



Taller de Programación II (75.52)

Trabajo Práctico

1er Cuatrimestre 2015

MensajeroO

Grupo 6

Integrantes del grupo:

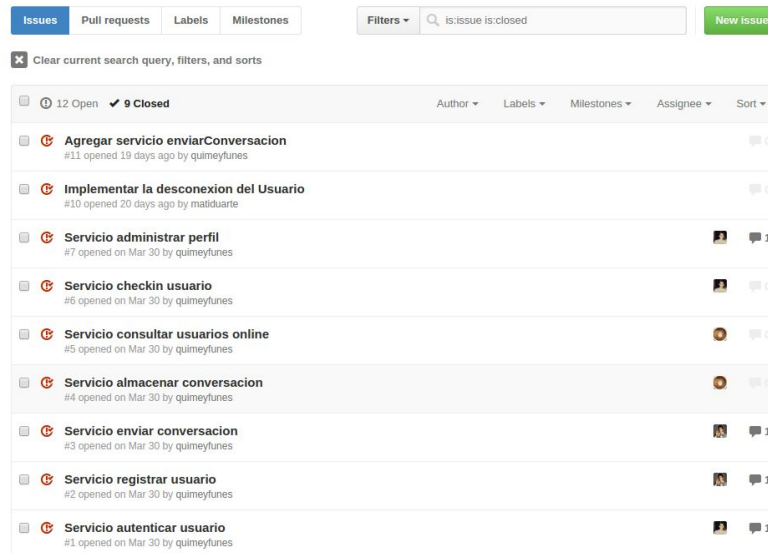
Apellido y Nombre	Padrón	E-Mail
Baracat, Juan Manuel	92775	juanmabaracat@gmail.com
Duarte, Matías Ariel	92186	duarte.mati@gmail.com
Funes Clapier, Quimey	92324	quimey.funes@gmail.com

Ayudante: Fusca, Gabriel

Repositorio: <https://github.com/matiduarte/taller2-mensajeroX>

Changelog:

Durante el desarrollo del Trabajo práctico utilizamos para gestionar y dividírnos las tareas, el sistema de **Issues** que provee **GITHub**.



Estas divisiones las fuimos haciendo en persona ya que todos los integrantes del grupo coincidimos en varias materias y por lo tanto nos podíamos juntar en la facultad. En menor medida utilizamos facebook y whatsapp para comunicarnos.

Las tareas fueron las siguientes:

- **Servicio administrar perfil** (Matías)
- **Servicio checkin usuario** (Matías)
- **Servicio consultar usuarios online** (Juan Manuel)
- **Servicio almacenar conversación** (Juan Manuel)
- **Servicio enviar conversación** (Quimey)
- **Servicio registrar usuario** (Quimey)
- **Servicio autentificar usuario** (Matías)
- **Implementar la desconexión del Usuario** (Matías)
- **Agregar servicio enviar Conversación** (Quimey)

La primera parte del desarrollo del trabajo quedó fuera del sistema de ticket ya que la división la hicimos personalmente; nos reunimos con el fin de hacer un análisis de requerimientos, y luego un diseño inicial entre todos los integrantes. Una vez terminado, dividimos la implementación en tres, y nos repartimos las tareas:

- **Cliente** (Matías)
- **Logger** (Matías)
- **Conversación** (Quimey)
- **Mensajes** (Quimey)
- **Servidor** (Juan Manuel)
- **Base de Datos** (Juan Manuel)

Checkpoint 2

División de tareas:

[Link GitHub](#)

Juan Manuel

- Implementar un nuevo estilo para los botones y agregarlo a todas las pantallas.
- Implementar -cerrar sesión- en el cliente.
- Enviar archivos entre servidor y cliente.
- Implementar servicio que actualice los datos del cliente en el servidor.
- Pantalla Acerca de.
- Pantalla configuración de perfil.
- Pantalla Ajustes.

Quimey

- Modificar el servidor para que acepte métodos de REST: GET, POST, PUT, DELETE.
- Implementación de la base de datos en el cliente.
- Implementar los Servicios.
- Implementar la Base de Datos.
- Pantalla Lista De Contactos.
- Pantalla Lista De Conversaciones.
- Implementar servidor multithread.

Matías

- Implementar servicio de localización.
- Enviar mensajes al servidor.
- Pantalla Conversación.
- Pantalla Autenticación.

API REST

Ejemplos utilizando POSTMAN

Registrar un Usuario:

url: <http://localhost:8080/usuario/>

operación: POST

KEY	VALUE
Nombre	Juan
Telefono	1112345678
Password	goku

Respuesta:

```
{
  "payload" : "Usuario registrado correctamente",
  "success" : "true"
}
```

Editar perfil:

url: <http://localhost:8080/usuario/>

operación: PUT

KEY	VALUE
Nombre	Juan
Telefono	1112345678
Password	goku
FotoDePerfil	default
Token	e255da3ce8f92a313557e734d67a9a24
EstadoDeConexion	true

respuesta

```
{
  "payload" : "Se modificaron los datos del usuario Juan correctamente. Token:",
  "success" : "true"
}
```

```
}
```

Enviar mensaje:

url: http://localhost:8080/conversacion/

operación: POST

KEY	VALUE
IdUsuarioEmisor	1112345678
Fecha	2015-06-04 15:32:55
Cuerpo	Hola, ¿cómo estás?
IdUsuarioReceptor	1111223344
Token	g345da3ce8f92a313557e734d67a9a24

respuesta:

```
{
  "payload" : "188c3d61cf52bda68a52f7afe6070727",
  "success" : "true"
}
```

Consultar usuario:

url: http://localhost:8080/usuario/1112345678

operación: GET

respuesta:

```
{
  "payload" : "{\n  \"EstadoDeConexion\" : \"true\",\n  \"FotoDePerfil\" : \"default\",\n  \"Nombre\" : \"Juan\",\n  \"Password\" : \"goku\",\n  \"Token\" :\n  \"b1d3cc2ebd0d1ebfcdeb16de173e99b6\",\n  \"idUsuario\" :\n  \"0a0625f4dba80e60e7bb4e37114f744f\"\n}\n",
  "success" : "true"
}
```

Obtener Conversaciones:

url: http://localhost:8080/usuarioConversacion/1112345678

operación: GET

respuesta:

```
{
```

```
"payload" : "{\n  \"conversaciones\" : [\n    {\n      \"id\" :  
\"0a0625f4dba80e60e7bb4e37114f744f-8ca745744c1910342bb2441b61951494\",  
\"ultimoMensaje\" : \"Todo bien. ¿Vos?\",  
\"usuarioFotoDePerfil\" : \"default\",  
\"usuarioNombre\" : \"Pepe\",  
\"usuarioTelefono\" : \"1111223344\"  
    }  
  ],  
  \"success\" : \"true\"  
}
```

Instalación

Desde la línea de comando descargue el código e instale su propia copia:

```
$ git clone https://github.com/matiduarte/taller2-mensajeroX.git  
$ cd taller2-mensajero/Servidor
```

Compilación

En la carpeta del servidor escribir:

```
$ mkdir build  
$ cd /build  
$ cmake ..  
$ make
```

Una vez finalizado, ejecutar el servidor con `./Servidor`

Entrega Final


[Milestone Entrega Final](#)

División de tareas:

Juan Manuel

- [CLI][SVR]Implementar localización.
- [CLI] [SVR]Documentación JavaDoc.
- [CLI][SVR]Completar informe.

Quimey

- [CLI][SVR] Documentación sphinx.
- [CLI]Cierre de sesión .
- [CLI][SVR]Seguridad a través del token.
- [CLI][SVR]Servicio enviar lista difusión.

Matías

- [CLI] Manual de usuario.
- [CLI]Agregar campo para ingresar ip en la pantalla de logueo.
- [CLI]Pantalla información de contacto.

API REST

Se agregó la funcionalidad de check-in:

Hacer Check-in:

url: <http://localhost:8080/usuario/checkin/>

operación: PUT

KEY	VALUE
Telefono	1112345678
latitud	-34.617834
longitud	-58.368113
Token	e255da3ce8f92a313557e734d67a9a24

respuesta

```
{  
  "payload" : "Facultad de Ingeniería",  
  "success" : "true"  
}
```