**Introduction**

This document intends to describe the design and architecture of SecretRoom File Server, which is project for the final year of software development at GMIT.

For the project, we were free to choose the technologies, scope and what to develop.

The idea of a file sever came from need to easily sharing files without the need to create an account, something like an online flash drive that is plug-and-play. The part to make limited time only came after using pastebin to share code, it occurred me that there was no file serve with the possibility to let the files stay public for a limited time and then destructing itself.

**System Requirements**

Requirements to interact with system:

Internet Connection

Internet Browser such as Firefox, Google Chrome …

Requirements to run system:

Internet Connection

500mb of ram memory

1GB Hardrive

Dual Core CPU

MySql

Tomcat

Java 1.8

**Technology Used and Why**

The project was implemented in java using the spring mvc framework. The choice of using spring was made after some considerations of which framework use, after some analysis the choice came to do spring for being a framework that uses the mvc design pattern and also easily manages all project dependencies. It allowed the creation of a web app that can be easily changed and upgrade.

As I had chosen to save all information on a database instead of managing myself.

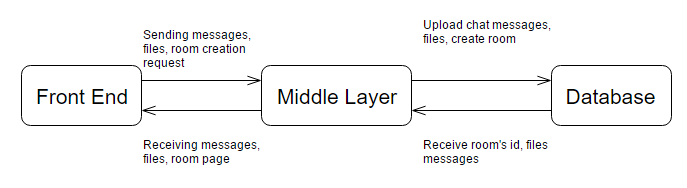
The data base management systems of choice was MySql. It is a free dbms which uses the sql syntaxes. MySql supports the creation of events that at any given time. Said possibility made the task of managing expired rooms\* easier, as it was only required to create an event and the dbms itself would remove all the expired, rooms, files and messages.

As for the front-end bootstrap was chosen. Bootstrap allows the creation of responsive and scalable pages for all devices pages. Bootstrap also offers compatibility for many different browsers, making no need for work-arounds on page code to make it look the same. Also bootstrap offers a great documentation, which makes development easier.

**Architecture of the Solution**

The system consists of the three parts. The front end which is the part visible to the user, it is made using bootstrap with html and css. This gets all the interaction of the user and sends to the middle layer.

The second part is the middle layer uses spring mvc for dependency injection. This layer manages all interactions made by the user on the front-end and treats it and send to the database, which includes request to create a room, download files, files upload and all chat’s messages. Also, this layer is responsible for getting data from the database and sending it back to the user.



The last part is the database, which is managed by the MySql dbms. On the database, all files and room information are stored, on the MySql was also created an event to delete expired automatically, said event runs every hour, it removes rooms and all data associated.

**Database Design**

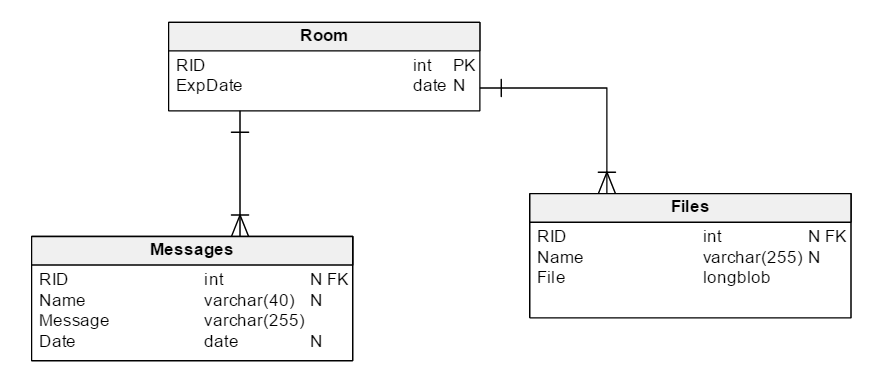
The database consists of 3 tables.

The first table is Room. Said table has 2 two fields, one of the being a primary key. The RID field stores room’s id, this is the id to recover the room and its files, as for being critical and a primary key, this field can never be set to null. The other field is ExpDate and it holds room’s expiration date, this date is checked by the event to delete room to know which rows can be removed.

The second table is Files. This tables contains a foreign key to the room table, and in case of removal of the primary key, the rows on files table that has an reference to it, will be removed too. The File field is of type longblob. Longblob was chosen as being the only type that holds a significant amount of data for a file (medium blob holds only up to 16mbs).

The Name field is a varchar that can holds up to 255 characters, 255 was decided based on the windows limitation for files name.

The third table is Messages. Out of its four fields, one is the RID. The RID is a foreign key to room table, it is set to on delete cascade for the message reason as files table. The date field is of type date, it is used to keep a timestamp for the messages. The name and message field are both varchar. The first one of size 40 and keeps name of the user that sent the message, while, the second one is of size 255 and is used to store the messages content.



**Features of the Implementation**

On this version of the app you are able to create a room a for easily sharing file and communication. Each room lives for 48 hours counting from the time of its creation.

You can also upload files which will be kept on the server the duration of the room, after that time, all files on room will be removed.

On the rooms, there is also a chat functionality in which users can send messages to anyone currently connected to the room.

All rooms are available to anyone with its link.

No information of any files, users’ ip or messages will be kept on server after room expiring.

**Limitations**

Currently there are a few limitations on the system. Due to the use of a free account on amazon’s aws service to host the application, a file size limitation to upload was set to avoid any billing, the current limit is 20mb per file.

Names of files can’t be bigger than 255 characters in length, this is a limitation set when designing the database. Also, like the file names, the usernames have limitation to its length, on this case, it can’t be bigger than 40 characters in length.

All the messages already sent on a room, can’t be retrieved after closing connection to the server, this limitation is due to the way the chat updates. The update chat function queries for messages that were saved after the function was called.

Another limitation that the system has is about the file update by the same user, only one file can be uploaded by the same user at any given time.

**Known Bugs**

If the server is restarted, all request made from pages that were open during the restart process will result in failure.

If the user sets its user name to empty(“”), it will result in all of his massages being shown out of place in the chat window.

If the file has space on its name, it causes a bug that said file can be uploaded but not downloaded, even though its name is shown on the list of available files.

**Recommendations for Future Development**

To enhance user experience on room’s page, the insertion of something to tell user the total size of room’s content and the size of each file independently, in addition to that, it would be helpful to the user to know how much time is left on his room.

Still on enhancement of user experience, the insertion of history to chat, would be of beneficial use, since it in its the current state can’t retrieve anything said previous to the connection.

For future development, the limitation of size for each file uploaded could be extended or removed. Also on files, the limitation of just one file uploaded at a time could be removed and a list of files to be uploaded and its progression shown to the user. Additionally, allow the user to resume a previous upload.

A system of monetizing could also be implemented in different forms. It could give the option to extend the current room’s lifespan after paying a certain amount of money.

**Bibliography**

<https://v4-alpha.getbootstrap.com/examples/cover/>

http://getbootstrap.com/

https://css-tricks.com/custom-scrollbars-in-webkit/

http://howtodoinjava.com/spring/spring-mvc/spring-mvc-download-file-controller-example/