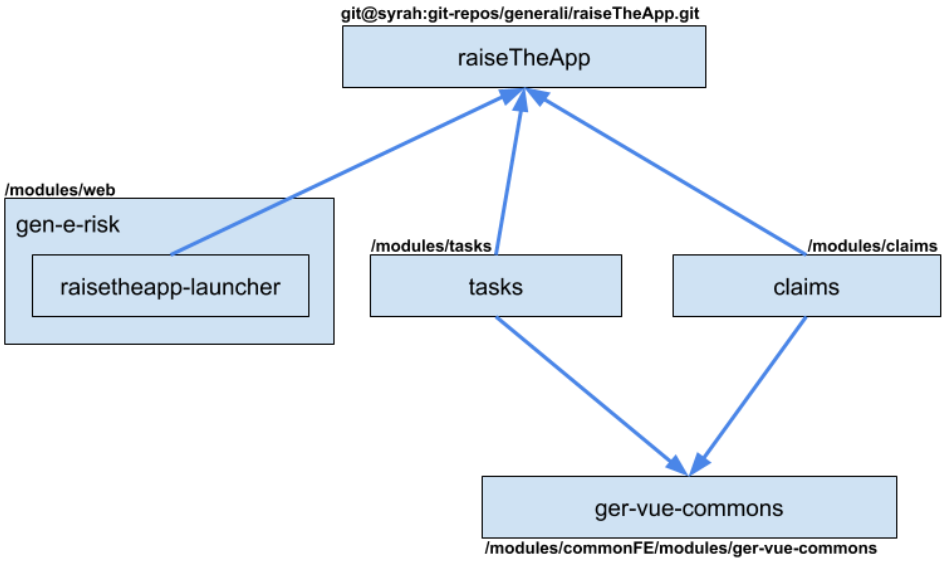
Sunto dei moduli presenti sul FE.
==========
Il codice FE è attualmente suddiviso nel modo seguente.

generisk (angular, jQuery, cocoon):
/modules/web/src/main/webapp/opengenerisk/ui/client

ger-vue-commons, **tasks**, **claims** (Vue.js):
/modules/commonFE/modules/ger-vue-commons
/modules/tasks/src/main/tasks-ui-src
/modules/claims/src/main/claims-ui-src

raiseTheApp, **raisetheapp-launcher** (Vue.js, typescript):
git@syrah:git-repos/generali/raiseTheApp.git
/modules/web/src/main/webapp/opengenerisk/ui/client/raisetheapp-launcher

Una cosa simile a quella riassunta in questa figura:



Qui di seguito alcune note relative ai vari progetti.

==========
/modules/tasks/src/main/tasks-ui-src
/modules/claims/src/main/claims-ui-src
==========
Se vengono effettuate modifiche a questi progetti il rebuild della parte FE parte in automatico con la build del modulo.
Il tutto è guidato dal “pom.xml” del modulo, dalle istruzioni identificate da:
<artifactId>frontend-maven-plugin</artifactId>

Per lo sviluppo “veloce” FE di Tasks e Claims si può usare la modalità “dev”, senza dover buildare tutto il progetto:
cd /modules/tasks/src/main/tasks-ui-src
npm run dev

In questa modalità è possibile mockare le vere chiamate rest usando un proxy definito (ad esempio per Tasks) in **“/tasks-ui-src/config/index.js”**, nella sezione “**dev.proxyTable**”.
Ad esempio la seguente configurazione fa da proxy alla chiamata **“https://gerci.exmachina.ch/tasks/tasks”** (quella che torna i risultati di ricerca a partire dal form di ricerca avanzata) e ritorna il file definito in “**/modules/tasks/src/main/tasks-ui-src/static/mock/data/search_results.json**”:

```
'/tasks': {
        target: `http://localhost:${port}`,
        changeOrigin: true,
        ws: true,
        pathRewrite: function (path, req) {
                return '/static/mock/data/search_results.json';
        },
        router: {}
}
```

==========
/modules/commonFE/modules/ger-vue-commons
==========
Contiene codice condiviso dalle applicazioni Tasks e Claims.
Quando si fa qualche modifica ai suoi sorgenti occore rebuildarlo, in modo che Tasks e Claims poi aggiornino la dipendenza.
Questo build purtroppo al momento non è ancora automatizzato.
Per lanciarlo ci sono due modi.

== Modo 1 ==
cd /modules/commonFE/modules/ger-vue-commons
npm run build

== Modo 2 ==
Metodo “easter egg”: questo progettino è una sorta di dashboard che esegue la compilazione di ger-vue-commons, Claims e Tasks in un colpo solo.

cd /modules/commonFE/modules/ger-dashboard
npm run go:skip.test

==========
raiseTheApp (‘a campanella)
==========
Vive in un suo repository indipendente, come progetto a se stante:
git@syrah:git-repos/generali/raiseTheApp.git

Quindi quando si effettuano modifiche va rebuildato e poi pushato sul repository prima di poter essere usato dalle altre applicazioni.
Per buildarlo lanciare due comandi (al momento sono previste due tipologie di build perchè usato in maniera leggermente diversa in Generisk e in applicazioni “Vue native” come Tasks e Claims):

cd /raiseTheApp
npm run build <=== build 1
npm run build:standalone <=== build 2

In Tasks/Claims, essendo applicazioni native Vue, il modulo “raiseTheApp” (sviluppato con Vue) è usato come una normale dipendenza NPM.
Viene quindi dichiarato nel package.json e poi importato all’occorrenza:
"raisetheapp": "git+ssh://git@192.168.50.7:git-repos/generali/raiseTheApp.git"

In Generisk (che non è un’applicazione antiva Vue) invece, per poter inserire “‘a campanella” è stata fatta una microapplicazione Vue.js di supporto che include come dipendenza raiseTheApp.
Questa applicazione si trova in
“**/modules/web/src/main/webapp/opengenerisk/ui/client/raisetheapp-launcher**”.
La build di questa applicazione è automatizzata dal “pom.xml” del modulo “web”, dalle istruzioni identificate da:
<artifactId>frontend-maven-plugin</artifactId>

È comunque possibile buildarlo a mano lanciando:
cd /modules/web/src/main/webapp/opengenerisk/ui/client/raisetheapp-launcher
npm run build

==========
ERRORI eventuali
==========
Dovessero comparire errori che bloccano la build FE in genere sono in genere dovuti a corruzioni del file “package-lock.json” o in una delle dipendente dentro la cartella “node_modules” di ciascun progetto.
In genere per risolvere (come recentemente capitato in ger-qa-prod con Tasks) basta **cancellare “package-lock.json” o tutta la cartella “node_modules”**.