**Introduzione**

Titolo= Issues tracker()

**Capitolo 1. Funzionamento di Github**

**Introduzione**

* + 1. **cos’e’ Git**

Git e’un sistema software di controllo di versione distribuito e creato da Linus Torvalds(inventore del Kernel Linux) nel 2005. Esso e’ stato creato nel corso dello sviluppo del Kernel di Linux perche nel caso in cui qualche aggiornamento non avesse dato gli effetti desiderati, si poteva sempre tornare indietro e recuperare la versione funzionante senza troppi problemi. Permette in realtà di controllare e gestire gli aggiornamenti di un progetto senza sovrascrivere nessuna parte del progetto stesso. Per poterci lavorare bisogna creare progetti(Repository) in cui mettere le versioni del suo file. Git poi permette di tenere traccia delle modifiche apportate a un progetto registrando una copia dei file del progetto in un database. Sostanzialmente Git scatta una foto dei file presenti nella cartella nel momento in cui si effettua il Commit(operazione di registrazione di una modifica di uno o più file all’interno di un Repository). Per essere efficiente, se un file non è stato modificato in un Commit Git non lo memorizza nuovamente nel Repository ma crea un semplice collegamento all’ultima versione precedentemente salvata di quello stesso file. In Git i file possono avere 3 stati: Modified,Staged e Commited. Commited significa che il file è stato salvato nel proprio database locale, Modified indica che il file è stato modificato ma non ancora salvato nel database con un Commit, Staged significa che il file è stato modificato e la sua versione attuale verrà salvata nel database con il Commit successivo,cioè si prepara il file per il prossimo Commit.Un progetto Git a quindi 3 componenti principali:la Git Directory:Dove Git registra i metadati e oggetti del database, la working Directory(cartella di lavoro): copia di una versione del progetto a nostra disposizione per l’uso e la modifica dei file,la Staging area: è un semplice file, contenuto nella cartella Git, che conserva le informazioni su ciò che dovrà entrare nel prossimo Commit.Lo schema di funzionamento di Git e’ il seguente:

1. Si modifica uno o più file presenti nella working Directory
2. Si aggiunge una copia nella Staging area
3. Si esegue l’operazione di Commit(cioè si copia i file che sono presenti nella Staging area nella Git directory)

< inserire l’immagine dello dei 3 stati http://progit.org/book/ch1-3.html>

Per installare Git,scaricare la versione recente nel sito ufficiale

<http://git-scm.com/downloads>

* + 1. **cos’e’ Github?**

**2.1 Introduzione**

Fondata nel 2008 da Tom Preston-Werner, Chris Wanstrath e PJ Hyett e avente sede a San Francisco, California Github e’ un software di controllo di versione, permette di aggiornare un file senza dover sovrascrivere le versioni precedenti. Nata per ospitare **progetti software** e basata sul software *open source* Git(visto sopra ).Github offre un ambiente web dov’e e’ possibile registrarsi, creare i propri progetti e lavorarci sopra in più versione senza dove perdere le ultime modifiche fatte. gli utenti della piattaforma social potranno lavorare contemporaneamente sulla medesima versione dello stesso progetto senza timore di apportare modifiche sostanziali. Tutte le vecchie versioni saranno conservate nella propria *Repository*, così da poterle recuperare in caso di necessità. Inoltre, per ogni utente al lavoro verrà creata una differente versione del progetto, così da non creare fastidiose sovrapposizioni o sovrascritture.

**2.2 Impostare e usare Github**

Per poter usare Github bisogna:

* **creare un account Github al sito** [**https://github.com/**](https://github.com/)

Si inseriscano il proprio nome d’utente i propri dati e si sceglie una password

* **Fare settaggio del software Git**

Si scarica l’ultima versione di Git per il proprio sistema operativo al sito <http://git-scm.com/downloads> e si imposta il nome utente e l’indirizzo di posta elettronica. Per far questo gli utenti Windows dovranno aprire l'applicazione **Git Bash**  gli utenti di Mac dovranno inserire al terminale questi comandi

*git config --global user.name "il tuo nome" : per lo username*

*git config --global user.email "*your\_email@youremail.com*" : Per* l’indirizzo

* **Comandi più usati in Github**

**2.3 Progetti In Github**

Per poter lavorare in Github bisogna prima creare un progetto che conterrà tutti i file e tutti i documenti legati al progetto. ci sono 2 tipi di progetti in Github

* Progetti pubblici

Progetti visti da tutti gli utenti che hanno un account Github. Permette al social network di favorire condivisione di progetti e fare lavorare più utente su uno stesso progetto anche se fisicamente non si conoscono

* Progetti privati (a pagamento)

Progetti riservati a un utente o a un’organizzazione o a un azienda che lavora su un progetto interno e vuole rendere visibile i file del progetto sono ai utenti dell’azienda o dell’organizzazione. Questo tipo di progetto su Github si crea a pagamento.

Ogni progetto e’ ospitato in una repository e può avere più versioni e anche più attività da svolgere per raggiungere l’obiettivo del progetto finale . Nella realizzazione di un progetto ci servono delle operazioni che vorremo che entro una certa data che fossero già fatte. Queste operazioni vengo raggruppa in un’unita chiamata Milestone

**2.4 Issues**

Le Issues sono un ottimo modo per tenere traccia delle attività, miglioramenti e bug per i progetti. Sono un po’ come email, tranne che possono essere condivisi e discussi con il resto del team. La maggior parte dei progetti software hanno un bug tracker di qualche tipo. Il bug tracker di GitHub si chiama Issue, e ha la sua propria sezione in ogni repository. Una Issue ha generalmente un titolo e una descrizione che descrivono le attività, etichette colorate aiutano a classificare e filtrare i vostri problemi. Un assegnatario è responsabile per lavorare sul problema in un dato momento. Commenti permettono a chiunque con accesso al repository per fornire feedback.

**2.5 Milestones**

In ingegneria dei software, una Milestone e’ una attività a durata nulla cioè un punto in cui certe attività dovrebbero già essere effettuate. . Una Milestone agisce come un contenitore per le Issues. Questo è utile per associare le Issues con le caratteristiche specifiche o fasi di progetto (ad esempio Versione 2.1 - Ticket da Mail o Versione 2.4 - WEB Api).

* + 1. **Protocollo OAuth**
       - 1. **Introduzione**

Sviluppato da Blaine Cook e Chris Messin nel 2006 l’**Open Authorization** comunemente chiamato **OAuth**, è un **protocollo di comunicazione open** mediante il quale un’applicazione (o un servizio web) può gestire in modo sicuro l’accesso autorizzato ai dati sensibili. Il protocollo è compatibile con qualsiasi tipologia di applicazione: desktop, web e mobile quindi permette a applicazioni esterne di autenticarsi a Github e di accedere a informazioni private(ES: un progetto) di un client Github senza richiedere credenziali(cioè username e password ) del cliente. Per poter usarlo si deve per prima registrare la sua applicazione a Github e dopo la registrazione verrà fornito una unica coppia di Client ID e Client Secret che verranno poi usate dall’applicazione per fare le richieste delle informazioni .L’applicazione dovrà importare un frame work(insieme di classi ) che ci permetta di inviare le richieste e di estrarre informazioni a Github.I possibili frame work che si possono usare sono OwinOAuthProviders(<https://github.com/owinmiddleware/OwinOAuthProviders)Octokit.n> e(https://github.com/octokit/octokit.net).Quest’ultimo e’ il frame work che e’ stato usato nella nostra applicazione(daremo una sua descrizione in un capitolo successivo).

* + - * 1. **Funzionamento del protocollo**

Abbiamo 3 entità coinvolti:

* L’applicazione che richiede le informazioni, o Consumer ( che e’ la nostra applicazione:MyApplication)
* Il servizio web che fornisce le informazioni, o Service Provider (Github o Facebook o Twitter)
* L’utente dell’applicazione che e’ iscritto sulla piattaforma del Service Provider(nel nostro caso l’utente e’ iscritto a Github e ha le sue credenziali)

Come già accennato in questa introduzione, per poter usare il servizio predisposto dal Service Provider, il Consumer(la nostra applicazione) deve essere registrarsi presso il Service Provider al termine dal quale verranno dati un Client ID e Client Secret necessari per lo scambio di dati. I passi principali nella comunicazione Consumer e Service Provider sono:

* Il Consumer ridirige l’utente ad un URL predisposto dal Service Provider per la fase di autenticazione. Nella richiesta HTTP sono presenti le due chiavi ricevute in fase di registrazione (Client ID e Client Secret) e un’URL al quale l’utente deve essere ridiretto al termine dell’operazione.
* L’utente accede alla pagina di autenticazione predisposta dal Service Provider dove inserisce le proprie credenziali di accesso, di solito username e password. Di solito la pagina di autenticazione lavora con il protocollo HTTPS il quale garantisce che i dati scambiati tra browser e web server sono codificati e non vengono trasferiti in chiaro come invece avviene nel protocollo HTTP. In questo caso la comunicazione risulta molto più sicura in caso di intercettazione.
* Il Service Provider verifica l’identità dell’utente. Se la verifica ha esito positivo, gli presenta il tipo di richiesta avanzata dal Consumer che l’utente deve approvare. Tale richiesta riguarda, di solito, la durata dell’accesso e la tipologia di informazioni che verranno rese disponibili al Consumer.
* Il Service Provider ridirige l’utente all’URL inviata precedentemente dal Consumer (al punto 1). Nella richiesta HTTP è presente una stringa (detta token) che deve essere utilizzata successivamente dal Consumer per richiedere eventuali informazioni al Service Provider.
* L’utente accede nuovamente alla pagina del Consumer. Per l’utente meno esperto, il passaggio da un server ad un altro, è praticamente trasparente.
* Il Consumer invia una HTTP al Service Provider nella quale richiede, ad esempio, i dati anagrafici dell’utente corrente. Nella richiesta deve anche fornire il token precedentemente ricevuto. Le informazioni disponibili, dipendono dal Service Provider. Facebook, ad esempio, espone i dati anagrafici dell’utente oppure i suoi ultimi messaggi in bacheca; LinkedIn espone i collegamenti di un utente oppure le sue iscrizioni ai gruppi; Google fornisce moltissime informazioni come ad esempio l’elenco dei contatti in rubrica.

**Il protocollo OAuth**, come abbiamo notato, **non obbliga l’utente a fornire le credenziali di accesso al provider** (il sistema esterno). L’utente viene ridiretto sul server del Service Provider per l’autenticazione al sistema.

**Capitolo 2. Tecnologia di sviluppo Asp.net e C#**

1. **.Net**
2. **Linguaggio di programmazione C#**
3. **Tecnologia Asp.net**
4. **Modello MVC**
5. **Controllore**
6. **Vista**
7. **modello**

**Capitolo 3. Frame work Octokit.net**

1. **Introduzione**
2. **Importazione del frame work in Visual studio**
3. **Uso del frame work**

**Capitolo 5. Applicazione realizzata**

1. **Installazione dell’applicazione**
2. **registrazione dell’applicazione su Github**
3. **funzionamento dell’applicazione**

**Capitolo 6. Risultati di qualche prova**

**Conclusioni**

**Riferimenti Bibliografici**