QATicket Development Guide

QATicket Development Guide

Developer Guide

Tecnologie utilizzate

- **Python 3.8**: utilizzato per lo sviluppo del backend
- **VueJS 3.0**: utilizzato per lo sviluppo responsive del frontend
- **JQuery 3.6**: utilizzato come libreria base per javascript del il frontend
- **Bootstrap 5**: layout di base css+js per sviluppo responsive e mobile friendly del frontend
- **Shibboleth2**: modulo Apache2 per l'autenticazione con il SSO UniMore.

Dipendenze Python

- **Django=3,<4**: framework di base per lo sviluppo del frontend
- **djangorestframework~=3.12.4**: estensione di Django per l'implementazione di REST API
- **django-polymorphic**: estensione di Django che permette la creazione di modelli polimorfici. Questa libreria semplifica l'implementazione di proprietario del ticket, che può essere sia una organizzazione che un individuo.
- **django-crispy-forms~=1.11.2**: estensione di Django che permette di creareform DRY (Don't Repeat Yourself), con la minor quantità di codice di templating possibile.
- **crispy-bootstrap5**: estensione per bootstrap5 per django-crispy-forms.
- **django-q**: libreria utilizzata per la schedulazione degli eventi.
- **redis**: utilizzata da django-q per la comunicazione tra backend e cluster di scheduling (`qcluster`).
- **croniter**: dipendenza di django-q per la schedulazione con regole cron-like.
- **django-ses**: utilizzato per l'invio di email tramite il servizio Amazon AWS Simple Email Service.
- **python-docx**: libreria utilizzata per la generazione dei report `.docx`.

Dipendenze Python Opzionali

- **mysqlclient**: per il supporto al database mysql

- **django-extensions**: utilizzato in modalità di sviluppo / debug per listare i link generati dal router REST.

Dipendenze Bootstrap

- **Bootstrap Icons 1.5**: fornisce le icone utilizzate per il frontend

Features

- Login con SSO unimore
- Login utenti locali (debug/amministrazione) al link https://qaticket.ing.unimore.it/s3cr3tl0g1n
- Configurazione Apache (reverse proxy + shibboleth + file statici + uploads)
- Pannello Amministrazione di base di Django
- API Rest
- Creazione e chiusura di Ticket
- Dettaglio ticket: visualizzazione e creazione di un nuovo evento
- Dettaglio ticket: possibilità di sottoscriversi ai ticket
- Eventi ticket: Apertura, Chiusura, Note, Informazioni Richieste, Duplicato, Escalation.
- Tags
- Organizzazioni: visualizzazione e gestione delle organizzazioni (aggiunta / rimozione utenti da parte dell'amministratore dell'organizzazione)
- Creazione ticket come organizzazione
- Scheduling degli eventi
- Schedulazione creazione ticket
- Allegati
- Schedulazione eventi con allegati
- Schedulazione cancellazione degli allegati non utilizzati
- Invio notifiche via mail con postfix
- Invio notifiche via mail con AWS
- Schedulazione invio notifiche via mail per non impallare il server web
- Client responsive
- Index con attività recenti, ticket creati e seguiti
- Mobile friendly

Organizzazione del codice

Il codice risiede nella cartella **uts/**, ed è organizzato con una gerarchia di cartelle simile alla seguente:

```
media
uts
uts_common
    management
        commands
```

```
    migrations
    static
        images
        js
            components
        styles
    templates
        uts
uts_report
    templates
        uts_report
uts_rest
uts_scheduler
    management
        commands
    templates
        uts_scheduler
uts_shibboleth
    templates
        uts_shibboleth
```

- uts/: contiene le impostazioni di Django ed il mapping degli url.
- uts_common/: modulo principale, così strutturato:
  - management/commands/: contiene gli script `populatedb` e `createadmin` utilizzabili tramite lo script `manage.py`, utili per la creazione di un db sperimentale per lo sviluppo.
  - static/: contiene immagini, javascript (**Vue**) e stili CSS.
    - js/components/: componenti **Vue** modulari utilizzati negli script delle pagine.
  - templates/: contiene i template html Django + Vue per il sito di base
  - decorators.py: contiene decoratori utili, utilizzati nel resto del programma.
  - signals.py: contiene i segnali, che eseguono codice dopo l'avvenimento di determinati eventi.
- uts_report/: modulo per la generazione dei report.
  - generators.py: funzioni utilizzate per generare i reports
- uts_rest/: modulo che offre le API REST.
  - serializers.py: ospita le classi per la serializzazione dei dati
  - urls.py: mappa le API REST sui vari link.
  - views.py: classi corrispondenti alle API rest.
- uts_scheduler/: modulo che si occupa della schedulazione degli eventi
  - management/commands: contiene lo script `createscheduledjobs` utilizzabile tramite lo script `manage.py`. Questo script viene utilizzato nella fase di deployement per creare i di manutenzione job ripetuti con cadenza fissa.
  - schedules.py: contiene i job che vengono schedulati utilizzando

django-q e che verranno poi dunque eseguiti dal `qcluster`.

- uts_shibboleth/: modulo che si occupa dell'autenticazione tramite il SSO di UniMore.

Installazione dell'ambiente di sviluppo

È possibile preparare un'ambiente di sviluppo semplicemente eseguendo lo script di installazione dalla root directory di questa repository.

Una volta installato il venv, sarà necessario installare **Redis** per permettere la comunicazione tra il django-q qcluster ed il webserver.

Dunque, bisogna configurare Django, facendo bene attenzione alle impostazioni per django-q.

Successivamente, creare il database con i seguenti comandi

```
./manage.py migrate
./manage.py createscheduledjobs
```

Una volta configurato, per avviare il server, aprire su un terminale in webserver con

```
./manage.py runserver
```

E su un altro terminale il q-cluster

```
./manage.py qcluster
```

Scelte progettuali e limiti

Ho preso molte scelte e alcune non sono state semplici, il discorso delle organizzazioni è molto complesso, soprattuto dal punto di vista della sicurezza. Per finire lo sviluppo in tempo breve ed evitare disastri, ho preso queste decisioni:

1. Le organizzazioni possono essere create solo dagli amministratori del sito tramite il pannello di amministrazione
2. L'amministratore di un organizzazione può aggiungere e rimuovere utenti dalla stessa
3. Tutti gli utenti che fanno parte di un organizzazione possono creare ticket come la stessa
4. Chiunque può vedere e seguire qualunque ticket, nella index però appariranno solo i propri ticket e i ticket seguiti.
5. Quando viene aperto un ticket come un organizzazione, tutti gli utenti all'interno della stessa vengono automaticamente iscritti al ticket.
6. Le notifiche via mail, vengono inviate solo agli utenti che le hanno abilitate dalle impostazioni e solo sui ticket a cui sono iscritti.