

NTT Data

Global IT Innovator



Nuovo Sistema Geco++

Architettura della soluzione

Codice: LI - DA 1.0

Data emissione/ultima modifica: 08/03/2019

	<i>Nome e Cognome</i>	<i>Data</i>	
Redazione:	Maurizio Mangogna		
Verifica:			
Approvazione:	Alessandro De Simone		
Accettazione:			
Distribuzione:			
Riferimento Cliente:			

EVOLUZIONE DEL DOCUMENTO

Versione	Descrizione
1.0	Prima emissione
1.1	Aggiunte informazioni sugli ambienti Azure

INDICE

1	INTRODUZIONE.....	4
2	Architettura della soluzione.....	5
2.1	Utilizzo di Docker.....	6
3	Modalità di sviluppo	6
3.1	Continuous Integration	6
3.2	Continuous Deployment	8
4	Ambienti di Collaudo e Produzione su AZURE	8

INDICE DELLE FIGURE

Figura 1 - Architettura della soluzione	5
Figura 2 - Esempio di dashboard SonarQube	7
Figura 3 - Esempio di utilizzo di CodeBrag.....	8

INDICE DELLE TABELLE

No table of figures entries found.

1 INTRODUZIONE

Questo documento descrive l'architettura proposta per nuovo sistema Geco++ di Lazio Innova, destinato a sostituire gli attuali sistemi Geco e GecoWeb.

La realizzazione del sistema sarà effettuata utilizzando un approccio Agile e in una prima wave, della durata di circa un anno, sarà raggiunto l'obiettivo di implementare una prima versione MVP (Minimum Viable Product) del sistema – cioè un sistema in grado di sostituire efficacemente i preesistenti sistemi Geco/GecoWeb pur riducendo al minimo indispensabile il numero di funzionalità implementate, allo scopo di contenere costi e tempi di realizzazione.

Wave successive potranno aggiungere nuove funzionalità e potenziare le funzionalità già esistenti.

Questo tipo di approccio rende importante definire opportunamente le modalità di realizzazione del progetto assicurando la necessaria:

- *Transferability*, qualità del software che indica quanto sia facile assegnare ad un nuovo team di sviluppo la gestione degli sviluppi manutentivi ed evolutivi – avere, cioè, un basso livello di *resource lock-in*
- *Changeability*, qualità che indica quanto sia semplice, in termini di tempi e costi, l'evoluzione del software a fronte di richieste di nuove funzionalità
- *Maintenability*, qualità che indica quanto sia semplice, in termini di tempi e costi, effettuare manutenzione sul software

Per indirizzare questa esigenza, questo documento includerà una specifica sezione che illustrerà l'impostazione tecnica proposta.

La realizzazione della prima wave sarà articolata in tre fasi:

- Fase 1
 - Gestione Utenti
 - Gestione Anagrafiche
 - Gestione Bandi
 - Presentazione Domande
- Fase 2
 - Istruttoria
- Fase 3
 - Rendicontazione

La versione corrente di questo documento include elementi relativi all'intera prima wave, ma deve considerarsi esaustivo limitatamente alla Fase 1.

Versioni successive del documento includeranno gli eventuali aggiornamenti necessari all'architettura per la realizzazione delle fasi 2 e 3.

2 ARCHITETTURA DELLA SOLUZIONE

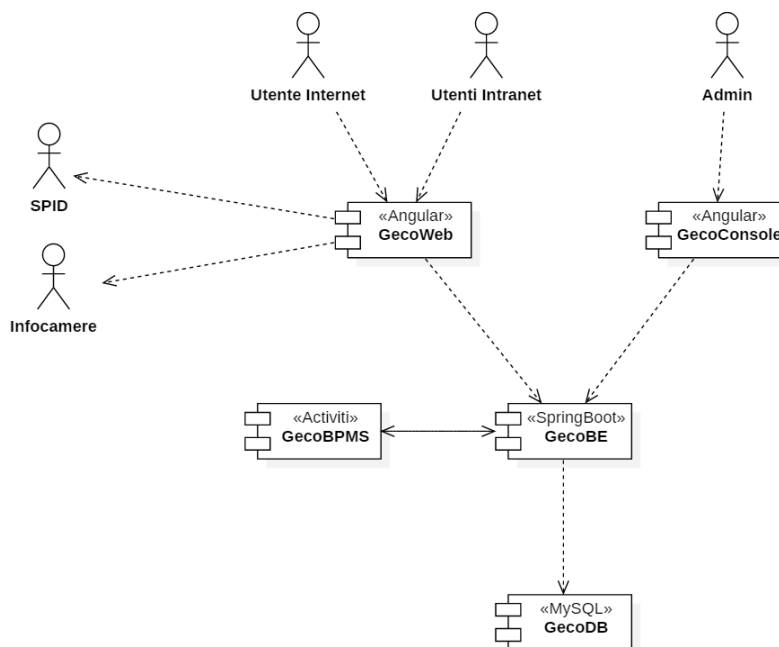


Figura 1 - Architettura della soluzione

Il diagramma in Figura 1 mostra la struttura della soluzione proposta e le interazioni con i principali componenti con cui il sistema dovrà interagire, che sono costituiti dai sistemi di autenticazione:

- **SPID**, fornito dal sistema IAM della Regione Lazio (<https://iam.regione.lazio.it/nidp/app/>)
- **Infocamere**, fornito dal sistema IAM di Infocamere (<https://www.impresainungiorno.gov.it/sso/go>) che consente alle aziende di autenticarsi mediante token

Gli attori indicati nel diagramma sono:

- **Admin**: si tratta di operatori di Lazio Innova con il compito di configurare opportunamente il sistema e di monitorarne il corretto funzionamento
- **Utente Intranet**: sono operatori di Lazio Innova che hanno il compito di svolgere specifici ruoli all'interno del processo di gestione dei bandi
- **Utente Internet**: sono utenti che accedono dal sito pubblico e svolgono sia ruoli legati alla presentazione delle richieste, sia ruoli all'interno del processo di gestione dei bandi (ad esempio la fase di istruttoria)

Ciascun attore rappresenta una tipologia di utenti del sistema che potrebbero essere soggetti ad ulteriore profilatura: ad esempio la funzionalità di Amministratore potrebbe essere articolata in varie categorie di utenti con differente visibilità delle varie funzioni di gestione del sistema.

Il sistema è costituito dai moduli seguenti, implementati utilizzando esclusivamente soluzioni open source:

- **GecoConsole**: è il modulo intranet utilizzato dall'Amministrazione per la configurazione e la gestione del sistema. Si tratta di una *Single Page Application* realizzata con tecnologia *Angular* (<https://angular.io/>) sul web server *NGINX* (<https://www.nginx.com/>).

- **GecoWeb:** è il modulo di FrontOffice, utilizzato dagli utenti internet. Il modulo è una *Single Page Application* sviluppata con *Angular* sul web server *NGNIX*. Non essendo ancora terminata la fase di progettazione della soluzione, in questa versione del documento si indica come un unico componente (appunto GecoWeb) svolga tutte le funzionalità relative all'accesso internet. E' possibile che con il completamento della fase di design si possa definire un ulteriore componente di frontend GecoBO così da poter partizionare le funzionalità fruibili da internet su due applicazioni separate, GecoFO (Front Office) per i soli utenti richiedenti e GecoBO (Back Office) per gli utenti che svolgono funzioni dell'amministrazione (ad esempio gli istruttori).
- **GecoBE:** è il modulo di backend del sistema. Espone servizi REST utilizzati dalle SPA di frontend e integra i moduli di BPMS e DB. Questo modulo è implementato con la tecnologia *Java Spring Boot* (<https://spring.io/projects/spring-boot>).
- **GecoDB:** database del sistema basato sulla piattaforma open source *MySQL* (<https://www.mysql.com/it/>).
- **GecoBPMS:** è modulo BPMS (Business Processes Management System) destinato ad implementare tutti i processi previsti dal sistema, come ad esempio la preparazione di un bando, di una richiesta, di un'istruttoria. Sarà basato sulla piattaforma open source *Activiti* (<https://www.activiti.org/>).

2.1 Utilizzo di Docker

La soluzione proposta si basa sull'utilizzo di un approccio container-based, in cui ciascun modulo sarà realizzato mediante la tecnologia Docker (<https://www.docker.com/>) e sarà progettata come applicazione cloud-native, utilizzando il modello architetturale comunemente noto come "*twelve-factor application*" (<https://12factor.net/>).

L'infrastruttura di gestione dei Docker in fase di sviluppo sarà basata sulla versione open source della piattaforma OpenShift (<https://www.okd.io/>), ma gli ambienti di collaudo e produzione potranno utilizzare qualsiasi infrastruttura di orchestrazione purché basata su tecnologia Docker, inclusa la piattaforma basata su Rancher (<https://rancher.com/>) attualmente in produzione (seppure con sole funzionalità di backend e dunque non esposta su internet) presso l'Amministrazione.

3 MODALITÀ DI SVILUPPO

3.1 Continuous Integration

Gli ambienti di sviluppo NTT DATA utilizzeranno un approccio di tipo *Continuous Integration* (CI) così articolato:

- Gli sviluppatori lavoreranno in locale e effettueranno, almeno quotidianamente, la *push* del codice sorgente dei vari moduli su un repository GIT (<https://git-scm.com/>)
- Il sistema di CI, basato sulla piattaforma open source Jenkins (<https://jenkins.io/>), scarica quotidianamente (tipicamente nelle ore notturne) tutto il codice prodotto ed effettua due operazioni principali:
 - Compilazione dei moduli e aggiornamento dei container Docker relativi nell'ambiente di test NTT DATA
 - Esecuzione dell'analisi della qualità del codice (tipicamente per i moduli basati su Java) mediante il tool open source SonarQube

Quotidianamente, dunque, il team leader, gli analisti ed i tester potranno verificare sugli ambienti di test NTT DATA la corretta esecuzione del codice prodotto il giorno precedente e verificare sulla dashboard SonarQube l'analisi di qualità.

SonarQube (<https://www.sonarqube.org/>) è un sistema di controllo automatico della qualità del codice che produce una dashboard di progetto che mostra le qualità "statiche", cioè determinate senza eseguire il codice, del software.

Questa piattaforma open source consente quindi di supervisionare in modo automatico la qualità del codice prodotto e l'utilizzo delle principali *best practices* di programmazione per i vari linguaggi presenti nel progetto.

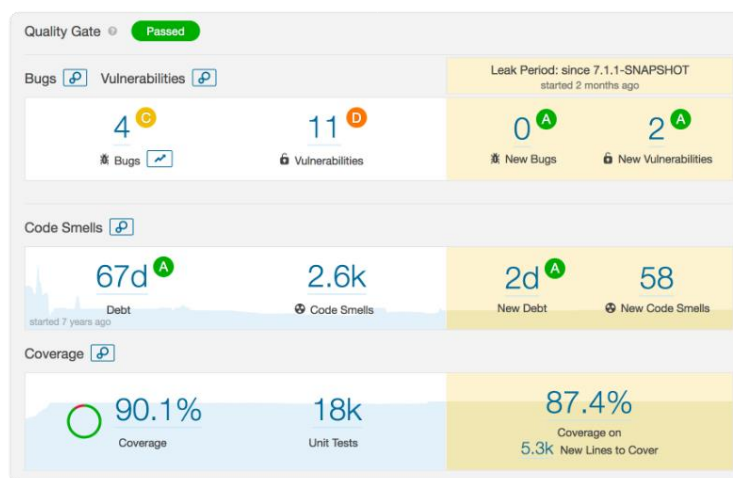


Figura 2 - Esempio di dashboard SonarQube

Qualora lo si ritenesse opportuno, ad esempio durante lo sviluppo di moduli software di particolare difficoltà, viene anche utilizzata la piattaforma open source CodeBrag (<https://github.com/softwaremill/codebrag>) per la verifica manuale della qualità del codice.

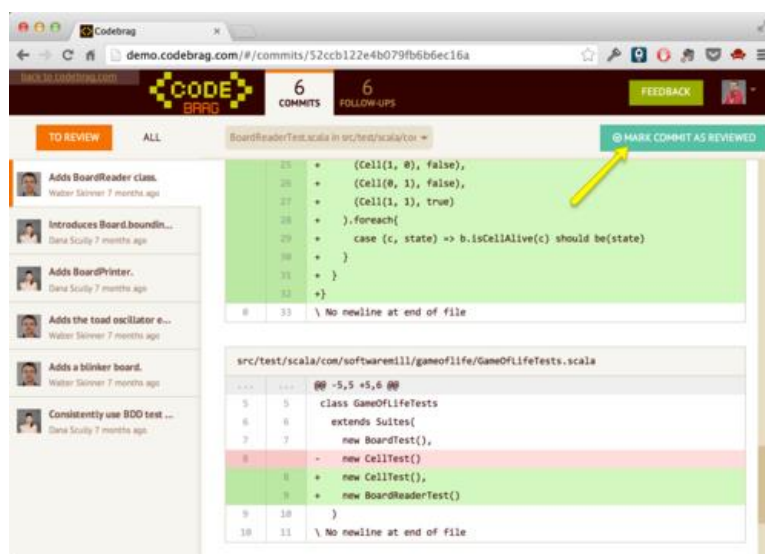


Figura 3 - Esempio di utilizzo di CodeBrag

L'utilizzo combinato di GIT, per la conservazione su repository versionato del codice sorgente, di SonarQube per l'analisi statica delle qualità del codice e di CodeBrag, per la realizzazione di sessioni di *code inspection* e *code review*, consente di assicurare la produzione di codice di alta qualità in grado di garantire buoni livelli di *manutenability*, *changeability* e *transferability*.

L'automazione gestita mediante il tool Jenkins assicura rapidità e riproducibilità dei risultati delle operazioni di compilazione ed installazione.

3.2 Continuous Deployment

Con periodicità concordata con l'amministrazione, il codice sorgente sarà inviato sugli ambienti di collaudo dell'Amministrazione in forma di codice sorgente così da poter essere compilata ed installata sull'infrastruttura di collaudo e produzione.

Lazio Innova dispone già di una infrastruttura di Continuous Deployment (basata su SVN e Jenkins), che potrà essere adattata all'utilizzo per il nuovo sistema.

4 AMBIENTI DI COLLAUDO E PRODUZIONE SU AZURE

Per la realizzazione del nuovo sistema Geco++ sono stati realizzati nuovi ambienti di collaudo e produzione sulla piattaforma cloud Microsoft Azure.

In particolare è stato realizzato un cluster per contenere gli ambienti di collaudo (test e pre-produzione) e un cluster esclusivamente per l'ambiente di produzione.

Poiché la nuova architettura è interamente basata su Docker, per la gestione dei container è stato utilizzato Kubernetes.

Ogni cluster è dotato di un controller di ingresso con funzione di bilanciamento del carico.

Sia per il cluster di collaudo che per quello di produzione, è stata usata la configurazione che segue per ciascun nodo:

CPU	4 CPU virtuali
RAM	16 GB

Al momento si prevedono:

- 3 nodi di questo tipo per l'ambiente di produzione
- 1 nodo di questo tipo per l'ambiente di collaudo

L'utilizzo di un'infrastruttura basata su Docker e di una soluzione di tipo *cloud-native* consentirà, se e quando necessario, di aumentare la potenza dell'infrastruttura semplicemente aggiungendo uno o più nodi ulteriori con le stesse caratteristiche, oppure con configurazione migliore, e ridistribuendo i container opportunamente.

Per ciascun cluster sono stati configurati:

- File system per l'archiviazione dei file, con capienza ampliabile all'aumentare del numero e dimensione dei file archiviati.
- Database MySQL v5.7

Per il database degli ambienti di collaudo la configurazione è la seguente:

CPU	1 vCore
RAM	2 GB
HD	5 GB

Invece per l'ambiente di produzione, la configurazione del database è la seguente:

CPU	4 vCore
RAM	5 GB per vCore
HD	100 GB