*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Addigy

Tunneler and Pubsub

Team # 01

**Team Members**

Carlos Ruiz

David Romero

**Product Owner**: Jason Dettbarn

**Instructor**: Masoud Sadjadi

All Addigy specific logos and slogan material is copyright Addigy, Inc. All other materials developed for this project are available open source under MIT License:

<http://opensource.org/licenses/MIT>

***Abstract***

*Tunneling*

In today's computer world it is practically impossible to create a straight connection from one computer to the other in order to communicate, or transfer a file, or even doing remote control. This study describes and evaluates a solution for this problem working behind firewalls, routers, and any security system preventing this connections among computers to be created.

This system will use a third party (the server) in order to establish a connection between two registered systems. This dedicated server will be in charge of redirecting the communication using TCP connections and a personalized protocol build on top of it. Both of the machines will also have a service running which will be in charge of communicating with the server.

*PubSub*

Communication between devices is a centerpiece of modern day computing. As such, there are many different ways in which devices can establish this communication, however how do we approach this in a way where we keep it scalable and simple to use. After much research, it was determined that a publish/subscribe model would be used.

This software will use RabbitMQ as the server that handles the creation and dispatching of messages. The software itself will simple facilitate the communication to said server,

abstracting the process in addition to adding new features such as presence and an idle timeout. The process is scaling off the existing APIs.

**Table of Contents**

**Introduction**

Current System

* Tunneler
* PubSub

Purpose of New System

- Tunneler

- PubSub

U**ser Stories**

Implemented User Stories

Pending User Stories

**Project Plan**

Hardware and Software Resources

Sprints Plan

*Sprint 1*

*Sprint 2*

*Sprint 3*

*Sprint 4*

*Sprint 5*

*Sprint 6*

*sprint 7*

**System Design**

Architectural Patterns

System and Subsystem Decomposition

Deployment Diagram

Design Patterns

**System Validation**

**Glossary**

**Appendix**

Appendix A - UML Diagrams

*Static UML Diagrams*

*Dynamic UML Diagrams*

Appendix B - User Interface Design

Appendix C - Sprint Review Reports

Appendix D - Sprint Retrospective Reports

**References**

**Introduction**

*Tunneler:*

The need and desired of users to access their personal computers remotely is an issue we face in today's market and there are limited resources to accomplish this goal and even more if is intended to be personalized.

The following sections provide characteristics and limitations of the current system as well as a description of the proposed system.

## Current System

*Tunneler:*

Being able to communicate with another computer or computers is a very desirable piece of feature in today's market. One service which enable us to achieve this goal are called Proxy servers. Proxy servers are computer systems or applications that acts as an intermediary between two endpoints (client and other servers). Client connects to the proxy server requesting some action and the proxy server redirects this request to the intended target.

There are three types of proxy servers but our interest are based on gateway or sometimes called tunneling proxy. Gateways are proxy servers that passes responses and requests unmodified from one point to the other.

*PubSub:*

The notion of having a client subscribe to an event to get live updates is an idea that see applications in many areas of technology. This ranges from messaging to issuing tasks to another machine. Many services exist, one such service is PubNub. PubNub allows for low latency streaming of content, allowing for easy implementation with over 20 SDK libraries available for use.

In order to communicate, users must utilize a publisher and subscriber key in order for producers to publish to a consumer via TCP connection. However, in order to utilize the service a user must pay for the service.

## Purpose of New System

*Tunneler:*

Our implementation will follow the same idea of GateWays. It will have a personalized server that will redirect all the connections and information to the corresponding tunnel (main server). There will be two softwares (personalized relaying servers) installed on both ends in order to start up a communication with the server and both systems. This will all be done using TCP connections and socket communication.

The new system will not need the interaction of the user to start up each program to establish a connection between the computers every time. The end machine which will receive the commands will have a “Tunneler” running. This so called tunneler will create and maintain an open tcp connection with the server (main server). Each tunnel connection will have a unique ID which will be stored into the client’s account and therefore accessible by the client.

On the other end the client will also have a software which initialize the connection to the server and it will tell the server who this client is and with which tunnel it wants to communicate.

*PubSub:*

The ultimate goal of our PubSub service is to achieve similar functionalities to that of PubNub, but tailored to Addigy’s needs. It will redirect tasks specified by the user to specified id of the request. Our system will utilize an AMQP (Advanced Messaging Queueing Protocol) by means of RabbitMq. The request made by the client will be sent to a queue that is awaiting to be requested by any worker that has a matching id.

The service will allow a user to input any command they wish to send to a terminal or command line and assume control of the worker terminal from a client. Utilizing the queueing system allows users to submit multiple tasks and service more requests.

**User Stories**

User stories are specifications of one or more sentences. They are part of agile software development methodologies that express what a user does or needs to do as part of his/her job function. User story is the basis for determining the functions a business system must provide, and to facilitate requirements management. Also captures the ‘who’, ‘what’ and ‘why’ of a requirement in a simple, concise way. The following sections consists of a summary of the user stories implemented for this project as well as the pending user stories that will be considered for the future development of the project.

## Implemented User Stories

User **Story #520 - Creating tunneler proof of concepts**

As a user, I would like to open up my browser, connect to a web page and remotely control my computer from anywhere. This will be useful for companies when having multiples computers in one department and having to perform any task on all of them them.

**Acceptance Criteria:**

* If am a user I can connect to a website and retrieve information from my computer
* The server (system) will handle my connection and redirected to my remote computer
* Remote computer will then get local information and send back to server and server back to the browser the user used

**User Story #534 - Editing documentation for tunneler and research technology**

As a developer, I will be editing and adding information to the documentation of this project so then we can see the progress of the project and that way we will also have a guide of what path to follow.

User **Story #520 - Creating tunneler proof of concepts**

As a user, I would like to open up my browser, connect to a web page and remotely control my computer from anywhere. This will be useful for companies when having multiples computers in one department and having to perform any task on all of them them.

**User Story #517 - Creating PubSub Proof of Concept**

As a developer, I must familiarize myself with the existing technologies in order to create a better and more robust PubSub service. This entails research and the application of my finds before proceeding with a larger model.

**Acceptance Criteria:**

* The publisher must successfully publish an event
* The client must be able to subscribe to the event using the service
* The subscriber must see the subscribed event.

# #518 - Edit document with PubSub research and information for proof of concept - Research Technology

During research, I will fill out what I have found during my research, what I know and the diagrams that have been created in designing the proof of concept.

## 

## 

## 

## 

## #516 - Researching publish/subscript system

As a developer, it is important to full understand the technologies I will be using, in addition to understanding what the ultimate goal of my efforts is. Time will be set aside to research different means of producing an effective publish/subscribe service.

# [#540](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/540) - Implementing data stream

### Description:

* As a developer I have to maintain a communication between services at all time, data is stream back and forth therefore it does not follow a one direction communication. This will tunnel the data coming from one place to the other in the order that gets received.

**Acceptance Criteria:**

* Tunneler shall receive data from server and send to server as it appear and not in any order as usually expected
* Client shall receive data from server and send to server as it appear and not in any order as usually expected
* Server shall receive data from “Tunneler/Client” and send to “Tunneler/Client” as it appear and not in any order as usually expected

**#545 - allowing remote control over tunneler**

**Description:**

* As a developer I shall allow remote control communication to flow smoothly over the tunneler. This adds more features and functionality to what the tunneler can handle.

### Acceptance Criteria:

* Both machines should have internet connection
* Port shall be provided in order to connect

**#561 - Refactor Client/Tunneler programs**

### Description:

* As a developer programs shall be written in a scalable and understandable way for future changes. This helps in the clarity of the program and makes it easier for future changes.

### Acceptance Criteria:

* Program shall be clean and clear

**#562 - Fix communication issue among programs**

### Description:

* As a developer I must make sure program runs and works as expected in order to provide the expected services to the user.

### Acceptance Criteria:

* Programs must communicate using the same protocol
* Information must not be out of sync

**#564 - Fix communication when receiving data**

### Description:

* As a developer I shall ensure that the program works as expected in every way, providing a working system in every aspect.

### Acceptance Criteria:

* Programs must receive the entire data before trying to get next block of data

**#576 - Fix multiple connections to the same tunnel**

### Description:

* As a developer I shall build a system in which multiple clients may communicate with the same source in order to provide multiple service to the user at the same time.

### Acceptance Criteria:

* Multiple clients may connect to the same end system to get information from.
* One client may connect only to a single tunnel
* Tunnel may receive multiple connections from different clients

**#577 - Fix handling server/tunnel disconnecting**

### Description:

* As a developer I shall build a system able to handle disconnections from every source to give a better user experience and a stable server.

### Acceptance Criteria:

* Tunnel connection may be disconnected from server without causing errors
* Client connection may be disconnected from server without causing errors

**#577 - Refactor server code**

### Description:

* As a developer, it is a must to build a system that possesses efficiency and that it is highly scalable in order to provide a better user experience and to easily adapt to different necessities without loosing performance.

### Acceptance Criteria:

* Server will have 1000 open threads to handle communication at all times

**#594 - Fix handling server/tunnel disconnecting and creating project diagram**

### Description:

* As a developer, documentation and project design are essential steps in building and designing a software product in order to correctly manage developing time and getting the intended product.

### Acceptance Criteria:

* Clear project goals to achieve
* Clear illustration of project structure

**#595 -** **Manual System testing**

### Description:

* As a developer, testing is an important extension of a system development process i order to provide a solid, bug free product to the users.

### Acceptance Criteria:

* System must meet every criteria described on its specifications
* System must pass all test cases

**#611 -** **Create pubsub agent for computers**

### Description:

* As a developer, the creation of this new software will combine the use of this product with some other publish and subscribe type of services and it will facilitate the use of it for the regular user.

### Acceptance Criteria:

* Must accept Json containing essential information
* Must start programs depending on action
* Must create connection to service RabitMq and be able to receive and publish commands following protocol
* Must handle errors from started programs or services

**#614 -** **Setup FIU virtual box and update all setup files**

**Description:**

* As a developer, server must be clean and running with the latest code and with no errors. Maintenance is essential for testing purposes and use of the product.

### Acceptance Criteria:

* shall be always running and Server running
* Handle errors
* Shall always be accessible

**#613 -** **Update documentation with latest information**

### Description:

* As a developer, documentation must always be updated as it plays an essential role to the development process, it facilitates the path to follow when developing

### Acceptance Criteria:

* Shall be clean and understandable

**#625 - Fix/Modify pubsub agent**

### Description:

* As a developer it’s a goal to make a efficient and simple to use product so every user feels comfortable while using it. Modification to this pubsub agent will now also start each separate software and it will launch the desired application to connect to.

### Acceptance Criteria:

* Agent shall receive command and execute it as presented
* Agent shall launch desired application to connect from without user interaction
* Agent shall be able to start/terminate each software without user interaction

**#628 - Manual testing for system connection**

**Description:**

* As a developer it is necessary to test the product before is available for the user as the goal is for the product to be used without errors.

### Acceptance Criteria:

* All software shall start and terminate without problems
* Communication between services shall not be corrupted or interrupted
* Product shall be easy to install without any errors
* All terminated connections shall be handled without server failure

**#627 - Modify client/tunneler code**

### Description:

* As a developer modifying and updating code is necessary in order to improve performance, readability and efficiency of the product. Client and tunnel now shall identify if an instance of the same program is running with same ids before trying to connect to the server.

Acceptance Criteria:

* Client/Tunnel shall check if is already running with same information
* Client/Tunnel shall respond success message if instance successfully started

**#626 - Update project documentation**

### Description:

* As a developer documentation and project organization is a must in order to develop and keep track of all the progress achieved.

### Acceptance Criteria:

* Project deliverable must be completed and up to date
* All code must be updated on Github

# Project Plan

This project will follow an agile development approach to ensure that the project is constantly being updated and goals are being met for deployment. Different tools will be used to follow this management structure and planning of the project.

## Hardware and Software Resources

*Tunneler:*

Resources that will be used for this project are the following:

**Python:**

Will implement the Client’s program that connects to the server. ‘Server’ will also be implemented using python as well as the ‘Tunneler’ program which will be installed on the end computer which will be remotely controlled.

**Mac/Linux for development:**

The team desired for this project to use a fast Mac/Linux development environment because of the advantages that offers. Easy installation of new tools using (apt-get or homebrew). Easy configuration of python running systems and server hosting.

*PubSub:*

Resources that will be used for this project are the following:

**Python:**

Python will be used in the development of the server and client scripts that will interact with RabbitMq. The pika AMQP library will be used to facilitate the development of the two scripts.

**Windows/OS X/Linux for development:**

Mac/Linux environments provide easy environment set up through their respective package managers. In additions, it is crucial to focus on OS X due to the fact that this application is intended for Mac usage. Linux provides a similar environment and will mostly be used for servers. Windows is also used due to how readily available tools are for the platform.

## Sprints Plan

### Sprint 1

(8/31/2015 - 9/11/2015)

**User Story #520 - Creating tunneler proof of concepts**

As a user, I would like to open up my browser, connect to a web page and remotely control my computer from anywhere. This will be useful for companies when having multiples computers in one department and having to perform any task on all of them them.

***Tasks***

* 533 Design system
* 527 Create Server python program
* 526 Create ‘Tunneler’ program
* 535 Create Client’s program to communicate with server
* 524 Set up MongoDB (optional/if needed future)
* 523 Python environment
* 522 Github structure

***Acceptance Criteria***

* If am a user I can connect to a website and retrieve information from my computer
* The server (system) will handle my connection and redirected to my remote computer
* Remote computer will then get local information and send back to server and server back to the browser the user used

***Modeling***

* Appendix A : Figure #5.001

**User Story #534 - Editing documentation for tunneler and research technology**

As a developer, I will be editing and adding information to the documentation of this project so then we can see the progress of the project and that way we will also have a guide of what path to follow.

**User Story #517 - Creating PubSub Proof of Concept**

As a developer, I must familiarize myself with the existing technologies in order to create a better and more robust PubSub service. This entails research and the application of my finds before proceeding with a larger model.

***Tasks***

* 529 #517 - Create a publishing script
* 530 #517 - Create a subscribe script
* 528 #517 - Design
* 531 #517 - Environment Set-up

# User Story #518 - Edit document with PubSub research and information for proof of concept - Research Technology

During research, I will fill out what I have found during my research, what I know and the diagrams that have been created in designing the proof of concept.

## User Story #516 - Researching publish/subscript system

As a developer, it is important to full understand the technologies I will be using, in addition to understanding what the ultimate goal of my efforts is. Time will be set aside to research different means of producing an effective publish/subscribe service.

***Acceptance Criteria***

* The publisher must successfully publish an event
* The client must be able to subscribe to the event using the service
* The subscriber must see the subscribed event.

***Modeling***

* Appendix A : Figure #5.002
* Appendix A : Figure #5.003

***Sprint 2***

(9/12/2015 - 9/25/2015)

Tunneler:

# [#553](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/553) - Initializing client

### Description:

* As a user, I have to initialize the client service every time I wish to connect to a tunnel or would like to change connection type (Web or ssh). This will help the user with the understanding of a simple process and will simplify the complexity of the program.

**Acceptance Criteria:**

* User shall open command prompt or terminal
* User shall run python service and provide three parameters <desired tunnel id> <local port> <destination port>

***Modeling:***

* Appendix A : Figure #5.005

# [#540](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/540) - Implementing data stream

### Description:

* As a developer I have to maintain a communication between services at all time, data is stream back and forth therefore it does not follow a one direction communication. This will tunnel the data coming from one place to the other in the order that gets received.

**Acceptance Criteria:**

* Tunneler shall receive data from server and send to server as it appear and not in any order as usually expected
* Client shall receive data from server and send to server as it appear and not in any order as usually expected
* Server shall receive data from “Tunneler/Client” and send to “Tunneler/Client” as it appear and not in any order as usually expected

**Tasks:**

* 544 #540 - Back and Forth communication among services
* 543 #540 - Tunneler communication with ssh service at specified port #
* 542 #540 - Server communication with client and tunneler
* 541 #540 - Client initialization for web and ssh requirements

***Modeling:***

* Appendix A : Figure #5.006 Implementing data stream (Initial communication setup)
* Appendix A : Figure #5.007 Implementing data stream (Established communication)

# [#](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/553)547 - Create an SDK for Javascript

### Description:

* As a developer, I understand the importance of a good library in order to get a job done. A publish subscribe Javascript SDK will begin development.

### Acceptance Criteria:

1. Basic features functional
2. Implementable
3. Easy to use

**Tasks:**

* #547 - Essential methods (subscribe, publish, initialize)

# 

# [#5](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/540)56 - Begin Research into Javascript

### Description:

* As a developer, I must acquainted myself with the language I am using, in this case Javascript. I will research how to develop a good library, in addition for which avenues are best for taking on the task of developing a publish subscribe library.

### Sprint 3

(9/28/2015 - 10/9/2015)

**#545 - allowing remote control over tunneler**

**Description:**

* As a developer I shall allow remote control communication to flow smoothly over the tunneler. This adds more features and functionality to what the tunneler can handle.

### Acceptance Criteria:

* Both machines should have internet connection
* Port shall be provided in order to connect

**Tasks:**

* #560 - Maintain communication open
* #559 - Receive port and open connection to local port

**#561 - Refactor Client/Tunneler programs**

### Description:

* As a developer programs shall be written in a scalable and understandable way for future changes. This helps in the clarity of the program and makes it easier for future changes.

### Acceptance Criteria:

* Program shall be clean and clear

**Tasks:**

* #572 - Refactor tunneler program
* #571 - Refactor client program

**#562 - Fix communication issue among programs**

### Description:

* As a developer I must make sure program runs and works as expected in order to provide the expected services to the user.

### Acceptance Criteria:

* Programs must communicate using the same protocol
* Information must not be out of sync

**Tasks:**

* #[567](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/567) - [Fix communication protocol on tunneler](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/567)
* #[566](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/566) - [Fix communication protocol on server](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/566)
* #[565](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/565) - [Fix communication protocol on client](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/565)

**#564 - Fix communication when receiving data**

### Description:

* As a developer I shall ensure that the program works as expected in every way, providing a working system in every aspect.

### Acceptance Criteria:

* Programs must receive the entire data before trying to get next block of data

**Tasks:**

* #570 - [Fix communication protocol on tunneler](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/567)
* #569 - [Fix communication protocol on server](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/566)
* #568 - [Fix communication protocol on client](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/565)

**#558 - Begin implementation of presence**

### Description:

* The beginnings of presence (that is who’s connected, are they connected and so forth) will be implemented. This will be based on research findings

**#546 - Create an SDK for Python**

### Description:

* As a developer, an easy to use and understand library is very important. I will begin transforming the work previously done in Javascript to Python. The library will be able to publish and consume messages using RabbitMQ as a broker

**#574 - Have both clients communicate with one another, using the libraries**

### Description:

* In order to demonstrate how versatile the library is, the two clients will communicate with another. In the real world, there are many uses for being able to communicate across platforms like this.

### Sprint 4

(9/12/2015 - 10/23/2015)

**#576 - Fix multiple connections to the same tunnel**

### Description:

* As a developer I shall build a system in which multiple clients may communicate with the same source in order to provide multiple service to the user at the same time.

### Acceptance Criteria:

* Multiple clients may connect to the same end system to get information from.
* One client may connect only to a single tunnel
* Tunnel may receive multiple connections from different clients

**Tasks:**

* #582 - Make tunnel handle multiple connections keeping their reference
* #581 - Make clients able to connect to a single tunnel

**#577 - Refactor server code**

### Description:

* As a developer, it is a must to build a system that possesses efficiency and that it is highly scalable in order to provide a better user experience and to easily adapt to different necessities without loosing performance.

### Acceptance Criteria:

* Server will have 1000 open threads to handle communication at all times

**Tasks:**

* #580 - [Spawn 1000 threads to handle communication](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/580)
* #579 - [Create three threads to handle connections](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/579)

**#577 - Fix handling server/tunnel disconnecting**

### Description:

* As a developer I shall build a system able to handle disconnections from every source to give a better user experience and a stable server.

### Acceptance Criteria:

* Tunnel connection may be disconnected from server without causing errors
* Client connection may be disconnected from server without causing errors

**Tasks:**

* #583 - [fix handling pending connections when disconnecting](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/583)

**#586 - Begin the set-up of a heartbeat in Javascript**

### Description:

* In addition to monitoring who is entering the channel, our API should be able to ‘tell’ the broker how long it should wait for activity from the client before disconnecting. This should be entirely handled by the broker.

### Acceptance Criteria:

1. Should successful disconnect a user after the set amount of time

**#587 - Begin the set-up of a heartbeat in Python**

### Description:

* In addition to monitoring who is entering the channel, our API should be able to ‘tell’ the broker how long it should wait for activity from the client before disconnecting. This should be entirely handled by the broker.

### Acceptance Criteria:

1. Should successful disconnect a user after the set amount of time

**#585 - Incorporate Presence in Javascript**

**Description:**

* A major requirement for our API is to monitor who is entering and exiting a channel. Initial work, including investigation on how to go about it will be commenced. All this monitoring should be handled on the brokers end, not the client.
* UPDATE: Will be using the presence exchange plugin in Rabbitmq

### Acceptance Criteria:

1. Should successful monitor who enters the channel
2. Should successful monitor who exits the channel

**#584 - Incorporate Presence in Python**

**Description:**

* A major requirement for our API is to monitor who is entering and exiting a channel. Initial work, including investigation on how to go about it will be commenced. All this monitoring should be handled on the brokers end, not the client.
* UPDATE: Will be using the presence exchange plugin in Rabbitmq

### Acceptance Criteria:

1. Should successful monitor who enters the channel
2. Should successful monitor who exits the channel

**#588 - Refactor the PubSub code**

### Description:

* Due to all the newly introduced presence code, some refactoring must occur in order to keep the code legible

### Sprint 5

(10/23/2015 - 11/06/2015)

**#594 - Fix handling server/tunnel disconnecting and creating project diagram**

### Description:

* As a developer, documentation and project design are essential steps in building and designing a software product in order to correctly manage developing time and getting the intended product.

### Acceptance Criteria:

* Clear project goals to achieve
* Clear illustration of project structure

**Tasks:**

* #593 - [Creating high-level project diagram](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/593)
* #592 - [Set up all readme files](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/592)
* #591 - [Update all github code](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/591)
* #590 - [Fix client design](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/590)
* #589 - [Fix tunneler design](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/589)

***Modeling:***

* Appendix A : Figure #5.008 - High level program diagram
* Appendix A : Figure #5.009 - Detailed High level program diagram
* Figure #5.021 - Tunnel start up
* Figure #5.022 - Client start up And sending request
* Figure #5.023 - Tunnel receiving request
* Figure #5.024 - Client receiving request

**#595 -** **Manual System testing**

### Description:

* As a developer, testing is an important extension of a system development process i order to provide a solid, bug free product to the users.

### Acceptance Criteria:

* System must meet every criteria described on its specifications
* System must pass all test cases

**Tasks:**

* #606 - [Lost of connection handling](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/606)
* #605 - [Communication between programs](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/605)
* #604 - [Client start up test](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/604)

PubSub:

**#586 - Begin the set-up of a heartbeat in Javascript**

### Description:

* In addition to monitoring who is entering the channel, our API should be able to ‘tell’ the broker how long it should wait for activity from the client before disconnecting. This should be entirely handled by the broker.

### Acceptance Criteria:

1. Should successful disconnect a user after the set amount of time

**#587 - Begin the set-up of a heartbeat in Python**

### Description:

* In addition to monitoring who is entering the channel, our API should be able to ‘tell’ the broker how long it should wait for activity from the client before disconnecting. This should be entirely handled by the broker.

### Acceptance Criteria:

1. Should successful disconnect a user after the set amount of time

**#602 - Create high level documentation and fix existing documentation for PubSub**

### Description:

* As developers, we need to be ready to convey our idea to peers from other areas of expertise. As such, diagrams and other means of documentation are crucial to the development progress as they help us explain our design to others. PubSub’s overall design will be represented in an easy to use diagram. More so, the file structure of the project will be re-arranged

**Tasks:**

* #603 - Update all README files
* #600 - High level design of PubSub
* #599 - Restructure file layout for PubSub

**#598 - Manual Testing of PubSub**

### Description:

* As a developer, it is good to ensure that the core features are up and running before incorporating another layer on top of my software. As such, more use cases will be tested.

### Sprint 6

(11/09/2015 - 11/20/2015)

**#611 -** **Create pubsub agent for computers**

### Description:

* As a developer, the creation of this new software will combine the use of this product with some other publish and subscribe type of services and it will facilitate the use of it for the regular user.

### Acceptance Criteria:

* Must accept Json containing essential information
* Must start programs depending on action
* Must create connection to service RabitMq and be able to receive and publish commands following protocol
* Must handle errors from started

**Tasks:**

* #616 - [Start development process](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/616)
* #615 - [Designing structure](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/615)

***Modeling:***

* Appendix A : Figure #5.021 - Pubsub running

**#614 -** **Setup FIU virtual box and update all setup files**

### Description:

* As a developer, server must be clean and running with the latest code and with no errors. Maintenance is essential for testing purposes and use of the product.

### Acceptance Criteria:

* shall be always running and Server running
* Handle errors
* Shall always be accessible

**Tasks:**

* #619 - [Update all readme and setup files](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/619)
* #618 - [Upload latest project](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/618)
* #617 - [Access FIU virtual box](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/617)

**#613 -** **Update documentation with latest information**

### Description:

* As a developer, documentation must always be updated as it plays an essential role to the development process, it facilitates the path to follow when developing

### Acceptance Criteria:

* Shall be clean and understandable

**Tasks:**

* #621 - [Create all project designs](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/621)
* #620 - [Update deliverable with latest information](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/620)

**#623 -** **Modify the libraries based on requirements that arise**

### Description:

* During the development of the application, new requirements may arise. As a result, such changes will be reflected in the pubsub class as necessary

**#624 -** **Research methods of implementation of heartbeat**

### Description:

* As a result of design shortcomings last sprint, a new approach will be developed. As a result, more research must be conducted in order to appropriately implement the feature

**#622 -** **Create a web interface for the application**

### Description:

* After finishing the back end components, a front end UI must now be developed.

### Sprint 7

(11/23/2015 - 12/04/2015)

**#625 - Fix/Modify pubsub agent**

### Description:

* As a developer it’s a goal to make a efficient and simple to use product so every user feels comfortable while using it. Modification to this pubsub agent will now also start each separate software and it will launch the desired application to connect to.

### Acceptance Criteria:

* Agent shall receive command and execute it as presented
* Agent shall launch desired application to connect from without user interaction
* Agent shall be able to start/terminate each software without user interaction

**Tasks:**

* #632 - [Test software and feature ssh/web/vnc](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/632)
* #631 - [Add feature to start application to connect to client software](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/631)
* #630 - [Add feature to terminate client/tunneler](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/630)
* #629 - [Add feature to start client/tunneler](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/629)

**#628 - Manual testing for system connection**

### Description:

* As a developer it is necessary to test the product before is available for the user as the goal is for the product to be used without errors.

### Acceptance Criteria:

* All software shall start and terminate without problems
* Communication between services shall not be corrupted or interrupted
* Product shall be easy to install without any errors
* All terminated connections shall be handled without server failure

**Tasks:**

* #635 - [Test complete software in action](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/635)
* #634 - [Test client/pubsub agent/tunneler setup file](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/634)
* #633 - [Test server installation](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/633)

**#627 - Modify client/tunneler code**

### Description:

* As a developer modifying and updating code is necessary in order to improve performance, readability and efficiency of the product. Client and tunnel now shall identify if an instance of the same program is running with same ids before trying to connect to the server.

### Acceptance Criteria:

* Client/Tunnel shall check if is already running with same information
* Client/Tunnel shall respond success message if instance successfully started

**Tasks:**

* #638 - [Check code and clean](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/638)
* #637 - [Add feature to tunneler to identify if already running](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/637)
* #636 - [Add feature to client to identify if already running](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/636)

**#626 - Update project documentation**

### Description:

* As a developer documentation and project organization is a must in order to develop and keep track of all the progress achieved.

### Acceptance Criteria:

* Project deliverable must be completed and up to date
* All code must be updated on Github

**Tasks:**

* #641 - [Update project deliverable](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/641)
* #640 - [Upload latest code to Github](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/640)
* #639 - [Update read-me files](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/639)

**#642 - Manual Testing of Web Application**

### Description:

* In order to assure the highest quality, testing should be done on the web application’s view. As such, due to the limited number of features on the page, there will be no need for automating the tests.

### Acceptance Criteria:

1. All connect via SSH test cases should pass
2. All connect via VNC test cases should pass
3. All connect via Web test cases should pass

### Tasks:

* #650 - Test connect via Web
* #649 - Test connect via VNC
* #648 - Test connect via SSH

**#643 - Modify the Web Application**

### Description:

* A cornerstone of the application is the ability to connect via web, VNC, and SSH. As it stands, only connection via SSH is currently available, and so the ability to connect via web and VNC will be introduced. In addition, the final website behaviors will be introduced.

### Tasks:

* #647 - Add addition page behavior
* #646 - Incorporate disconnect feature
* #645 - Allow for connection via Web
* #644 - Allow for connection via VNC

# System Design

This section explains how the whole system was constructed and designed. It will also explain their purpose and how could be use together.

## Architectural Patterns

Tunneler:

* Server was implemented using python from scratch, no framework was used for this part of the project. It is used to redirect all data received.
* Client/Tunneler were implemented using python and no outside framework was used. These two software are used a source and target for communication and connection purpose each having different behavior.

PubSub:

* PubSub was built on top RabbitMQ using its existing APIs to interface with it and it internal plugins and features. It was built using Python and Javascript
* The service works on a system of clients and a server that will store and dispatch messages to the correct recipients. The users interested in a conversation will subscribe to a channel, each with their own messaging queue which will store messages directed to the owner of the queue.
* Queues will be able to be deleted or set up for automatic deletion.

## System and Subsystem Decomposition

Tunneler:

The three software used to implement this concept of tunneling network connections and information were implemented using python without the use of any external framework. Ther **Server** is the middle man between **Client** and **Tunneler**, is the piece of software which redirects every bit of data from the source to the desired target and backwards. The **Client** is started passing the destination (**Tunneler**) information in order to let the server know to whom do it wishes to communicate. The **Tunneler** on the other hand is the target software which simply waits for any request and it responds with the data from the service requested.

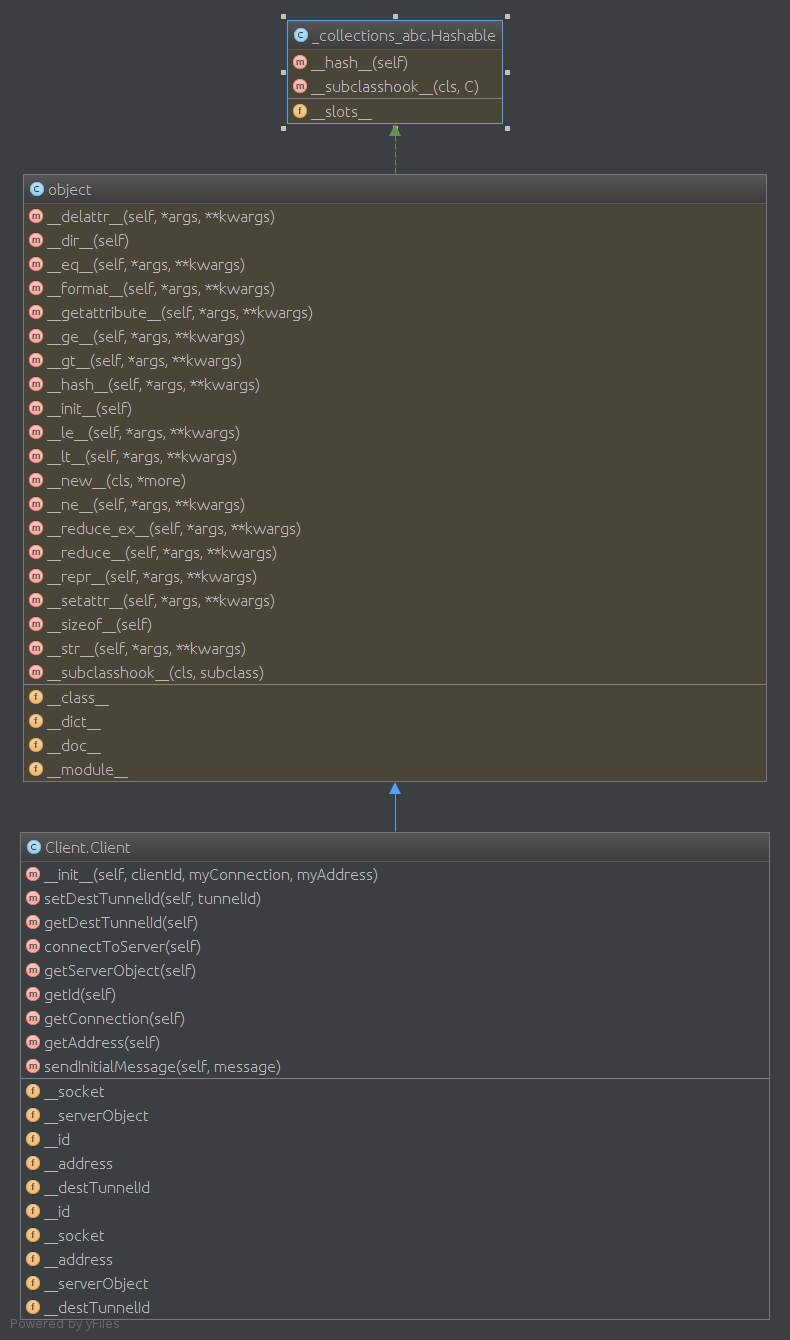
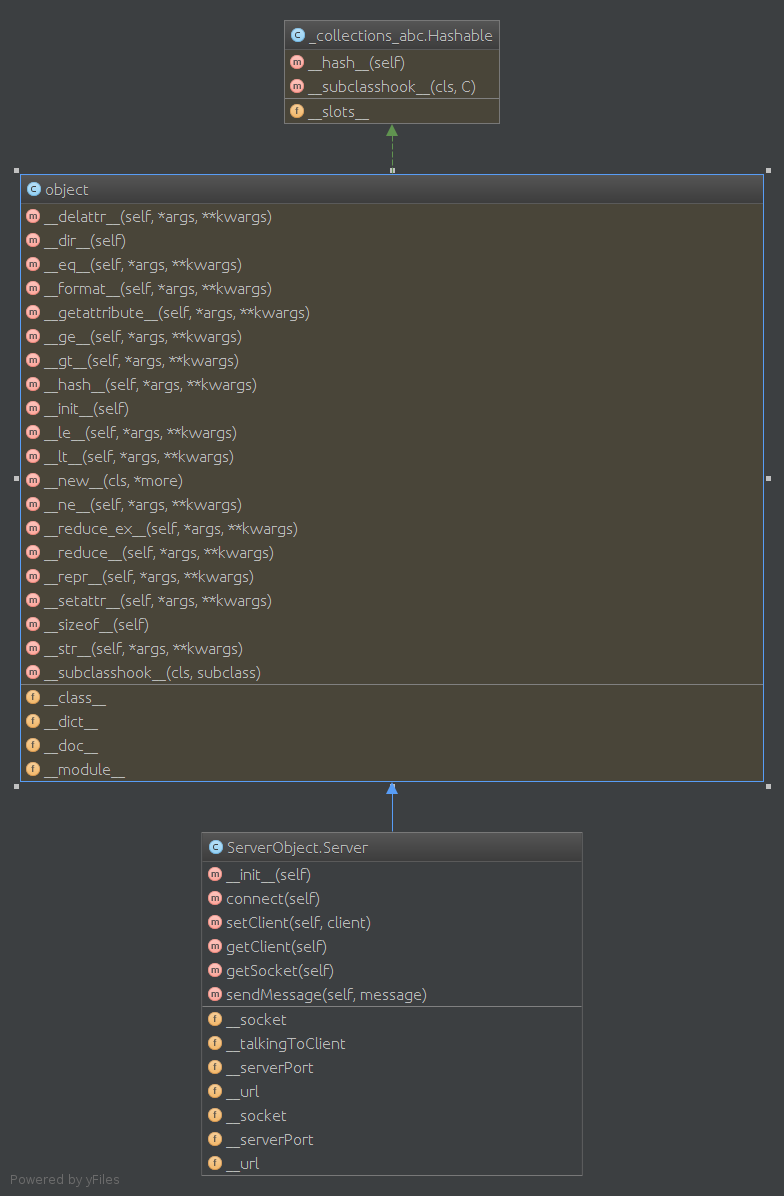
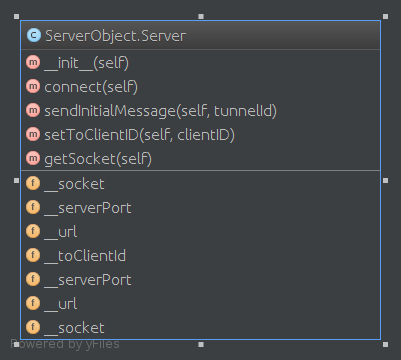
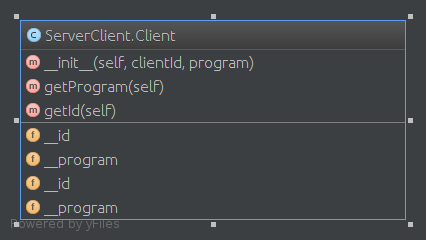
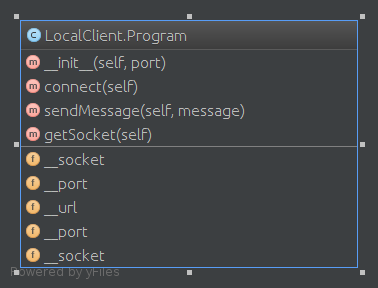
Figure C01 - Client Class Diagrams

Figure C02 - Tunneler Class Diagram

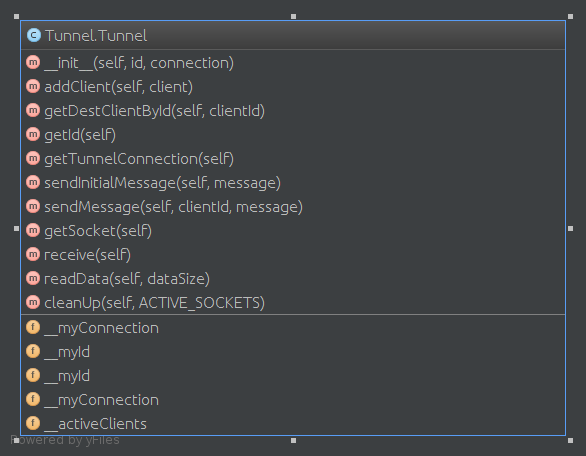
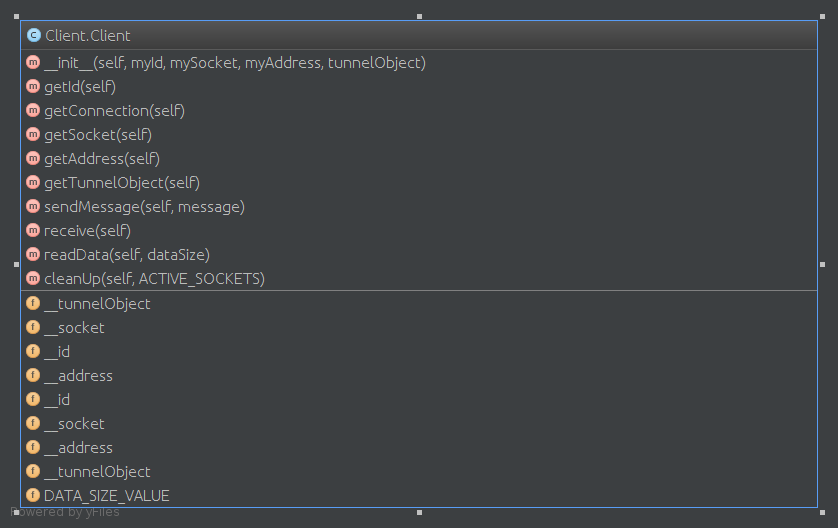


Figure C03 - Server Class Diagram

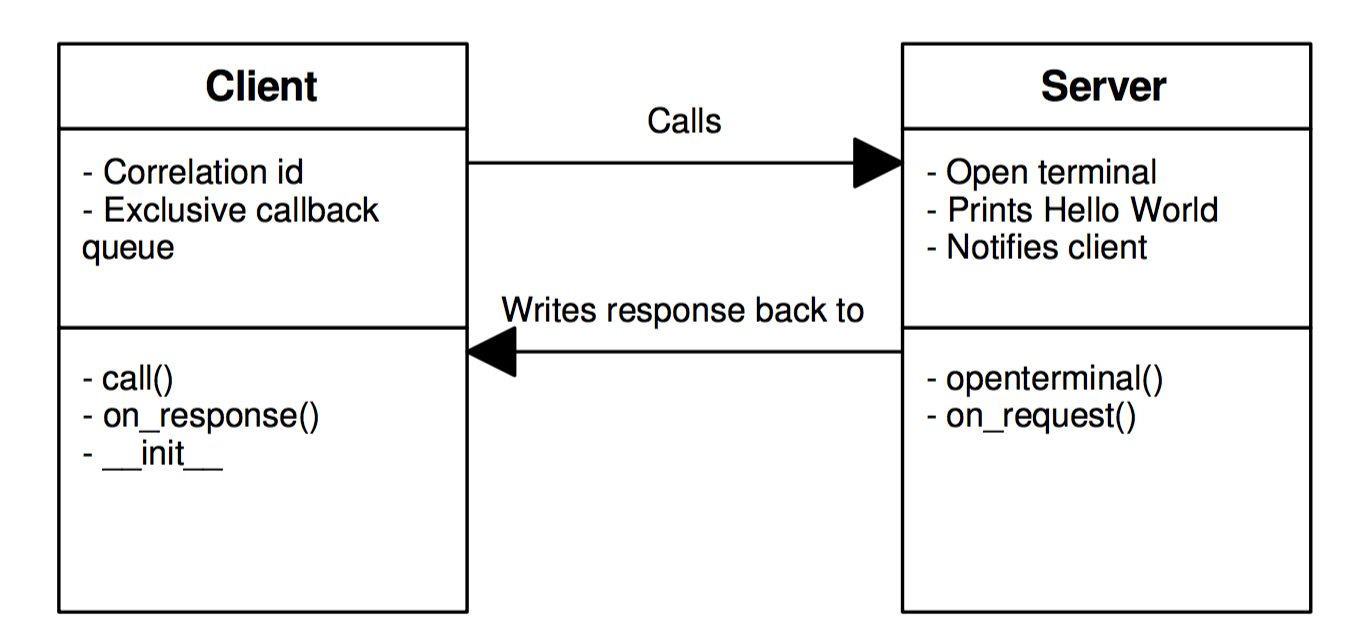


Figure C04 - PubSub proof of concept class diagram

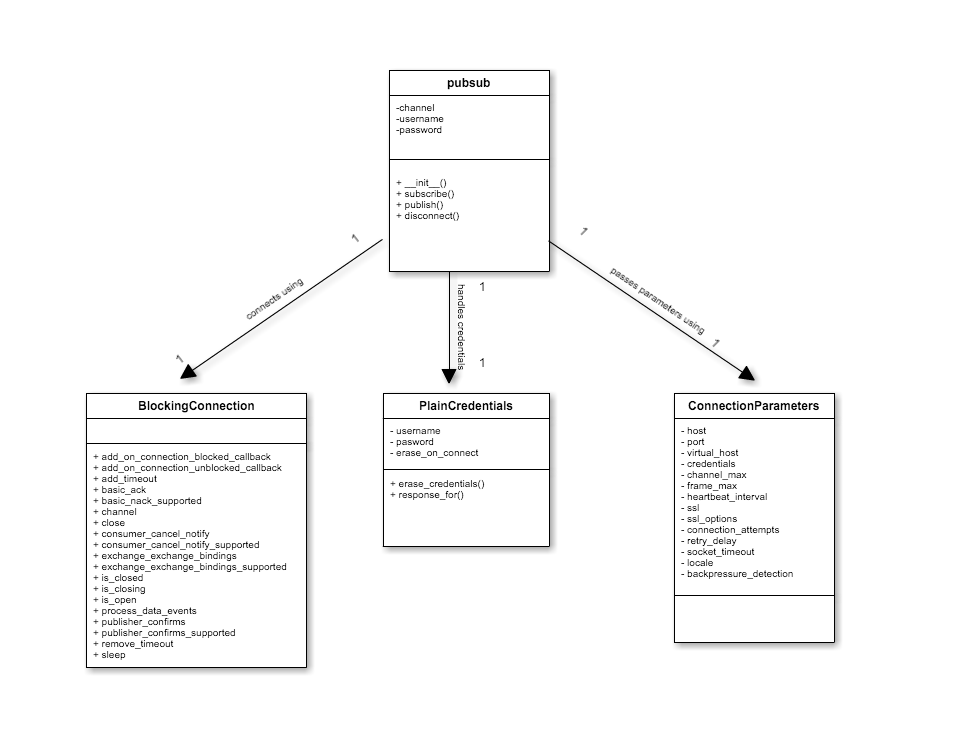


Figure C05 - An overview of the Python SDK

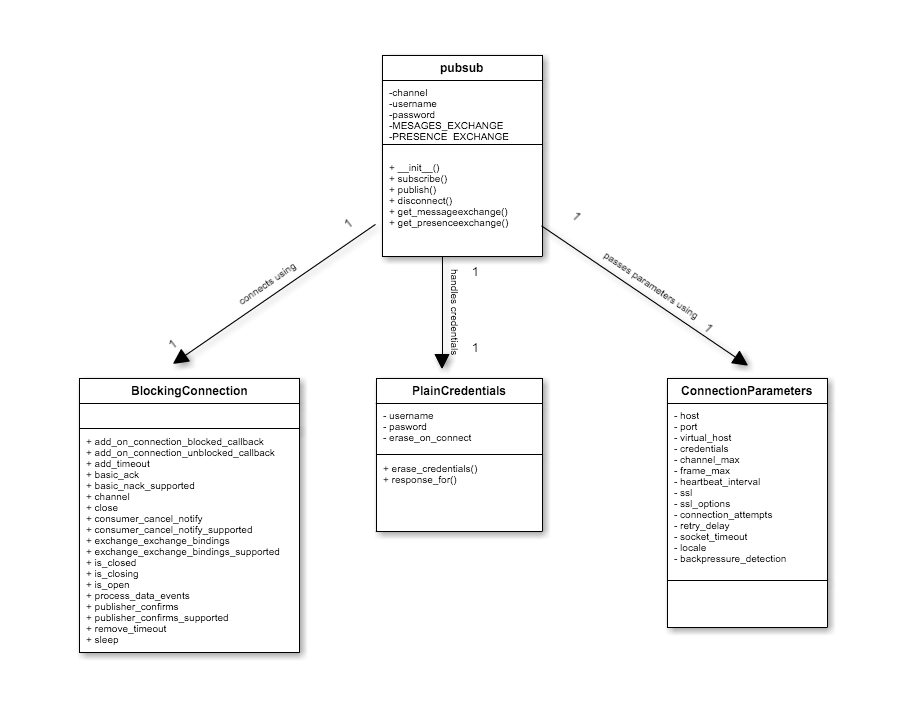


Figure C06 - Python SDK as of Presence changes introduced

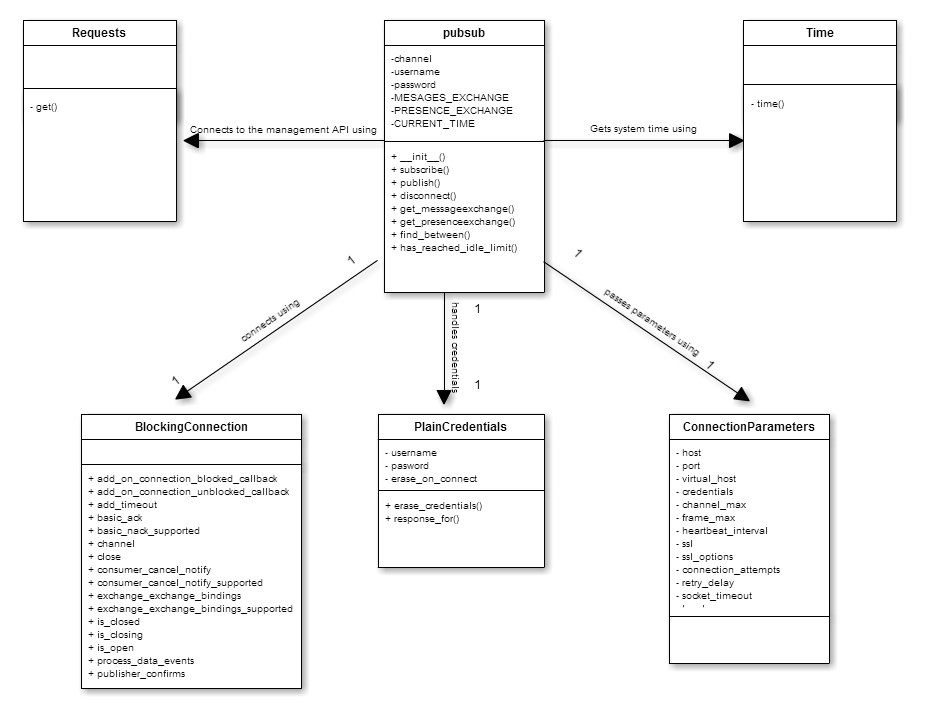


Figure C07 - Initial Python SDK Heartbeat design

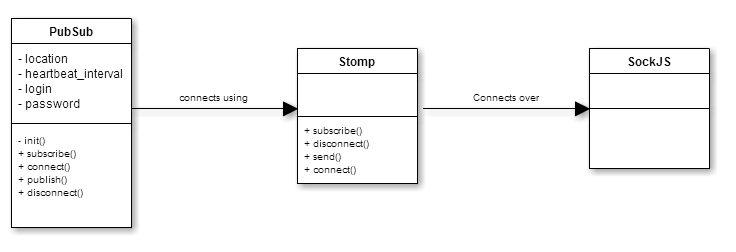


Figure C08 - Javascript SDK design

## Deployment Diagram

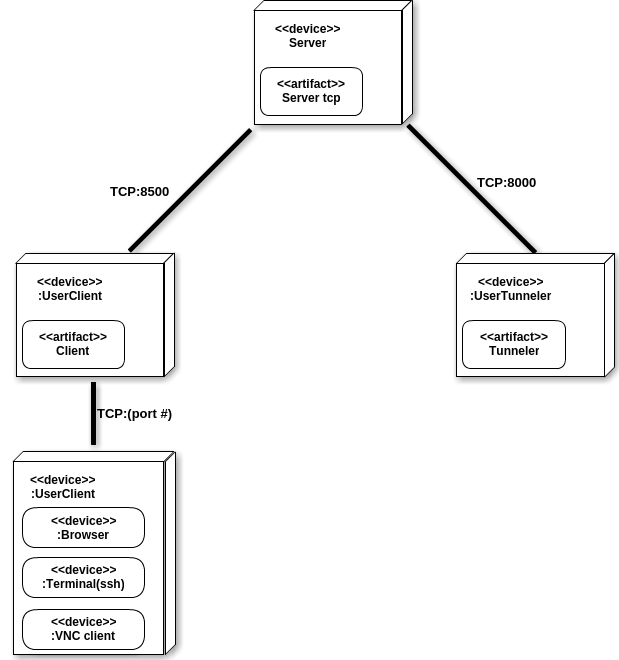


Figure D1 - Tunneler Deployment Diagram

## 

Figure D2 - PubSub Deployment Diagram

## Design Patterns

Tunneler:

The design of the Tunneler project was created so it required some sort of identification of source and target software for simpler communication algorithm. This way the Server has a simple way to identify to which connection is the current message being sent.

PubSub:

The main design pattern followed throughout the development cycle was the publish/subscribe model. This was done because it was a simple yet robust way to transport messages to a large number of recipients. Also, this model allows for the decoupling of methods, meaning no two methods will be dependant on one another.

# System Validation

The system validation for the tunneler program was done through manual testing of the components due to the difficulty of different environments communicating at the same time.

**User Story # 553 - Initializing client**

System Tests

|  |  |
| --- | --- |
| Identifier | ST01\_ClientProgramStartUpNoParameters |
| Purpose | Ensure client program starts correctly |
| Setup | open terminal and run client using “python” command (no parameters) |
| Input | no parameters |
| Expected result | missing parameters |
| Result | as expected! |

|  |  |
| --- | --- |
| Identifier | ST02\_ClientProgramStartUpWithParameters |
| Purpose | Ensure client program starts correctly |
| Setup | open terminal and run client using “python” command (no parameters) |
| Input | no parameters |
| Expected result | client listening to given port for incoming connections |
| Result | as expected! |

**User Story # 562 - Communication**

System Tests

|  |  |
| --- | --- |
| Identifier | ST03\_ProgramCommunication |
| Purpose | Ensure client, server, and tunnel communication is correct |
| Setup | connect client and tunnel to the server and begin a ssh session. |
| Input | ssh session to tunnel, navigate through the system |
| Expected result | responsive communication, no errors |
| Result | error in communication. “Reading half data” |

|  |  |
| --- | --- |
| Identifier | ST04\_ProgramCommunicationAfterFix |
| Purpose | Ensure client, server, and tunnel communication is correct |
| Setup | connect client and tunnel to the server and begin a ssh session. |
| Input | ssh session to tunnel, navigate through the system |
| Expected result | responsive communication, no errors |
| Result | as expected. |

**User Story # 578 - Connection handling**

System Tests

|  |  |
| --- | --- |
| Identifier | ST05\_ServerConnectionHandling |
| Purpose | Disconnecting client or tunnel while communication through server |
| Setup | established connection and communication in progress (vnc) |
| Input | vnc connection establish among clients than tunnel disconnects |
| Expected result | server catches the disconnection and closes all connections related with tunnel |
| Result | error in connection. |

|  |  |
| --- | --- |
| Identifier | ST06\_ServerConnectionHandlingAfterFix |
| Purpose | Disconnecting client or tunnel while communication through server |
| Setup | established connection and communication in progress (vnc) |
| Input | vnc connection establish among clients than tunnel disconnects |
| Expected result | server catches the disconnection and closes all connections related with tunnel |
| Result | as expected. |

**User Story # 547 - Create an SDK for Javascript**

|  |  |
| --- | --- |
| Identifier | ST07\_SucessfulConnect |
| Purpose | Ensuring that any client implementing this SDK can successfully connect to a RabbitMQ server |
| Setup | Import pubsub.js into the html test client  Instantiate pubsub |
| Input |  |
| Expected result | Implementing client successfully established a connection to the RabbitMQ server |
| Result | Implementing client successfully established a connection to the RabbitMQ server |

|  |  |
| --- | --- |
| Identifier | ST08\_CloseClientDuringConnect |
| Purpose | Ensuring that any client implementing this SDK can successfully handle an abrupt cancel during connection |
| Setup | Import pubsub.js into the html test client  Instantiate pubsub  Close implementing client immediately |
| Input |  |
| Expected result | Implementing client does not establish a connection with RabbitMQ server |
| Result | Implementing client does not establish a connection with RabbitMQ server |

|  |  |
| --- | --- |
| Identifier | ST12\_SucessfulSubscribe |
| Purpose | Ensuring that any client implementing this SDK can successfully subscribe to a queue in RabbitMQ |
| Setup | Import pubsub.js into the html test client  Instantiate pubsub |
| Input | Use the pubsub.subscribe() function to subscribe to a queue in RabbitMQ |
| Expected result | Implementing client successfully subscribes to the queue in the server |
| Result | Implementing client successfully subscribes to the queue in the server |

|  |  |
| --- | --- |
| Identifier | ST13\_SubscribeToNoneExistantQueue |
| Purpose | Ensuring that any client implementing this SDK can successfully handle subscription to a non-existent queue |
| Setup | Import pubsub.js into the html test client  Instantiate pubsub |
| Input | Use the pubsub.subscribe() function to subscribe to a queue in RabbitMQ that has not yet been specified |
| Expected result | Implementing client successfully discards incoming messages |
| Result | Implementing client successfully discards incoming messages |

|  |  |
| --- | --- |
| Identifier | ST14\_SuccessfullyPublishToQueue |
| Purpose | Ensuring that any client implementing this SDK can successfully publish to a specified queue |
| Setup | Import pubsub.js into the html test client  Instantiate pubsub |
| Input | Use the pubsub.send() function to send a message to the specified queue |
| Expected result | Implementing client successfully sends message to specified queue |
| Result | Implementing client successfully sends message to specified queue |

**User Story # 546 - Create an SDK for Python**

|  |  |
| --- | --- |
| Identifier | ST09\_SucessfulConnect |
| Purpose | Ensuring that any client implementing this SDK can successfully connect to a RabbitMQ server |
| Setup | Import pubsub.py into the test client  Instantiate pubsub |
| Input |  |
| Expected result | Implementing client successfully established a connection to the RabbitMQ server |
| Result | Implementing client successfully established a connection to the RabbitMQ server |

|  |  |
| --- | --- |
| Identifier | ST10\_CloseClientDuringConnect |
| Purpose | Ensuring that any client implementing this SDK can successfully handle an abrupt cancel during connection |
| Setup | Import pubsub.py into the test client  Instantiate pubsub  Close implementing client immediately |
| Input |  |
| Expected result | Implementing client does not establish a connection with RabbitMQ server |
| Result | Implementing client does not establish a connection with RabbitMQ server |

|  |  |
| --- | --- |
| Identifier | ST15\_SucessfulSubscribe |
| Purpose | Ensuring that any client implementing this SDK can successfully subscribe to a queue in RabbitMQ |
| Setup | Import pubsub.py into the test client  Instantiate pubsub |
| Input | Use the pubsub.subscribe() function to subscribe to a queue in RabbitMQ |
| Expected result | Implementing client successfully subscribes to the queue in the server |
| Result | Implementing client successfully subscribes to the queue in the server |

|  |  |
| --- | --- |
| Identifier | ST16\_SubscribeToNoneExistantQueue |
| Purpose | Ensuring that any client implementing this SDK can successfully handle subscription to a non-existent queue |
| Setup | Import pubsub.py into the test client  Instantiate pubsub |
| Input | Use the pubsub.subscribe() function to subscribe to a queue in RabbitMQ that has not yet been specified |
| Expected result | Implementing client successfully discards incoming messages |
| Result | Implementing client successfully discards incoming messages |

|  |  |
| --- | --- |
| Identifier | ST17\_SuccessfullyPublishToQueue |
| Purpose | Ensuring that any client implementing this SDK can successfully publish to a specified queue |
| Setup | Import pubsub.js into the html test client  Instantiate pubsub |
| Input | Use the pubsub.send() function to send a message to the specified queue |
| Expected result | Implementing client successfully sends message to specified queue |
| Result | Implementing client successfully sends message to specified queue |

**User Story # 574 - Have both clients communicate with one another, using the libraries**

|  |  |
| --- | --- |
| Identifier | ST18\_SucessfulPropogateMesssageAcross |
| Purpose | Ensuring that any client implementing either SDK can successfully send and consume a message |
| Setup | Import pubsub.py into the test client  Import pubsub.js into the html test client  Instantiate pubsub on both clients |
| Input | Use the pubsub.subscribe() function on the Python client  Use the pubsub.subscribe() function on the Javascript client  Use the pubsub.publish() function on the Python client  Use the pubsub.send() function on the Javascript client |
| Expected result | The messages sent from both clients should appear on both clients |
| Result | The messages sent from both clients appeared on both clients |

**User Story # 632 - Test software and feature ssh/web/vnc**

System Tests

|  |  |
| --- | --- |
| Identifier | ST20\_startigSSHsession |
| Purpose | Ensure ssh session starts correctly |
| Setup | start pubsub agent and send command to connect using ssh |
| Input | from web connect using ssh |
| Expected result | services started/ terminal with ssh session started |
| Result | as expected |

|  |  |
| --- | --- |
| Identifier | ST21\_startigWEBsession |
| Purpose | Ensure web session starts correctly |
| Setup | start pubsub agent and send command to connect using web |
| Input | from web connect using web service |
| Expected result | services started/ browser open on desire page |
| Result | as expected |

|  |  |
| --- | --- |
| Identifier | ST22\_startigVNCsession |
| Purpose | Ensure vnc session starts correctly |
| Setup | start pubsub agent and send command to connect using vnc |
| Input | from web connect using vnc service |
| Expected result | services started/ vnc client software |
| Result | as expected |

**User Story # 628 - Manual testing for system connection**

|  |  |
| --- | --- |
| Identifier | ST23\_ServerInstallation |
| Purpose | Ensure server install correctly using setup.sh file |
| Setup | run setup.sh file |
| Input | terminal ./setup.sh |
| Expected result | server directory moved to different location and started |
| Result | as expected |

|  |  |
| --- | --- |
| Identifier | ST24\_ClientTunnelerPubsubAgent setup file |
| Purpose | Ensure all software files are moved correclty using setup file |
| Setup | run setup.sh file |
| Input | terminal ./setup.sh |
| Expected result | all software is moved to desired location |
| Result | as expected |

# Glossary

* **Audit Data**: Webhost running locally for testing purpose.
* **Agent**: For communicating with server and on user’s computer for receiving/sending information purpose.

Tunneler:

* **Client:** For receiving client connections and sending information to server
* **Server:** For receiving tunnels and client connections and redirecting messages to them
* **Tunnel:** For receiving messages to the from server and giving to local service

PubSub:

* **Publish:** Sending of a message to a queue or channel.
* **Subscribe:** The act of consuming messages from a queue.
* **Exchange:** Delegator messages to respective queues/destinations.
* **Plugin:** Features available for use on RabbitMQ that must be activated in order to be used.
* **Client**: Any software that implements PubSub.
* **Queue:** A data structure used in our case as a means to store messages. The queue follows a first in first out model (FIFO).

# Appendix

## Appendix A - UML Diagrams

### Static UML Diagrams

Tunneler:

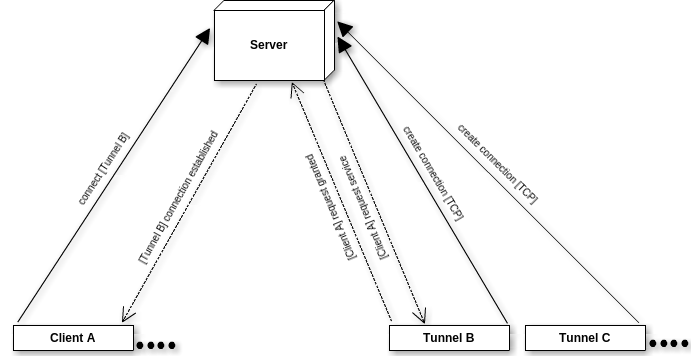


Figure #5.008 - High level program diagram

### 

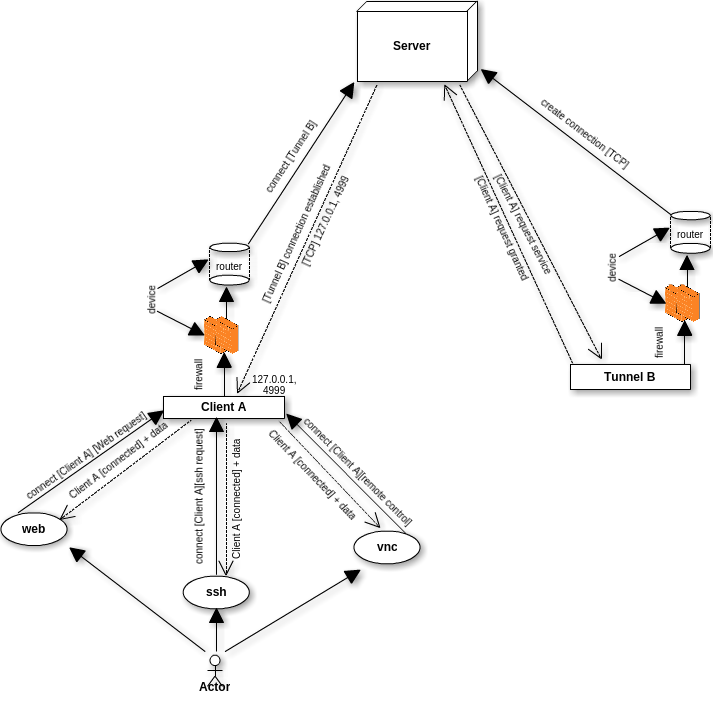


Figure #5.009 - Detailed High level program diagram

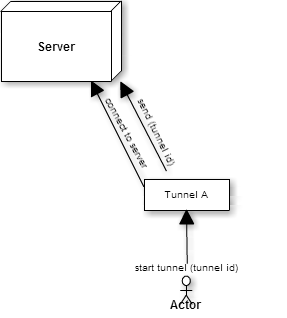


Figure #5.021 - Tunnel start up

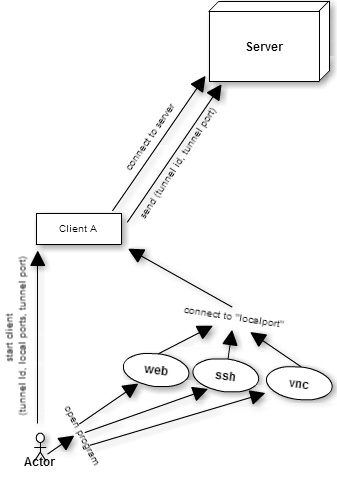


Figure #5.022 - Client start up And sending request

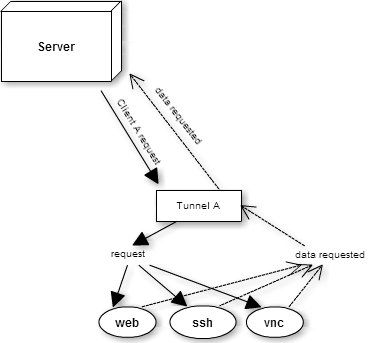


Figure #5.023 - Tunnel receiving request

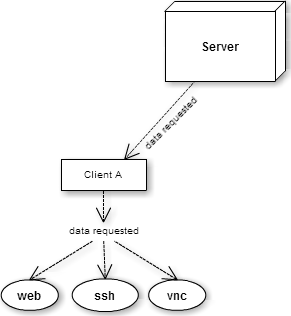


Figure #5.024 - Client receiving request

### 

Figure 5.014 - High level representation of PubSub

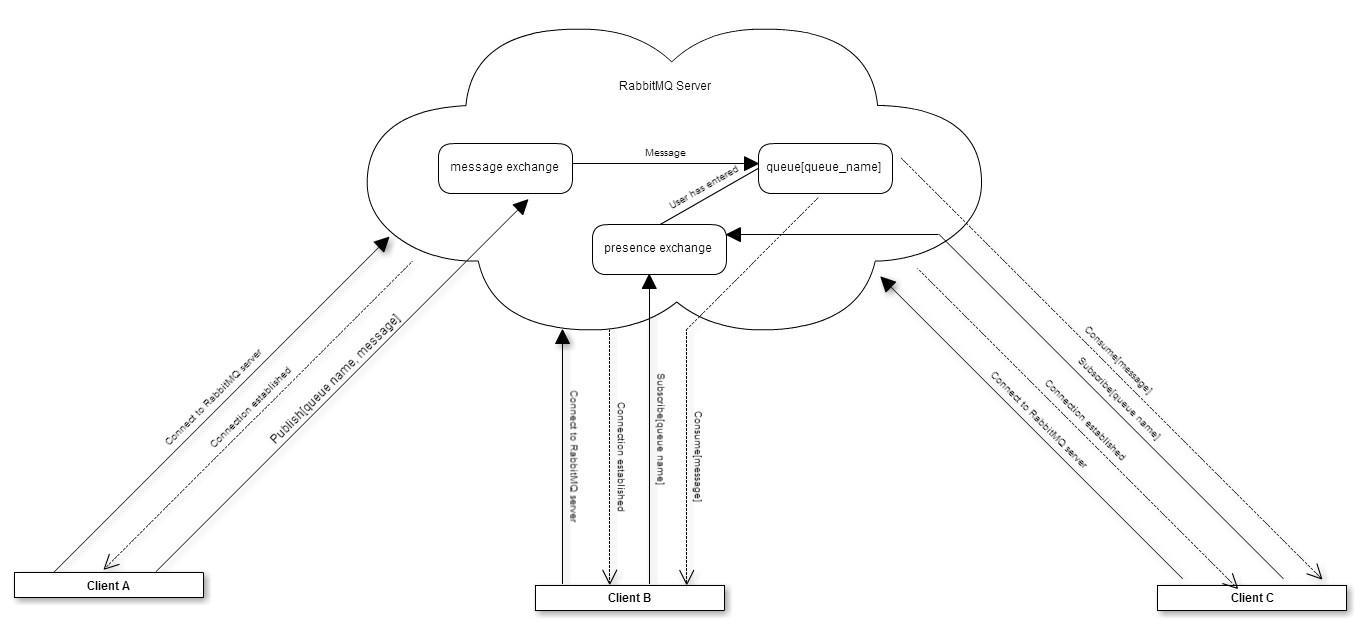


Figure 5.015 - More detailed high level diagram of PubSub

### 

### 

### 

### Dynamic UML Diagrams

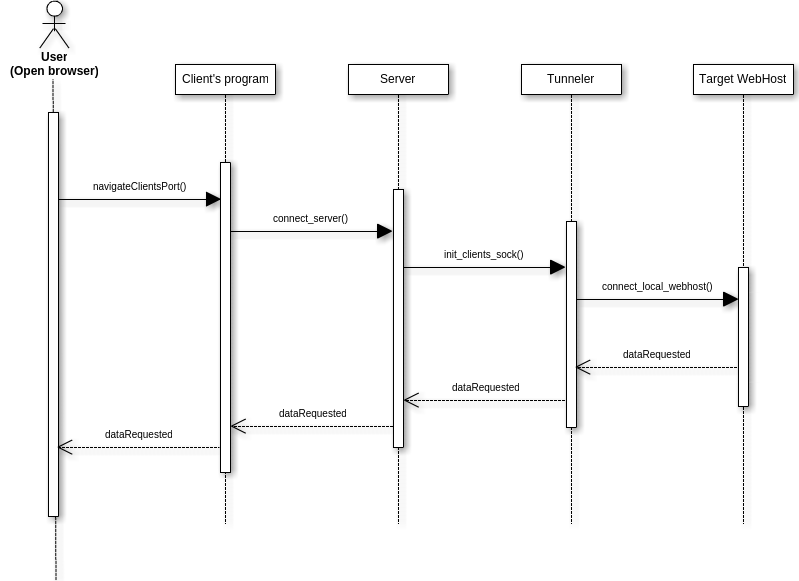


Figure #5.001 - Proof of concept System Communication

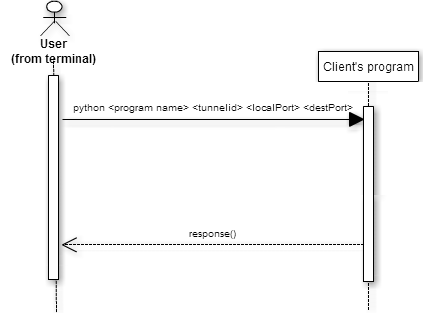


Figure #5.005 - Initializing client

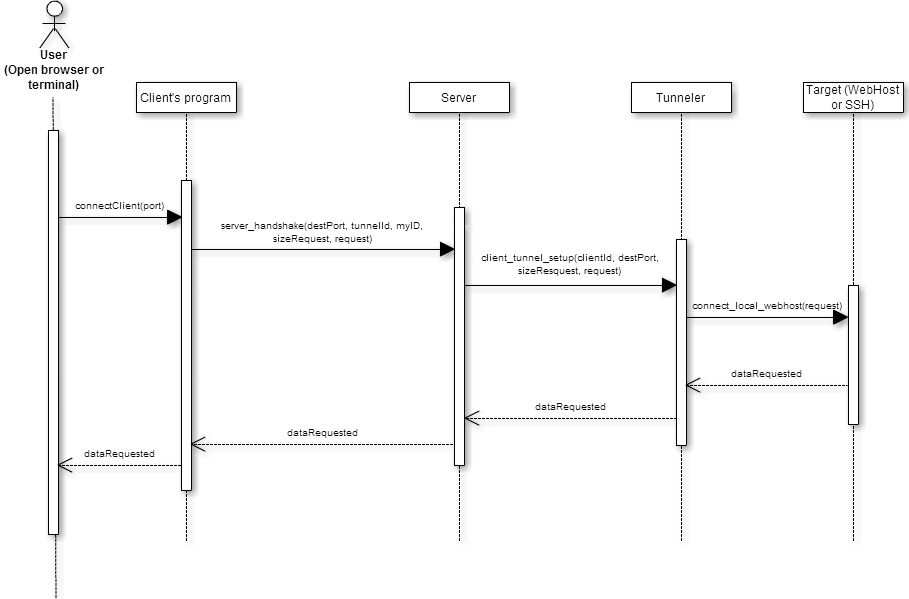
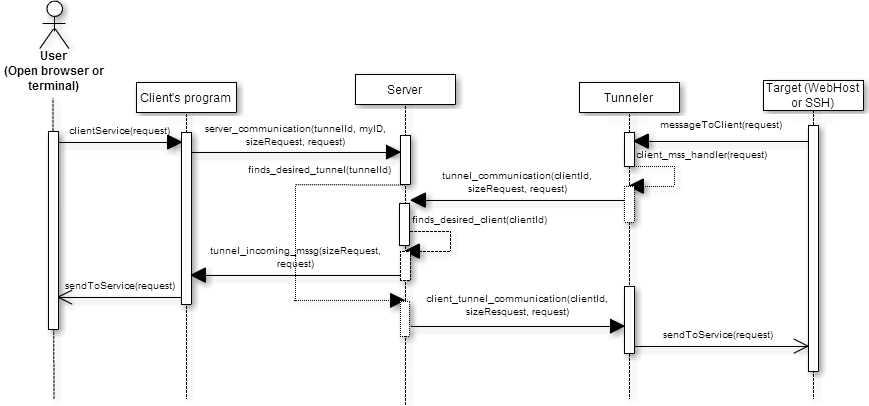


Figure #5.006 - Implementing data stream (Initial communication setup)

Figure #5.007 - Implementing data stream (Established communication)

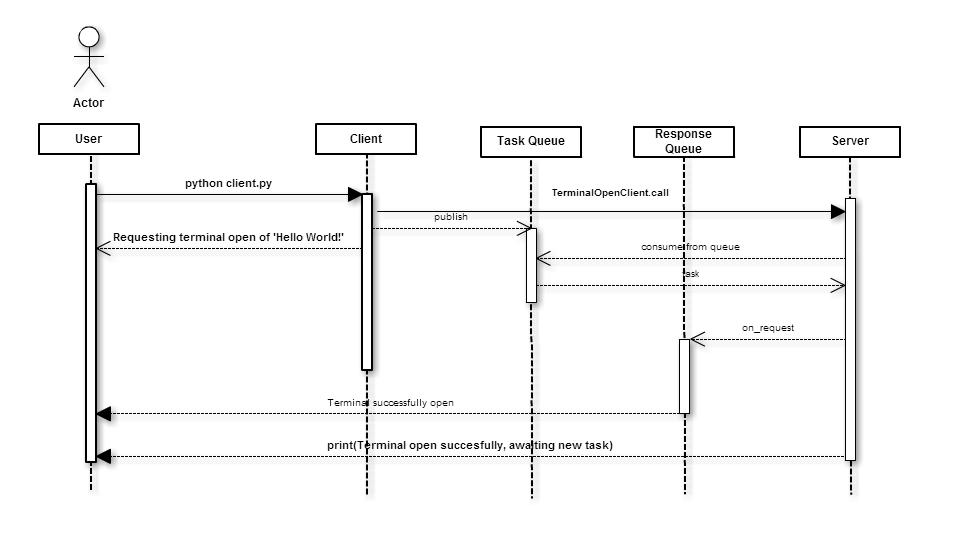


Figure #5.002 - Proof of concept PubSub

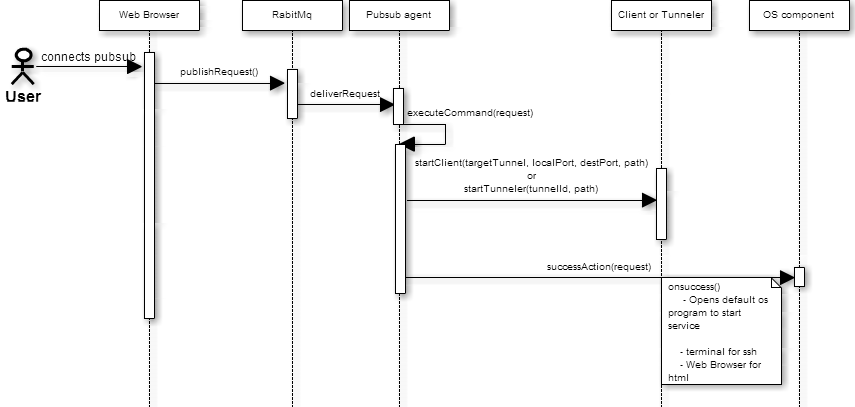


Figure #5.021 - Pubsub agent running

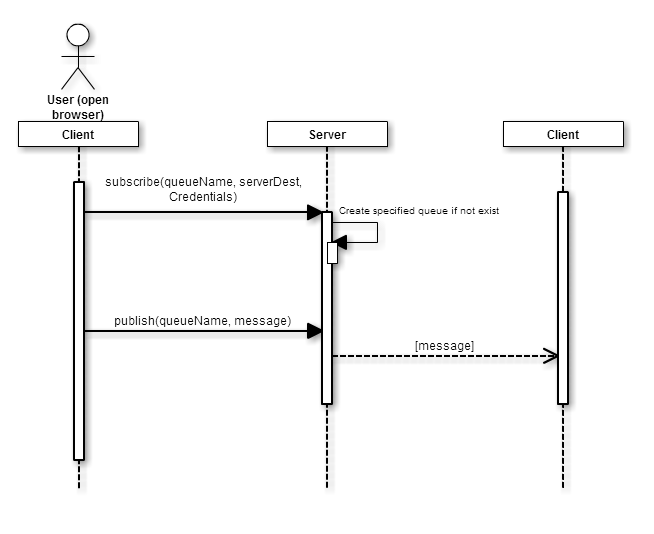


Figure #5.013 - Client from user story #539

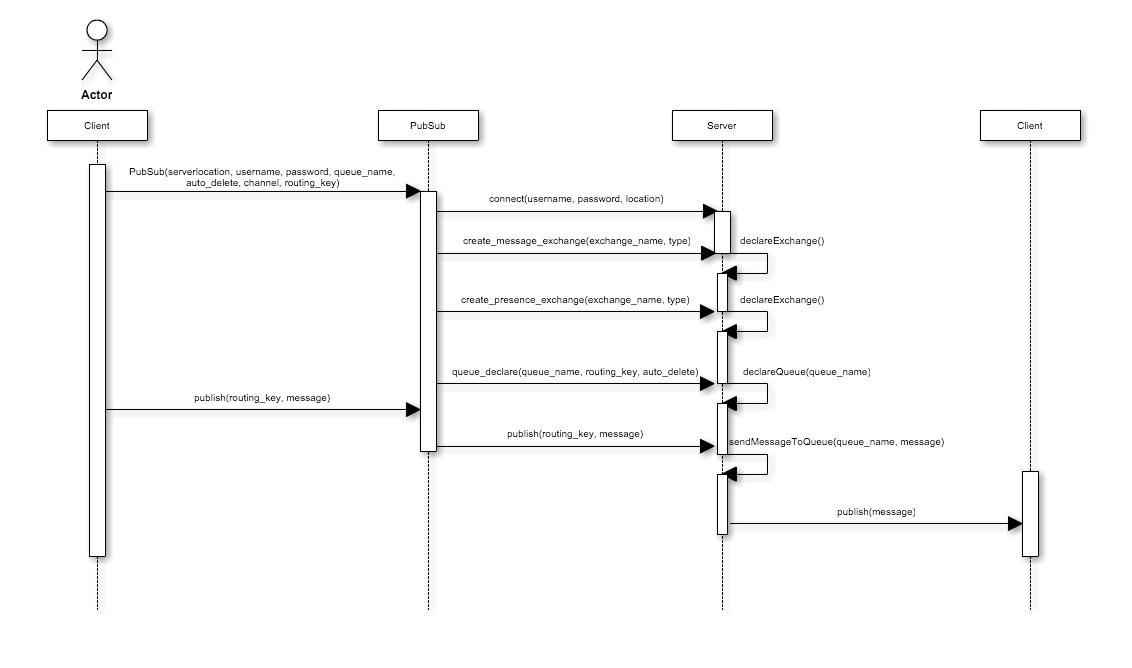


Figure #5.014 - Overall publishing model

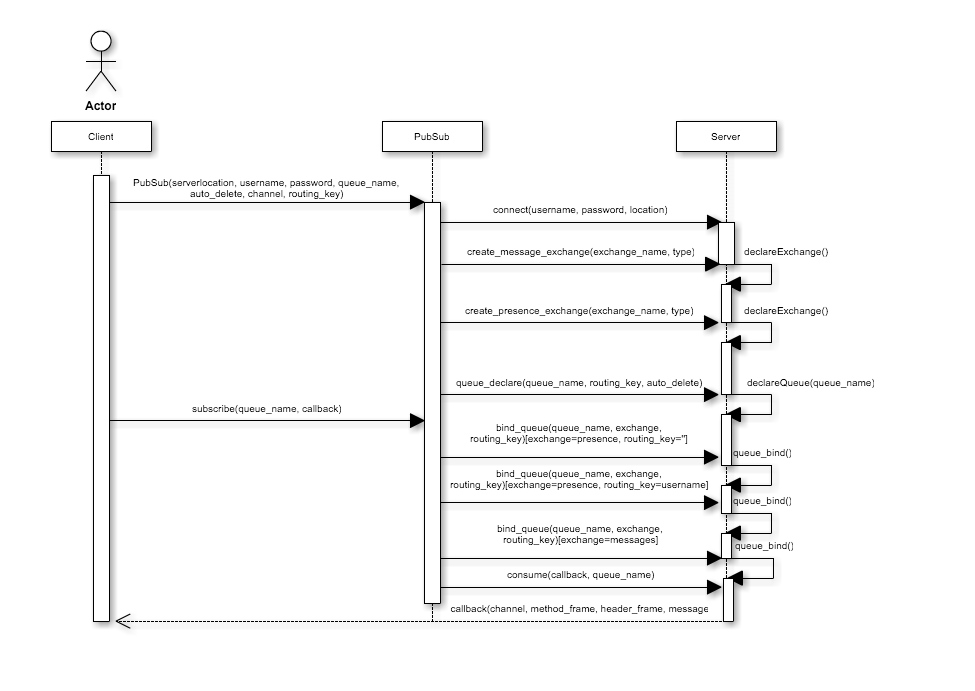


Figure #5.015 - Overall subscribe model

## Appendix B - User Interface Design

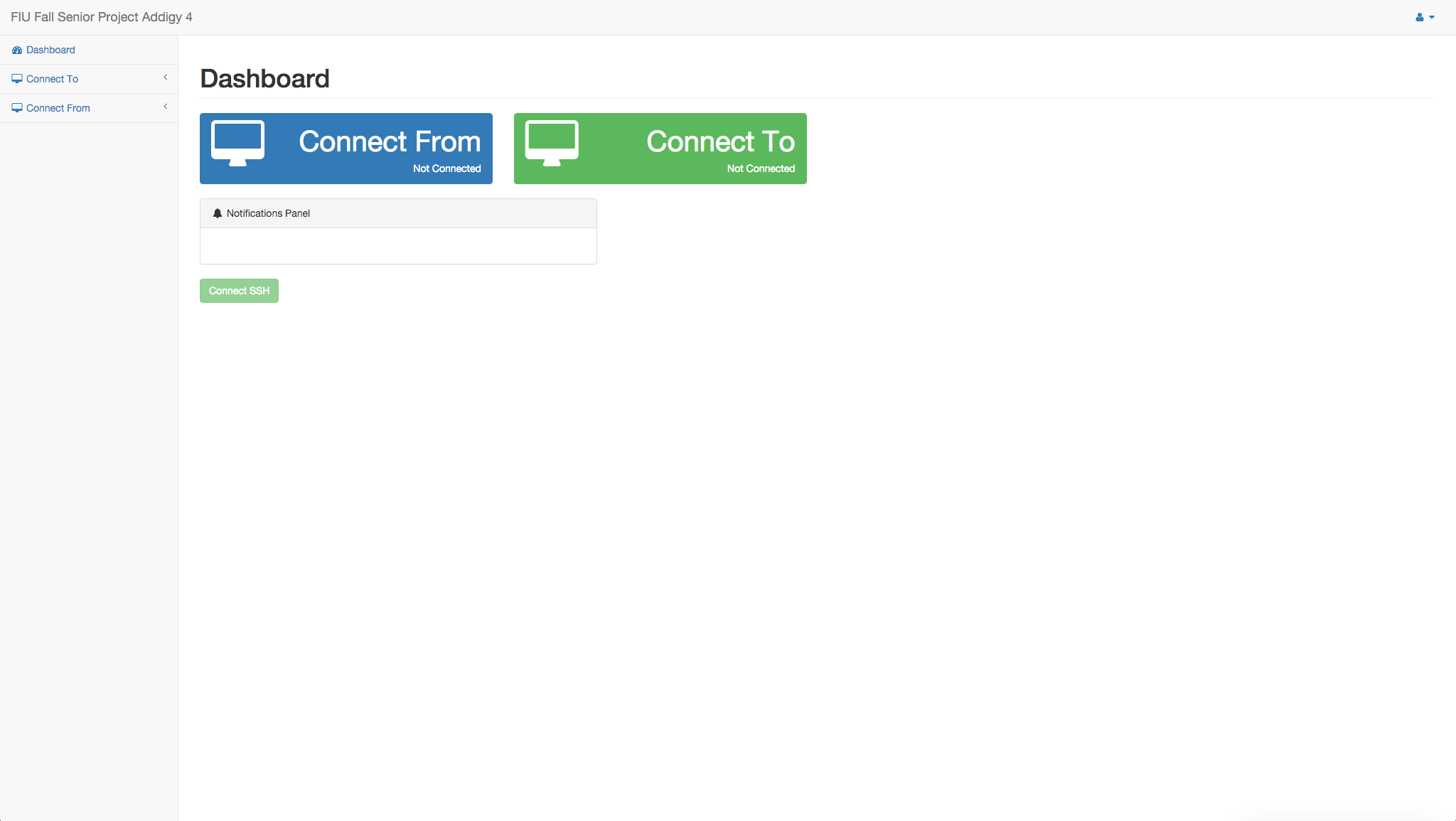


Figure 5.015 - User Interface for the web application

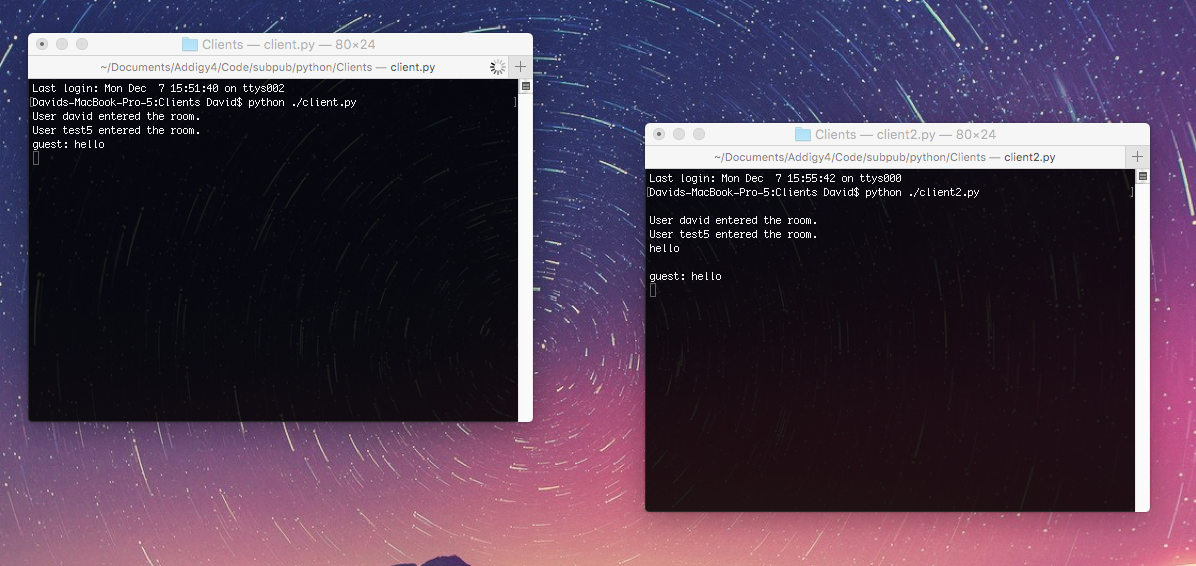


Figure 5.016 - Desktop clients implementing the desktop solution

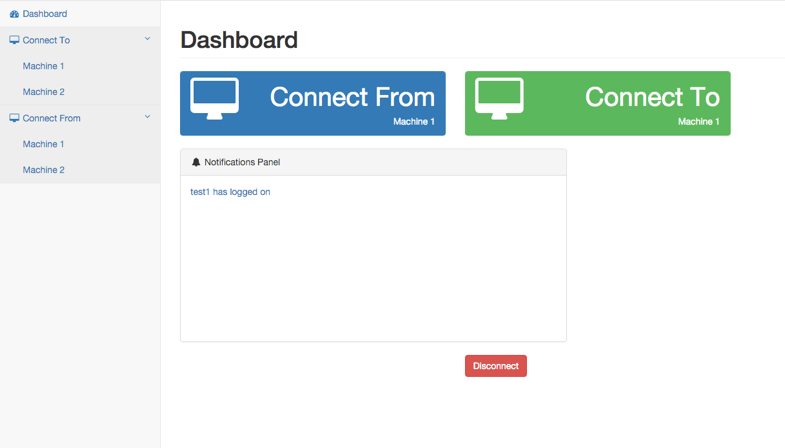


Figure 5.017 - Final view for the web client

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** Sept 11, 2015

**Attendees:**Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Achieved:

* Research on different methodologies for point to point communication among systems.
* Research on the publish/subscribe model
* Setting up python environment.
* Creation of a simple local python webhost to implement proof of concept.
* Designing Client/Server/Tunnel communication and interaction
* Designed simple clients to test development with RabbitMQ
* Creation of Github structure for project to follow development process.
* Implementation of Client (source) software.
* Implementation of Tunneler (target) software.
* Implementation of Server (middle man) software.
* Implementation of PubSub software for python
* Editing documentation for sprint one results.

**Sprint 2 Report**

**Date:** Sept 25, 2015

**Attendees:** Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Achieved:

* Implementation of data stream communication between Client/Server/Tunneler.
* Tunneler accepting ssh session request.
* Client initialization for web session or ssh session.
* Initializing client passing parameters for communication purpose.
* Researched into Javascript
* Javascript SDK was developed using results of the research
* Using the Javascript SDK, a web client was developed.

**Sprint 3 Report**

**Date:** Oct 10, 2015

**Attendees:** Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Achieved:

* Refactoring of Client and Tunneler implementation to follow proper OOP guidelines and style and to be more scalable.
* Manual testing for Server connection handling and software communication.
* Fixing of communication among server-client and server-tunneler.
* Adding remote control feature to software implementation.
* Refined the Python SDK.
* Ensured both platforms were cross platform compatible
* Got a better idea of what design to approach presence.
* Using what was learned about presence in RabbitMQ, it was introduced into PubSub gradually

**Sprint 4 Report**

**Date:** Oct 23, 2015

**Attendees:** Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Achieved:

* Fix multiple connection to the same tunnel.
* Adding feature to tunneler software to handle request from different clients.
* Client software may only connect to a single tunnel at a time.
* Refactoring of Server implementation to handle better connections and the use of threads for communication handling.
* Presence was introduced to the Python SDK
* Presence was introduced into the Javascript SDK
* Code for both the Python and Javascript solutions was reworked to be a little neater.

**Sprint 5 Report**

**Date:** Nov 06, 2015

**Attendees:** Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Acchieved:

* Creation of high level project diagrams.
* Updating information for all readme files within the project.
* Updating github code to the latest implementation.
* Fix and refactor tunneler design.
* Refactoring Server to better handle disconnections from tunnel/client
* Testing of server when client/tunneler disconnects.
* Testing communication between three software in real workspace.
* Testing starting up client software.
* Began initial designs of idle session timeout.

**Sprint 6 Report**

**Date:** Nov 20, 2015

**Attendees:** Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Achieved:

* Implementation of new Pubsub agent software.
* Designing structure of new Pubsub agent software.
* Setting up FIU virtual box so it host Server software.
* Updating all documents and github code.
* A web application was made that married both projects.
* Logic was modified based off the PubSub agent softwares requirements
* One final attempt at implementing idle session timeout.

**Sprint 7 Report**

**Date:** Dec 04, 2015

**Attendees:** Masoud Sadjadi, Carlos Ruiz, David Romero. Jason, Javier Carmona, Ayme

**Discussed Topics:**

Acchieved:

* Testing new ssh/web/vnc connection using pubsub agent software to start communication.
* Add feature to start application to connect to client software.
* Add feature to terminate communication and software.
* Update project documentation to the latest development status for final report.
* Manual testing for connection and communication using ssh/web/vnc sessions
* Behavior of web application was tested.
* Any missing visual elements and logic were tended to.

## 

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** September 11, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* All user stories for tunneler were completed successfully before the marked deadline.
* All user stories for PubSub were drafted in a timely manner.
* Designing of proof of concept was achieved in a simple straightforward manner.
* Proof of concept was achieved using a simple web request from a browser using implemented software, Connecting to Client, communicating through Server, and redirecting information to the Tunneler which then connected to a local Webhost and sent the response back.
* Final Project deliverable was kept up to date with in the user stories sections and other sections regarding the product were updated as progress was made in the project.
* Designed a simple, straightforward proof of concept that was able to connect to a RabbitMQ server. It would simply open a terminal and respond with a message notifying the original sender that their message has been received and the terminal has been opened.

**Sprint 2 Retrospective**

**Date:** Sept 25, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* All user storied designated for this sprint 2 were completed regarding Tunneler software.
* New implementation for client-server and tunneler-server communication was achieved in order to create and maintain a data stream communication between them.
* Client now able to start ssh session and Tunneler now able to receive communication and ssh session.
* Scrums meetings were a little off but for future sprints should be more up-to date and often.
* An SDK was developed for web clients using Javascript. This newly introduced solution allows for similar behavior over the web.
* A test client was devised that would be using the newly developed solution.

**Sprint 3 Retrospective**

**Date:** Oct 10, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* Some errors were found when testing how the server were handling when Client/Tunneler disconnects and a new method was implemented.
* Client and Tunneler code was refactored because previous implementation was too straightforward therefore not very scalable or following a OOP design.
* New feature added to software which will now be able to create remote control sessions. Used and tested with vnc software.
* A simple presence was incorporated into the project. The current system operates much like originally proposed idea, but instead of working off queue, it will work off exchanges, in order to cover more than a simple queue would with its FIFO behavior.
* Cross platform compatibility was tested and two test clients were successfully able to communicate with each other, able to engage in a simple conversation.

**Sprint 4 Retrospective**

**Date:** Oct 23, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* Due to some error in communication when connecting multiple clients to the same tunnel code had to be refactored to correctly respond to specific clients when asking for specific information.
* Server implementation was change for better scalability and connection handling now using a fix number of threads to handle communication between client and tunneler.
* Testing of software was performed and all errors were fixed.

**Sprint 5 Retrospective**

**Date:** Nov 06, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* For better understanding of the project concept High level project diagrams were created for users and to be used as reference of the process.
* All files were updated on github, all the readme files and documentation as it was a little delayed this sprint because of all the fixing that have been done to the project but for next sprints this will be updated every time something changes.
* Test were done to the communication between the three software on real environment and some changes had to be made to the tunneler for better communication and also to the Server to better handle disconnection without closing itself.
* Test of PubSub’s publish and subscribe was conducted in order to make sure that the software could not accomplish simple tasks, but also withstand trickier situations.

**Sprint 6 Retrospective**

**Date:** Nov 20, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* All user stories were accomplished for this sprint before the actual deadline.
* Design and implementation of a new PubSub agent was completed in order to connect both projects into one and show a representation of how can both of the projects be combined with different features on a real environment.
* FIU virtual box were set up to host Server in order to be used for demo presentation and final showcase.
* New approaches to an easily configurable idle session timeout was researched.
* As both projects were being combined, new requirements came up, which meant new requirements started to arise, as such they were incorporated.

**Sprint 7 Retrospective**

**Date:** Dec 04, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, Ayme Morrina, Javier Carmona, David Romero, Carlos Ruiz

**Discussed Topics:**

* For this sprint most of the time was spent finishing some documentation changes and updated all github code and readme files for all the new software implemented.
* Manual testing for the new Pubsub agent was done and all the errors found were fixed.
* Incorporating missing features to web client, this included missing behaviors and other visual touches.
* Missing setup scripts were placed into the github repository
* Tested web client

# 

# References

You must reference any work that is not your own.