MyTalk

Sofware di comunicazione tra utenti senza requisiti di installazione

clockworkTeam7@gmail.com

Manuale sviluppatore

v 2.0

Informazioni sul documento

Nome documento	Manuale sviluppatore
Versione documento	v 2.0
Data creazione	2013/05/20
Data ultima modifica	2013/06/30
Uso documento	Esterno
Redazione	Furlan Valentino La Bruna Agostino
Verifica	Gavagnin Jessica
Approvazione	Ceseracciu Marco
Lista distribuzione	gruppo *Clockwork* Zucchetti SPA Prof. Tullio Vardanega

Sommario

Il presente documento vuole guidare lo sviluppatore nella espansione del prodotto **MyTalk** esponendo e spiegando le funzionalità offerte.

Diario delle modifiche

Autore	Modifica	Data	Versione
Ceseracciu Marco	Approvazione del documento	2013/06/30	v 2.0
Gavagnin Jessica	Verifica del documento	2013/06/29	v 1.7
La Bruna Agostino	Approfondita sezione 5.1.1	2013/06/28	v 1.6
Furlan Valentino	Aggiunti diagrammi delle classi delle classi che descrivono le componenti che possono essere estese	2013/06/27	v 1.5
La Bruna Agostino	Aggiunti esempi di codice delle componenti che possono essere estese	2013/06/27	v 1.4
Furlan Valentino	Corretta la spiegazione rispetto la classe JavaConnectionSQLite	2013/06/26	v 1.3
Furlan Valentino	Inserite immagini che aiutino lo sviluppatore a configurare l'ambiente di sviluppo	2013/06/25	v 1.2
Furlan Valentino	Tolto il riferimento alle Norme di Progetto	2013/06/24	v 1.1
Bain Giacomo	Approvazione del documento	2013/05/26	v 1.0
La Bruna Agostino	Verifica del documento	2013/05/25	v 0.6
Gavagnin Jessica	Stesura capitolo Guida allo sviluppo	2013/05/23	v 0.5
Gavagnin Jessica	Stesura capitolo Configurazione Eclipse	2013/05/22	v 0.3
Ceseracciu Marco	Stesura capitolo Recupero Progetto per sviluppatori	2013/05/21	v 0.2
Ceseracciu Marco	Creazione documento	2013/05/20	v 0.1

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di fornire linee guida per lo sviluppo del prodotto **MyTalk**

1.2 Descrizione generale

Il prodotto **MyTalk** è uno strumento che permette la comunicazione di tipo VoIP (Voice over IP) tra due o più utenti in tempo reale tramite Internet e WebRTC, senza l'installazione di programmi od estensioni. Permette alle aziende di ridurre i costi di gestione dei servizi di comunicazione interni

1.3 Glossario

Per la spiegazione dei termini tecnici, abbreviazioni ed acronimi presenti all'interno dei documenti, si faccia riferimento al documento esterno `Glossario_v2.0.pdf`, dato in allegato. Verrà sottolineata in ogni documento la prima occorrenza del termine da specificare

2 Recupero Progetto per sviluppatori

2.1 Abilitazione

Per accedere al repository per sviluppatori del progetto bisogna richiedere l'abilitazione:

1. Iscriversi al sito GitHub (`www.github.com`)
2. Mandare una mail a clockworkteam7@gmail.com con oggetto RICHIESTA ABILITAZIONE riportando nel contenuto il nome utente da abilitare e le motivazioni della richiesta
3. Ricevere una mail di conferma dell'abilitazione dell'account; in caso di rifiuto riceverà una mail con dettagliate le motivazioni del rifiuto

2.1.1 Esempio mail

To: clockworkteam7@gmail.com
From: pincopallino@indirizzoemail.com
Object: RICHIESTA ABILITAZIONE
Message:
Nick: PincoPallino
Motivazione: Necessità di modificare parti del codice per adattarlo al proprio progetto

2.2 Creazione di una copia locale del repository

1. Accedere al sito `https://github.com/ClockWorkTeam/ClockWork` e scaricare il repository in versione compressa (Figura 1)

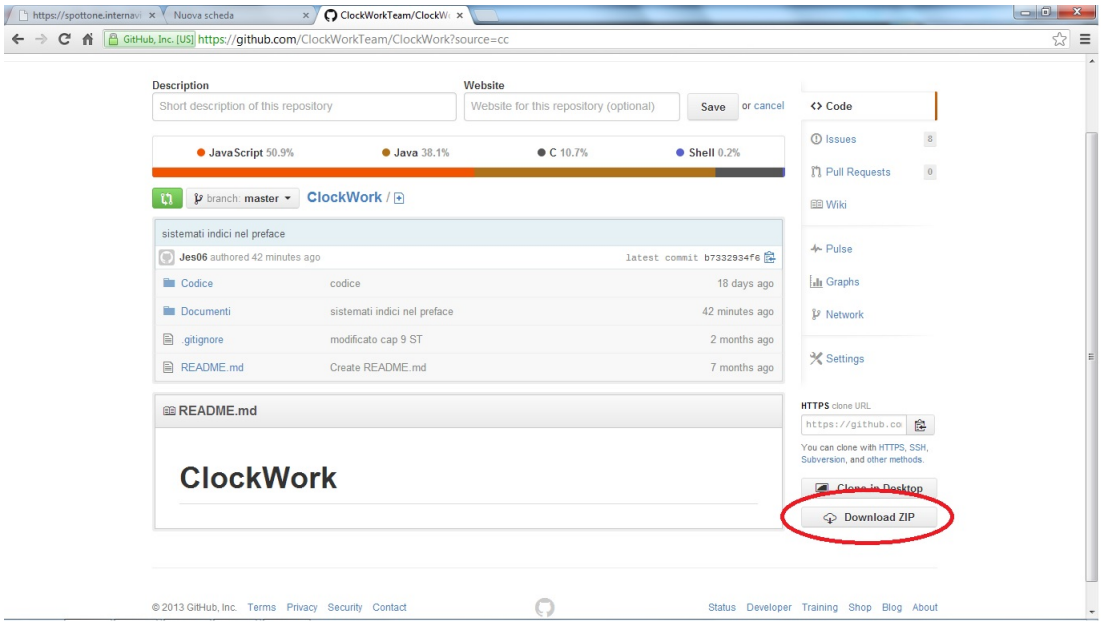

Figura 1: Download versione compressa

2. Alternativamente per poter utilizzare il repository in locale è necessario installare la versione del client git per il proprio sistema operativo[1]. In questa maniera è possibile comunicare e tenere aggiornata la propria copia con quella degli altri sviluppatori del progetto

3. A questo punto è necessario inserire, una volta avviato il client, i propri dati di registrazione a Github, inoltre è necessario inserire il percorso in cui posizionare la propria cartella locale del repository (Figure 2 e 3)

[1] Github for Windows (scaricabile da `http://windows.github.com/`), Github for Mac (scaricabile da `http://mac.github.com/`) o Git-Cola per Linux (scaricabile da `http://git-cola.github.io/` o dal software center se presente).

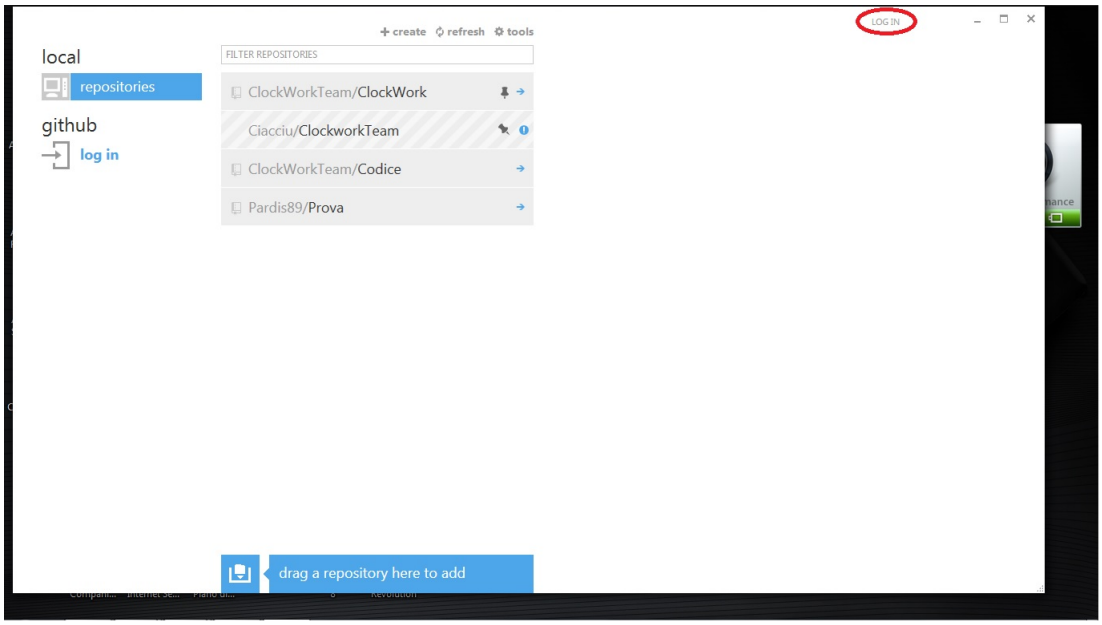

Figura 2: Accesso al client Github

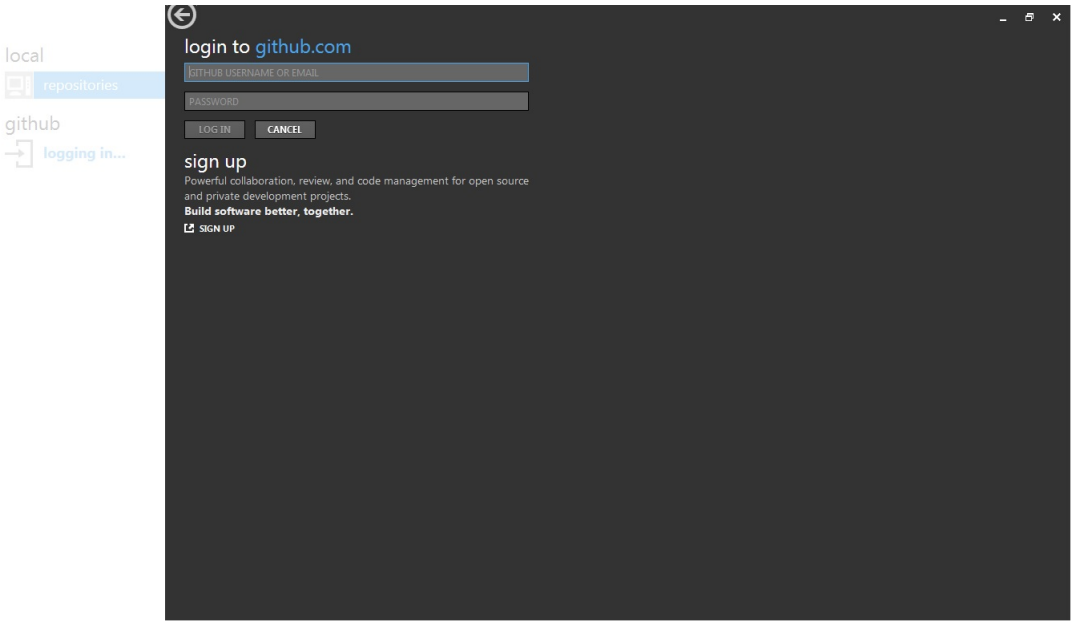

Figura 3: Login al client Github

4. Una volta effettuata l'autenticazione ed essere stati abilitati all'utilizzo del repository quest'ultimo sarà presente nella lista dei repository utilizzabili. A questo punto bisogna evidenziare il repository interessato e selezionare il comando *clone*. La clonazione del repository avviene automaticamente nella cartella locale scelta durante l'installazione (Figura 4)

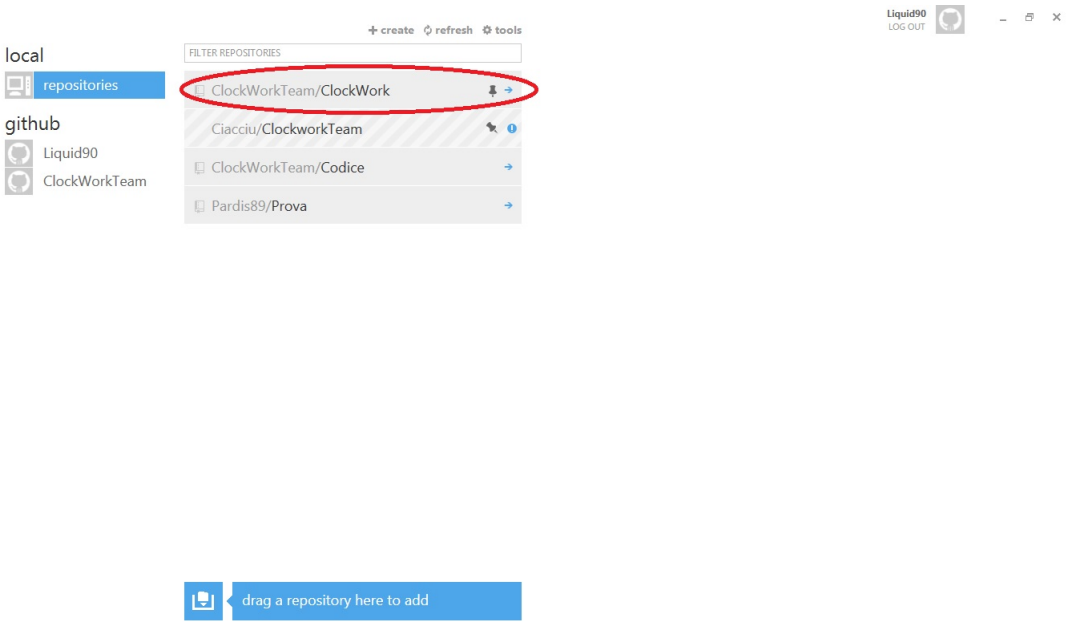

Figura 4: Clonazione del repository

5. A questo punto il repository locale è sincronizzato con quello online, è possibile accedere ai file del repository localmente (Figura 5)

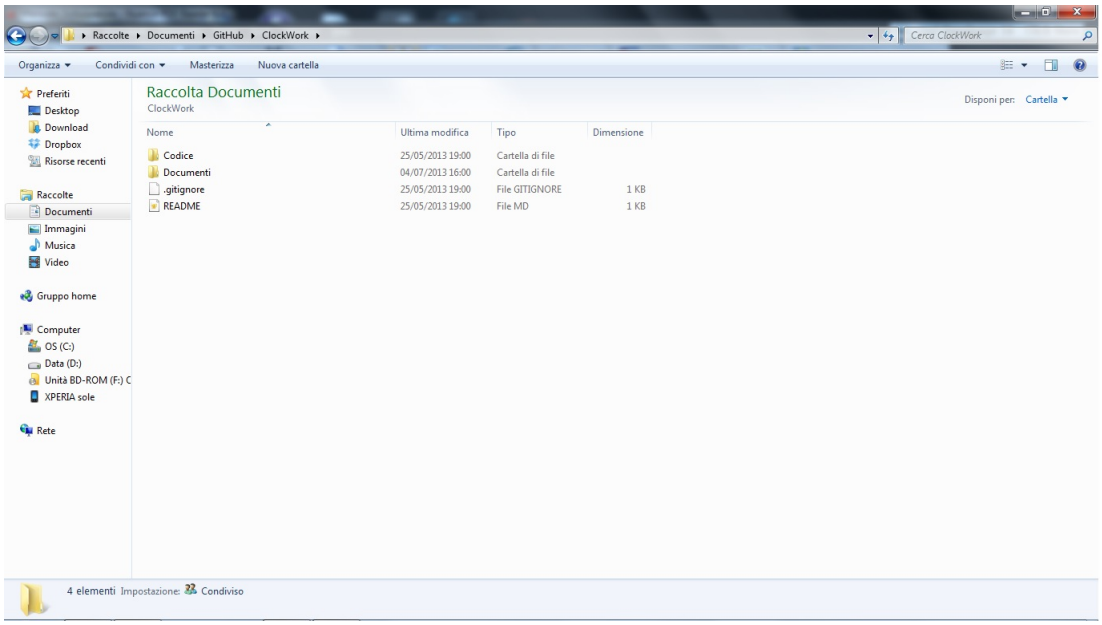

Figura 5: File in locale

3 Configurazione Eclipse

Il gruppo *Clockwork* consiglia l'utilizzo di IDE Eclipse per lavorare su **MyTalk**. Di seguito viene riportata una breve guida alla configurazione di questo strumento per lavorare al prodotto

3.1 Preparazione Eclipse

1. Scaricare Eclipse dal sito ufficiale[2]. Si consiglia la versione per Java Developers[3]
2. Nel caso non sia presente nella versione installata, bisogna scaricare EGit[4] per gestire il repository
3. Nel caso non sia presente nella versione installata, bisogna scaricare Junit[5] per gestire i test

3.2 Importazione del progetto in Eclipse

Per poter lavorare in Eclipse bisogna importare il progetto dal repository:

1. Andare su File -> Import (Figura 6)

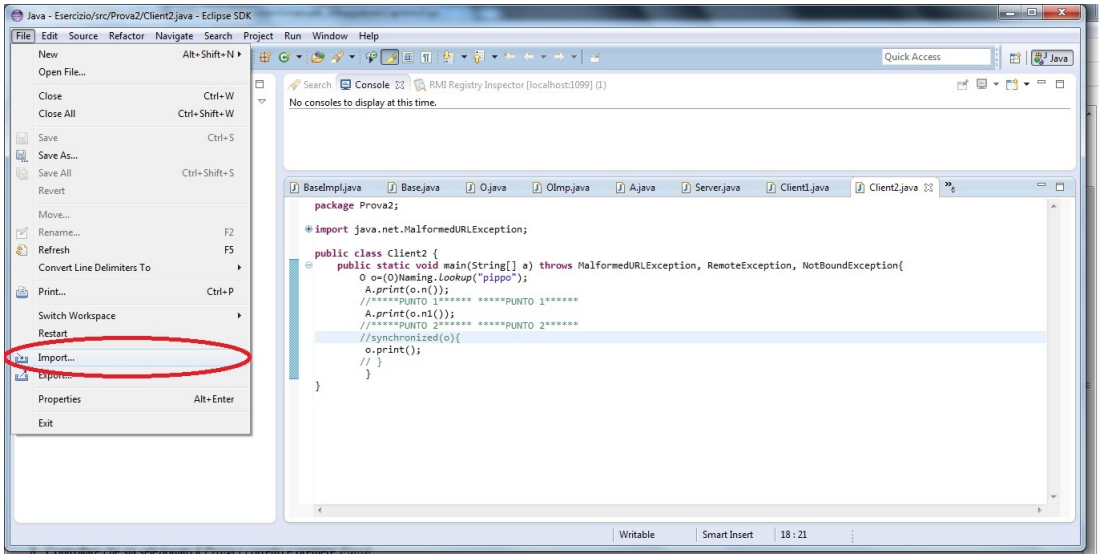

Figura 6: Selezione import da eclipse

[2] `http://www.eclipse.org/downloads/`
[3] Questa decisione è stata presa poiché la base di sviluppo del prodotto in questione è Java.
[4] `http://www.eclipse.org/egit/`
[5] `http://github.com/KentBeck/junit/downloads`

2. Nella finestra *Select* selezionare Git -> Projects from Git[6]. Premere *Next* (Figura 7)

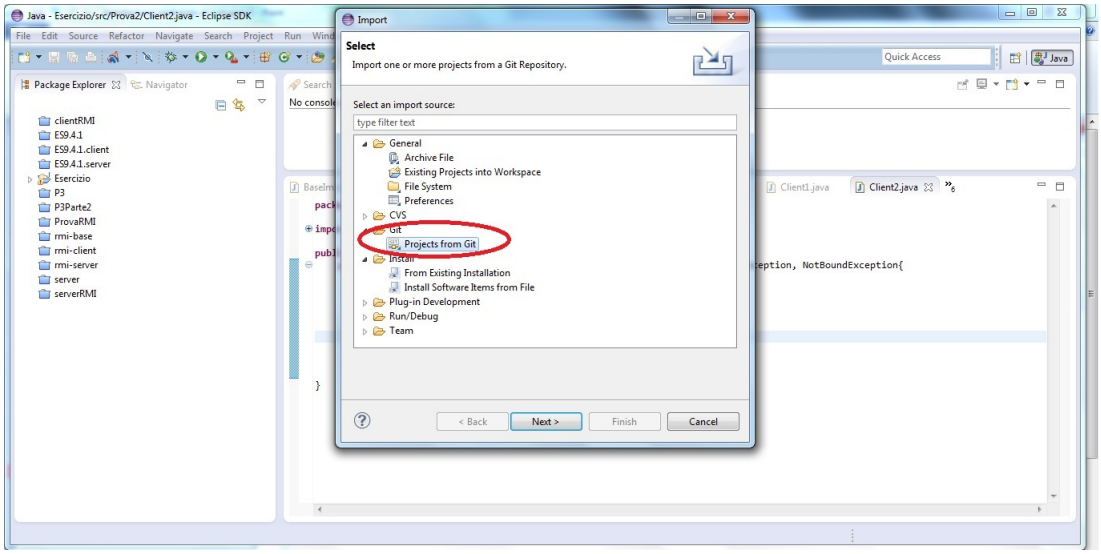

Figura 7: Selezione progetti da Git

3. Nella finestra *Select Repository Source* selezionare URI. Premere *Next* (Figura 8)

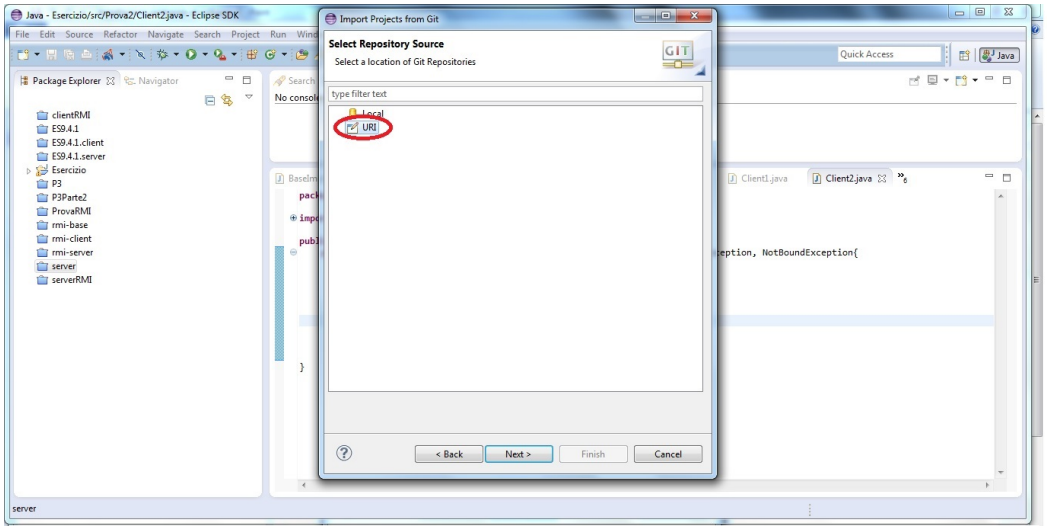

Figura 8: Selezione opzione URI

[6] In caso di mancanza di tale opzione assicurarsi che sia installato correttamente EGit.

4. Nella finestra *Select Git Repository* inserire nel campo URI il link necessario per effettuare lo scaricamento del progetto. I campi necessari una volta inserito l'URI si autocompleteranno, esclusa la password che bisogna inserire manualmente. Premere *Next* (Figura 9)

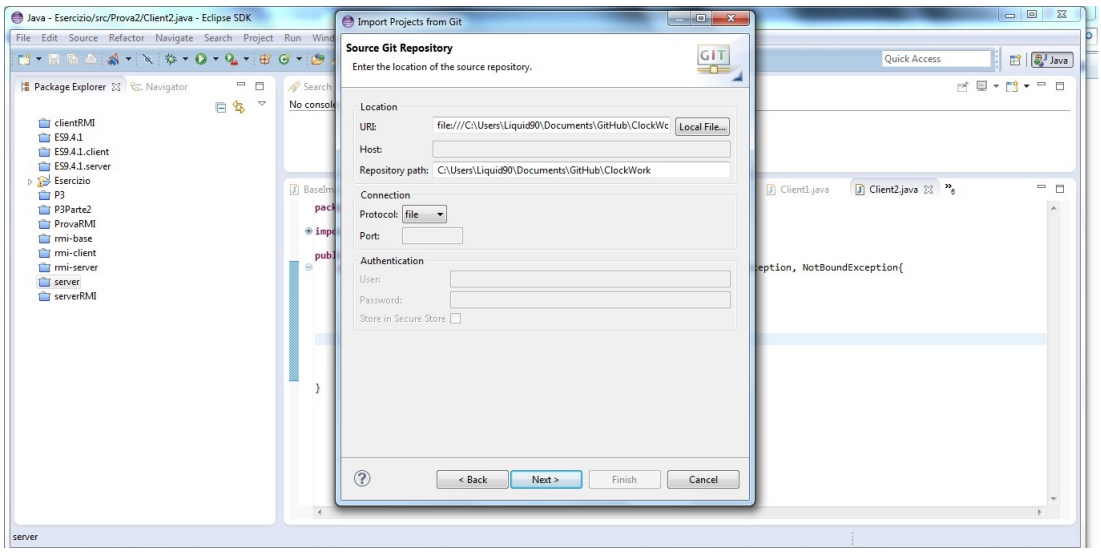

Figura 9: Selezione URI contenente progetto

5. Nella finestra *Branch Selection* selezionare *master*. Premere *Next* (Figura 10)

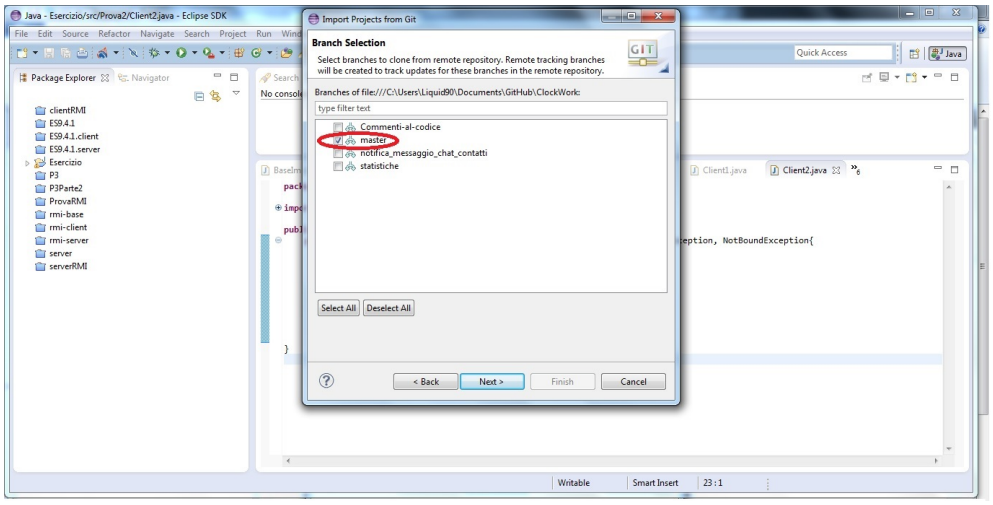

Figura 10: Selezione del branch

6. Nella finestra *Local Destination* premere il bottone *Browse* e selezionare la cartella dove risiede il vostro localhost come cartella di destinazione. Premere *Next* (Figura 11)

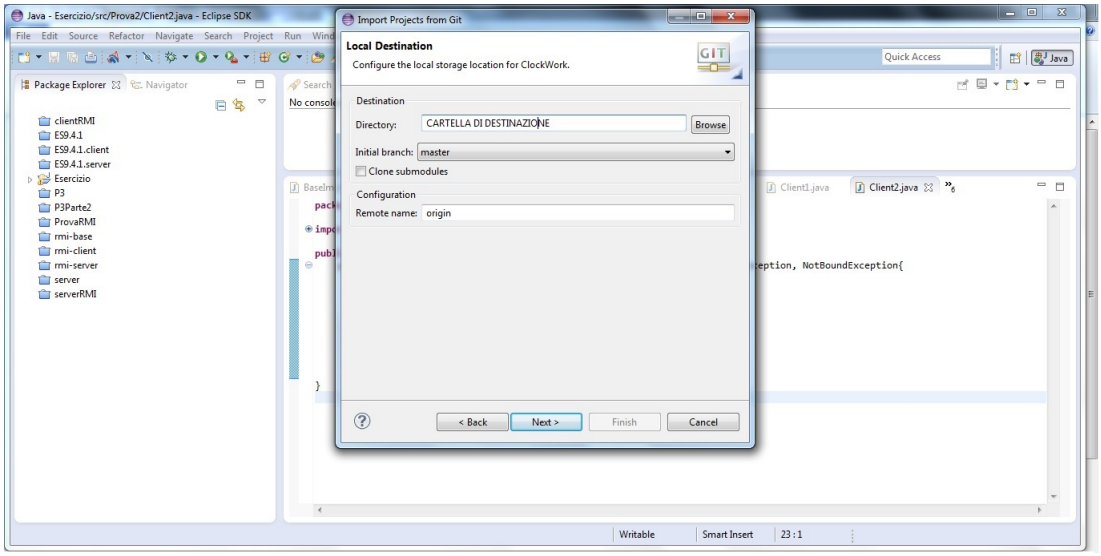

Figura 11: Selezione cartella di destinazione del progetto

7. Selezionare poi nella finestra *Select a wizard to use for importing project* l'opzione *Import existing project*. Premere *Next* (Figura 12)

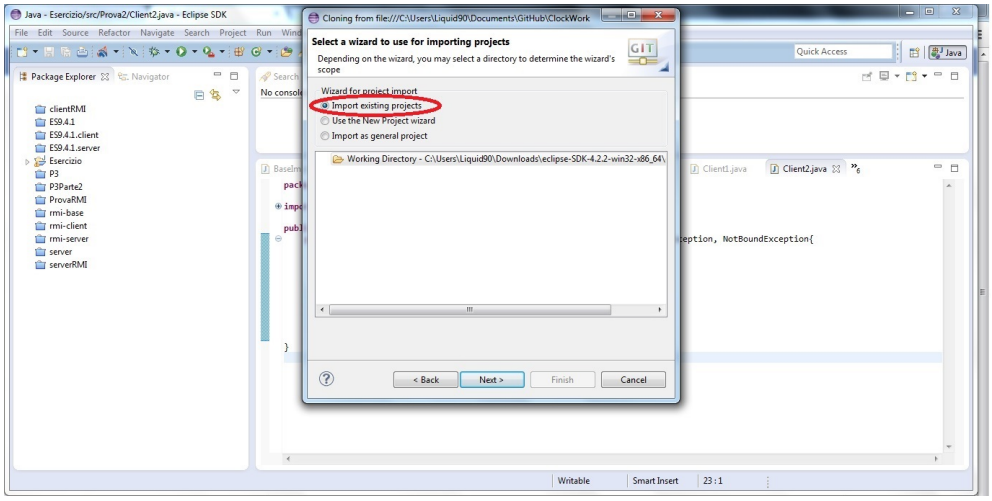

Figura 12: Selezione progetto da importare

8. Controllare che sia selezionato il *Project* corretto e premere *Finish* (Figura 13)

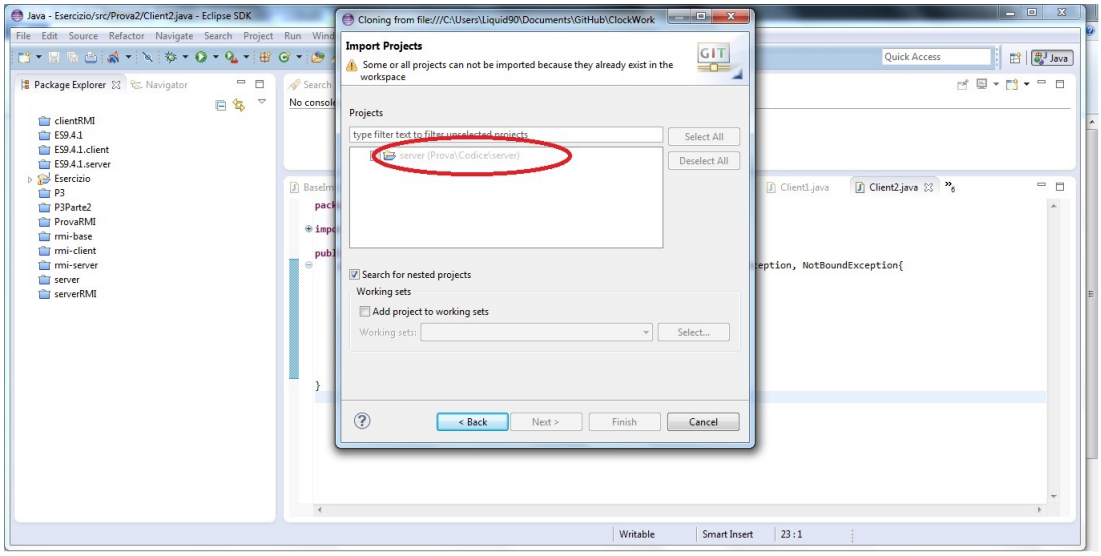

Figura 13: Verifica finale

9. Il vostro progetto è importato in Eclipse (Figura 14). Premendo su *run* si avvierà il server

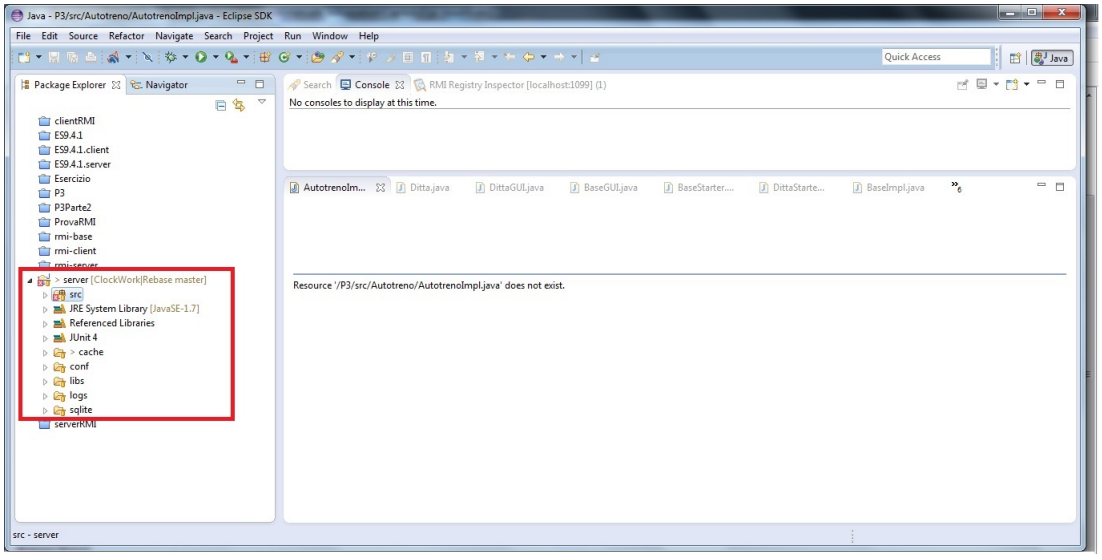

Figura 14: Progetto importato

4 Esecuzione MyTalk

Per poter eseguire **MyTalk** è necessario

1. Possedere un server apache nel quale andrà inserito il progetto
2. Eseguire la parte server tramite Eclipse

5 Guida allo sviluppo

Il gruppo *Clockwork*, in fase di progettazione, ha pensato di rendere il prodotto **MyTalk** facilmente estendibile in merito a:

- Database
- Servizi di comunicazione

5.1 Database

Quando il progetto **MyTalk** è iniziato, era richiesto un database leggero, di conseguenza è stato utilizzato SQLite. Si è deciso dunque di implementare nel package `mytalk.server.dao` la classe `JavaConnectionSQLite`, in maniera da accedere al database da una sola classe, tuttavia nel caso si volesse adottare un database diverso da SQLite basterà sostituire quest'ultima classe con un'altra classe che fornisca il collegamento al nuovo database scelto

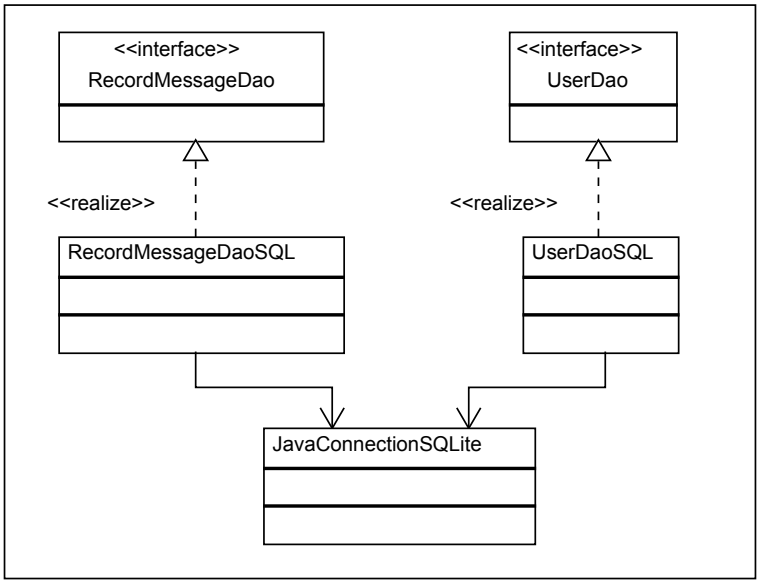

Figura 15: Estensione database

5.2 Servizi di comunicazione

Durante la fase di progettazione si è deciso di creare per ogni funzione prevista da **MyTalk** una classe nel package `mytalk.client.view` che si occupa di gestire a livello visivo l'operazione, una classe nel package `mytalk.client.communication` che si occupa di gestire la comunicazione tra client e server, ed una classe nel

package `mytalk.server.transfer` che si occupa di gestire la richiesta del client a cui invia una risposta. Se si vuole aggiungere una nuova funzionalità comunicativa al progetto **MyTalk** basterà aggiungere una classe per ogni package precedentemente elencato. Le figure 16 e 17 illustrano come aggiungere nuove funzionalità al progetto **MyTalk**

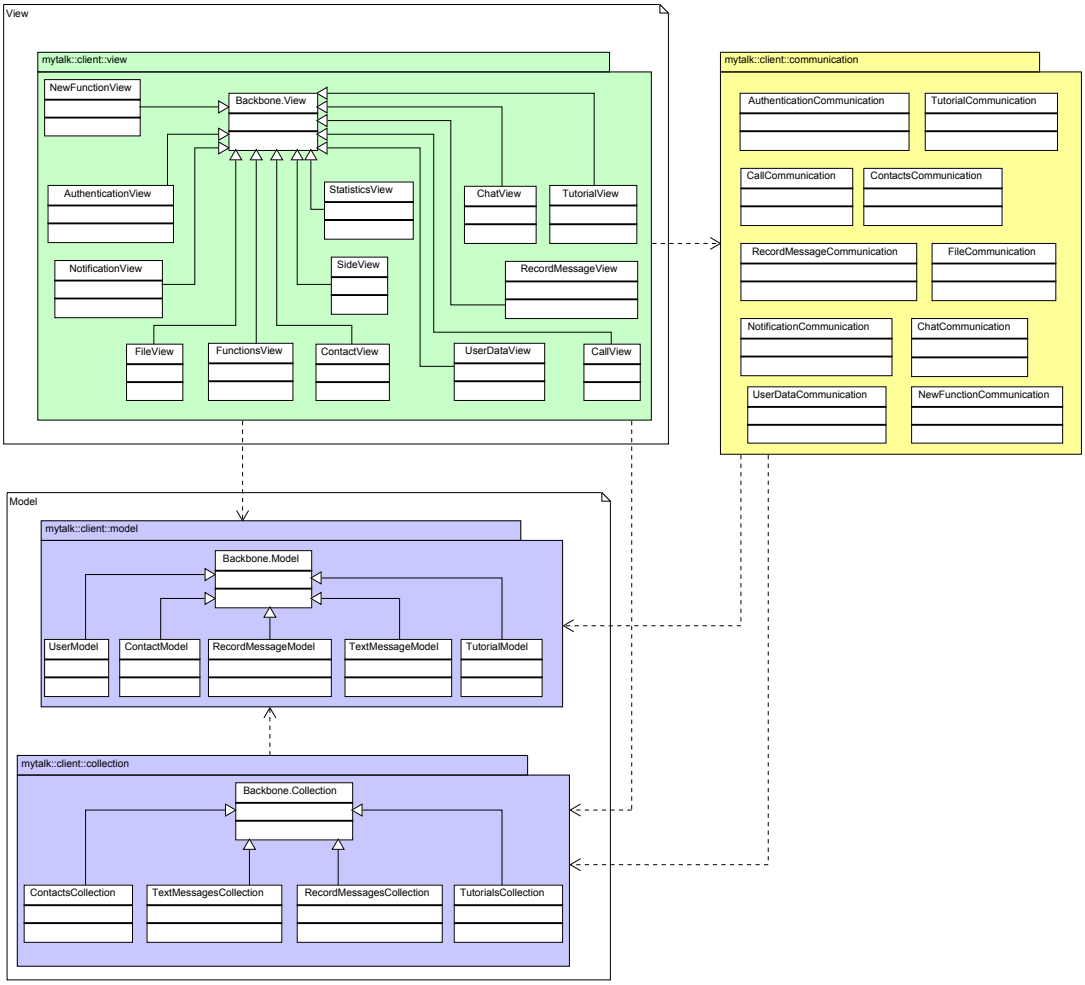

Figura 16: Aggiunta funzionalità al client

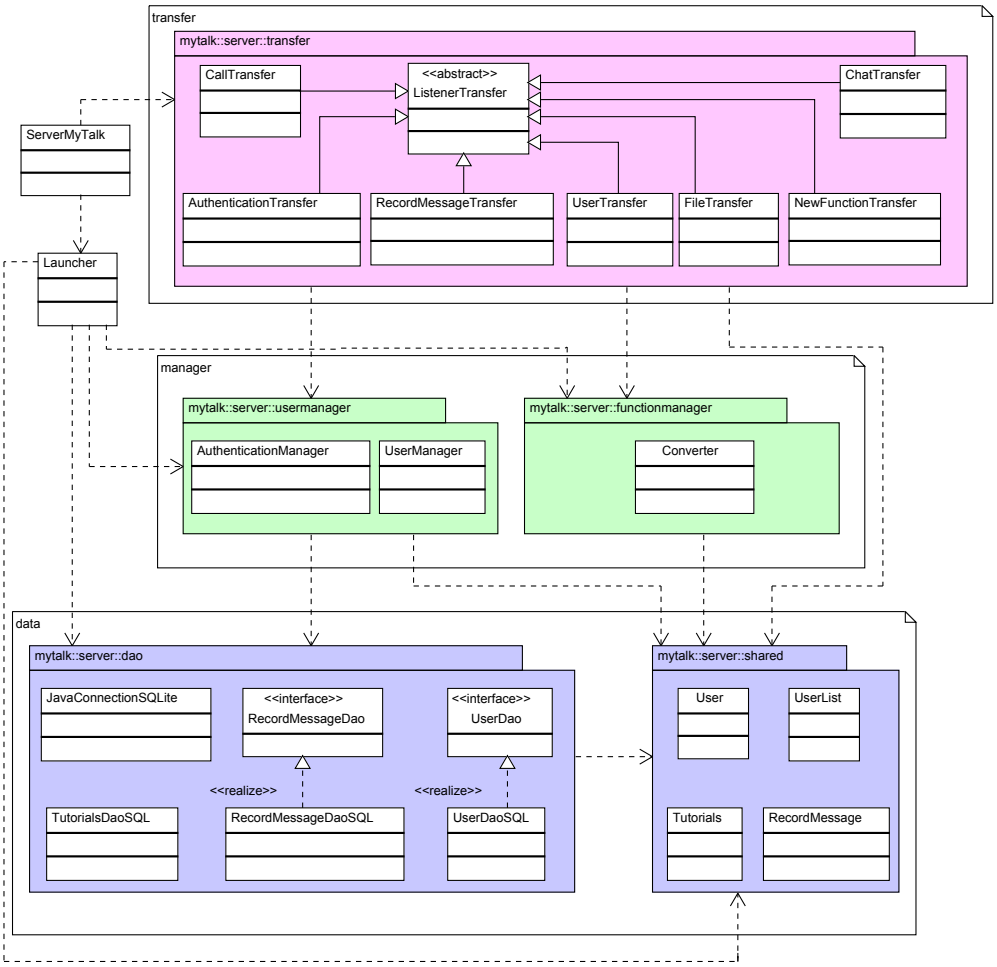

Figura 17: Aggiunta funzionalità al server

5.2.1 Istruzioni per la creazione di una classe in `mytalk.client.view`

Per poter realizzare una nuova classe nel package `mytalk.client.view` si deve:

1. Indicare le classi che andremo ad utilizzare per la creazione della nuova vista. Tra queste bisognerà sempre inserire nell'elenco jquery, underscore e backbone.

```
define(['jquery',
'underscore',
'backbone',
'communication/NewFunctionCommunication' ],
function($, _, Backbone, SideView, NewFunctionCommunication){
```

2. Estendere la nuova classe da Backbone.View

```
var NewFunctionView = Backbone.View.extend(
```

3. Scrivere il codice della nuova vista

4. Inserire se necessario, nelle viste appropriate, il punto in cui la nuova vista viene usata

5.2.2 Istruzioni per la creazione di una classe in `mytalk.client.communication`

Per poter realizzare una nuova classe nel package `mytalk.client.communication` si deve:

1. Inserire subito la seguente linea di codice:

```
define(['connection'], function(Connection){
```

Questo permette la comunicazione con il server con il seguente comando

```
Connection.send(Messaggio da inviare);
```

2. Se necessario creare variabili visibili all'interno di tutto il codice della classe, farlo appena eseguito il punto 1

3. Inserire il codice della nuova classe communication all'interno di un'istruzione return

```
return{
        funzioni della nuova communication
}
```

5.2.3 Istruzioni per la creazione di una classe in `mytalk.server.transfer`

Per poter realizzare una nuova classe nel package `mytalk.server.transfer` si deve:

1. Importare le seguenti librerie

```
import org.jwebsocket.api.WebSocketPacket;
import org.jwebsocket.kit.RawPacket;
import org.jwebsocket.kit.WebSocketServerEvent;
import org.jwebsocket.listener.WebSocketServerTokenEvent;
import org.jwebsocket.token.Token;
```

2. Creare una nuova classe che estende da ListenerTransfer

```
public class NewFunctionTransfer extends ListenerTransfer
```

3. Scrivere il codice della nuova classe