Model

id

ora

alert_id

alert

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Repository**(***kwargs*)

Basi: `sqlalchemy.orm.decl_api.Model`

id

name

start

end

is_active

is_deleted

evaluation_mode

owner_id

owner

authorizations

tweets

alerts

conditions

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**Tweet**(***kwargs*)

Basi: `sqlalchemy.orm.decl_api.Model`

snowflake

content

location

place

poster

insert_time

post_time

image_url

repositories

conditions

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**User**(***kwargs*)

Basi: `sqlalchemy.orm.decl_api.Model`

email

username

password

isAdmin

owner_of

authorizations

to_json()

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**MadeOf**(***kwargs*)

Basi: `sqlalchemy.orm.decl_api.Model`

aid

cid

alert

condition

__init__(***kwargs*)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

class nest_backend.database.tables.**ConditionType**(*value*)

Basi: enum.Enum

Vedi Specifica delle Conditions su GitLab .

hashtag = 0

location = 1

time = 2

coordinates = 3

place = 4

user = 5

class nest_backend.database.tables.**OperationType**(*value*)

Basi: enum.Enum

An enumeration.

assign = 3

class nest_backend.database.tables.**ConditionMode**(*value*)

Basi: enum.Enum

An enumeration.

all_or = 0

all_and = 1

12.3 .routes - Percorsi API

Nota: La documentazione dei percorsi API è gestita da Swagger UI, disponibile a /docs una volta che il webserver del backend è avviato.

Si allega una stampa dell'interfaccia swagger:

- Schermata di swagger

CAPITOLO 13

nest_crawler - Crawler in Python

nest_crawler.**associate_condition_tweet**(*conditions_type*, *tweet*)

nest_crawler.**authenticate**()

nest_crawler.**is_coordinate_inside_bounding_box**(*latitude*, *longitude*, *radius*, *tweet_latitude*, *tweet_longitude*)

nest_crawler.**is_repo_alert_triggered**(*repository_id*)

nest_crawler.**search_repo_conditions**(*repository_id*)

nest_crawler.**send_notification_email**(*alert*)

nest_crawler.**send_notification_tweet**(*alert*)

CAPITOLO 14

`nest_frontend` - Interfaccia utente in React

Avvertimento: JSDoc non sembra essere in grado di gestire correttamente il lato frontend del progetto.

Per maggiori dettagli sul funzionamento dei vari componenti, si suggerisce di guardare le **docstring presenti nel codice sorgente**.

Seguono sotto le uniche classi che JSDoc è stato in grado di gestire (anche se vengono mostrate solo parzialmente).

14.1 `.objects` - Oggetti vari di utility

class Condition()
Condition class for an undefined/unknown condition.

See [the Condition spec](https://gitlab.steffo.eu/nest/g2-progetto/-/wikis/sprint-2/Specifica-delle-Conditions).

class ConditionHashtag()
Require a tweet to contain a specific hashtag to be gathered.

class ConditionUser()
Require a tweet to be posted by a certain user to be gathered.

class ConditionTime()
Require a tweet to be posted before or after a certain time to be gathered.

class ConditionLocation()
Require a tweet to have coordinates associated and to be posted within the {@link MapArea}.

class NotImplementedError()
Error thrown when a function is not implemented in the current class/instance.

class BackendCommunicationError()
An error in the N.E.S.T. frontend-backend communication.

class ViewNotAllowedError()
Error thrown when trying to access a backend view which doesn't exist or isn't allowed in the used hook.

class ServerNotConfiguredError()
Error thrown when trying to access a backend view when outside a {@link ContextServer}.

class FetchAlreadyRunningError()
Error thrown when trying to access a backend view while another access is ongoing.

This is not allowed due to potential race conditions.

class FetchError()
Abstract class for {@link DecodeError} and {@link ResultError}.

class DecodeError()
Error thrown when the frontend can't parse the data received from the backend.

class ResultError()
Error thrown when the backend returns a falsy *"result"* value.

class SerializationError()
Error thrown when a string couldn't be serialized into an object.

class Filter(*negate=false*)
A filter applicable in the Analysis mode.

Parametri

- **negate** -- If the filter output should be reversed.

class FilterContains()
Checks if a tweet contains a string.

class FilterHashtag()
Check if a tweet contains an hashtag.

class FilterPoster()
Check if a tweet was posted by a certain user.

class FilterWithLocation()
Check if a tweet contains *location* metadata.

class FilterWithPlace()
Check if a tweet contains *place* metadata.

class FilterInsideMapArea()
Check if a tweet's *location* is inside a {@link MapArea}.

class FilterInsideTimeRay()
Check if a tweet's *post_time* is inside a {@link TimeRay}.

class FilterWithImage()
Check if a tweet has an associated *image_url*.

class FilterIsRetweet()
Check if a tweet is a retweet.

CAPITOLO 15

Altri collegamenti

- genindex
- modindex

Indice del modulo Python

n

Indice

Simboli

A

B

C

M

N

O

P

R

S

T

U

V

W