Комитет по образованию Правительства Санкт-Петербурга  
Государственное бюджетное учреждение  
среднего профессионального образования  
«Колледж информационных технологий»

МДК 03.01

«Технология разработки программного обеспечения»

**отчёт**

по практической работе №12

Выполнил студент 474 гр.:

Черных Артём

Преподаватель:

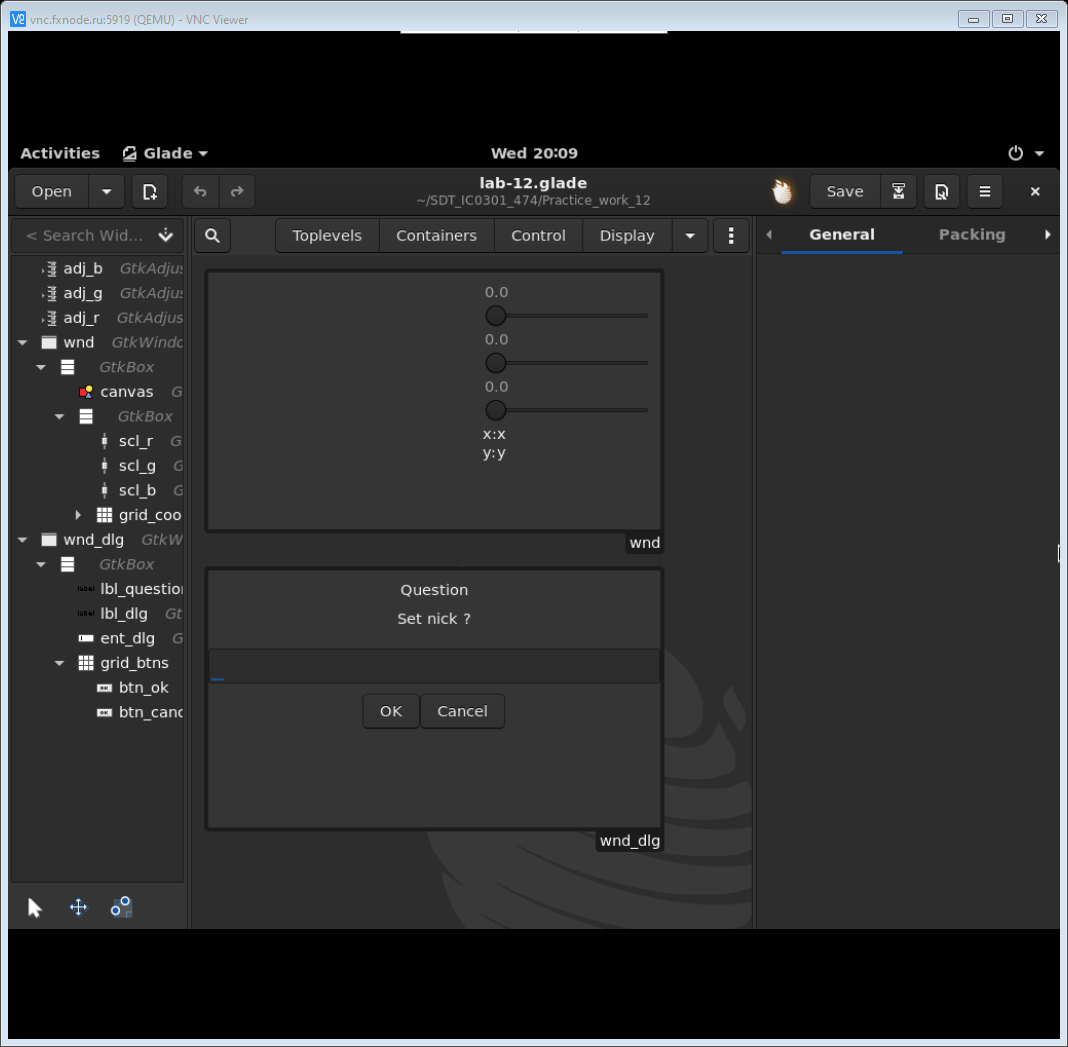
Фомин А.В.

Санкт-Петербург 2020

**Использование сервера redis для обмена данными между клиентами**

**Основное задание**

1. На рисунке 1 интерфейс приложения.



Черных Артём

Рисунок 1 – Файл «lab-12.glade»

1. Создано сетевое приложение, отображающее координаты мыши каждого подключённого клиента в реальном времени, его имя (идентификатор) и цвет.

В функции GtkDrawingArea:on\_draw() (ui.canvas:on\_draw()) получение из хранилища список подключённых клиентов и извлечение значения ключей для каждого из них, отрисовывая заданного цвета в указанных ключах координатах.

Идентификатор генерируются при запросе от пользователя ввести ник(nick), в случаи отказа вводить ник, он генерируется случайно.

Файл «lab-12.lua»:

lgi = require 'lgi'

redis = require 'redis'

gtk = lgi.Gtk

glib = lgi.GLib

gtk.init()

bld = gtk.Builder()

bld:add\_from\_file('lab-12.glade')

ui = bld.objects

id = ''

btn\_dlg = false

x = 0

y = 0

btn = 0

sr = 0

sg = 0

sb = 0

function wnd\_dlg\_close()

ui.wnd\_dlg:hide()

ui.wnd.on\_destroy = gtk.main\_quit

main\_part()

ui.wnd:show\_all()

end

function ui.btn\_ok:on\_clicked()

id = ui.ent\_dlg.text

btn\_dlg = true

wnd\_dlg\_close()

end

function ui.btn\_cancel:on\_clicked()

wnd\_dlg\_close()

end

ui.wnd\_dlg.title = 'lab-12'

ui.wnd.title = 'lab-12'

ui.wnd\_dlg.on\_destroy = gtk.main\_quit

ui.wnd\_dlg:show\_all()

function main\_part()

function ui.canvas:on\_button\_press\_event(ev)

print('press')

btn = 1

ui.canvas:queue\_draw()

update\_keys()

end

function ui.canvas:on\_button\_release\_event(ev)

print('release')

btn = 0

ui.canvas:queue\_draw()

update\_keys()

end

function ui.canvas:on\_motion\_notify\_event(ev)

x = ev.x

y = ev.y

print(x, y)

ui.coordinate\_x.label = x

ui.coordinate\_y.label = y

ui.canvas:queue\_draw()

update\_keys()

end

function ui.scl\_r:on\_value\_changed()

sr = ui.scl\_r:get\_value()

ui.canvas:queue\_draw()

update\_keys()

end

function ui.scl\_g:on\_value\_changed()

sg = ui.scl\_g:get\_value()

ui.canvas:queue\_draw()

update\_keys()

end

function ui.scl\_b:on\_value\_changed()

sb = ui.scl\_b:get\_value()

ui.canvas:queue\_draw()

update\_keys()

end

function ui.canvas:on\_draw(cr)

cr:set\_source\_rgb(1, 1, 1, 1)

cr:paint()

local cid = cli:lrange('clients', 0, -1)-- get client list ("title", log, trace statement)

for i = 1, #cid do-- for each client

--cname = tonumber(cli:get(cli[i] .. name))

cx = tonumber(cli:get(cid[i] .. '-x'))

cy = tonumber(cli:get(cid[i] .. '-y'))

cbtn = tonumber(cli:get(cid[i] .. '-btn'))

csr = tonumber(cli:get(cid[i] .. '-sr'))

csg = tonumber(cli:get(cid[i] .. '-sg'))

csb = tonumber(cli:get(cid[i] .. '-sb'))

cr:set\_source\_rgb(csr, csg, csb, 1)

if cbtn == 0 then

cr:rectangle(cx - 5, cy - 5, 10, 10)

else

cr:rectangle(cx - 10, cy - 10, 20, 20)

end

cr:fill()

cr:move\_to(cx, cy - 10)

--cr:show\_text(cid[i] .. ': '..name)

cr:show\_text(cid[i])

end

end

ui.wnd.on\_destroy = gtk.main\_quit

ui.wnd:show\_all()

--connect to redis server

cli = redis.connect('redis.fxnode.ru', 6379)

--statement on nick or indentificator

--output current coordinate

math.randomseed(os.time())--to random not to repiat, set from current time

--generate random client id

if btn\_dlg == false then

--id = ''

for i = 1, 8 do--8 random symbols

local j = math.random(1, 16)

id = id .. string.sub('0123456789abcdef', j, j)--take symbol

end

end

print('client id', id)

function update\_keys()-- update keys

cli:set(id .. '-ts', os.time())

cli:set(id .. '-x', x)

cli:set(id .. '-y', y)

cli:set(id .. '-btn', btn)

cli:set(id .. '-sr', sr)

cli:set(id .. '-sg', sg)

cli:set(id .. '-sb', sb)

glib.timeout\_add(glib.PRIORITY\_DEFAULT, 1000, update\_keys)--set timer 1000

end

function update\_view()--put some data in window

ui.canvas:queue\_draw()

glib.timeout\_add(glib.PRIORITY\_DEFAULT, 25, update\_view)-- 25

end

glib.timeout\_add(glib.PRIORITY\_DEFAULT, 25, update\_view)--25

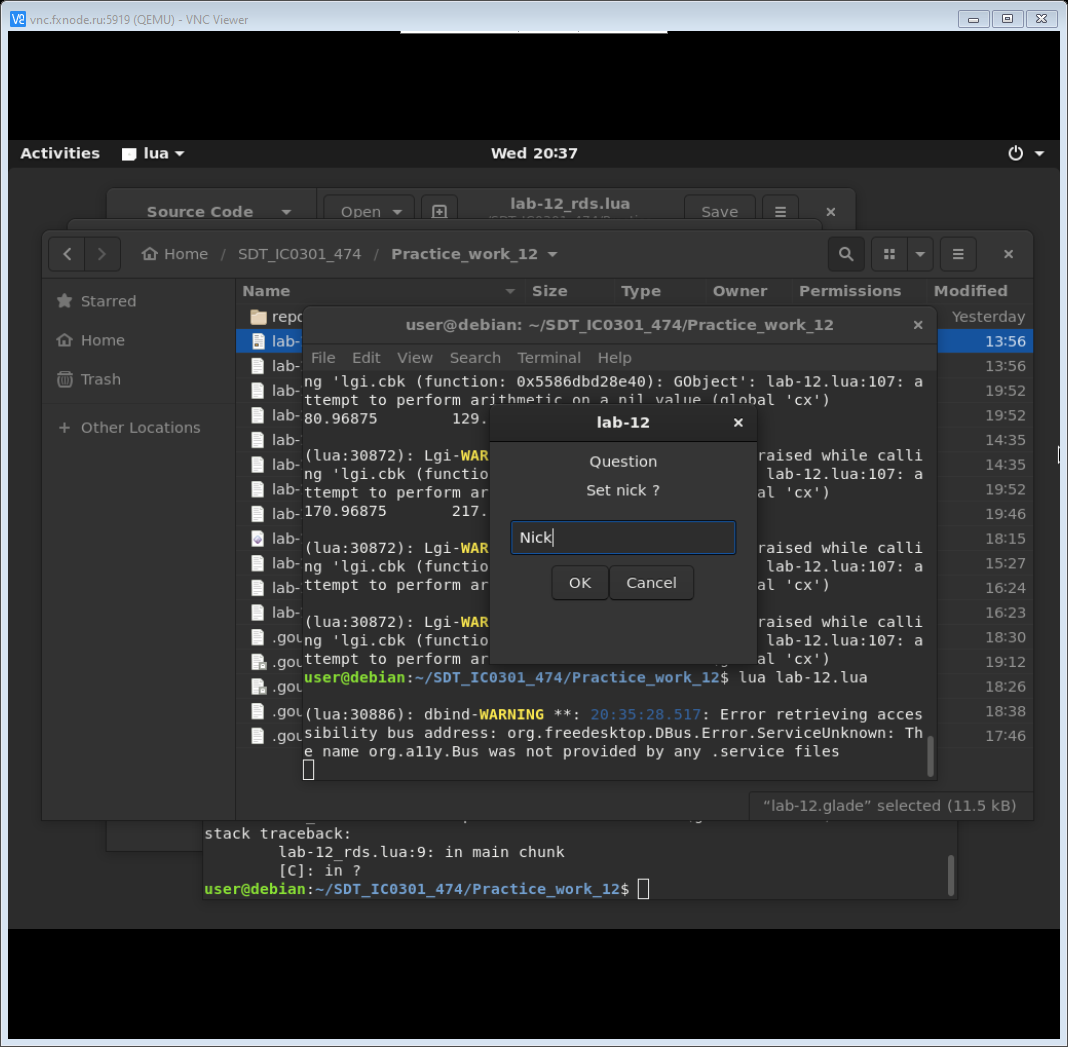
update\_keys()-- initialize keys

cli:lpush('clients', id)-- add client to clients list

end

gtk.main()

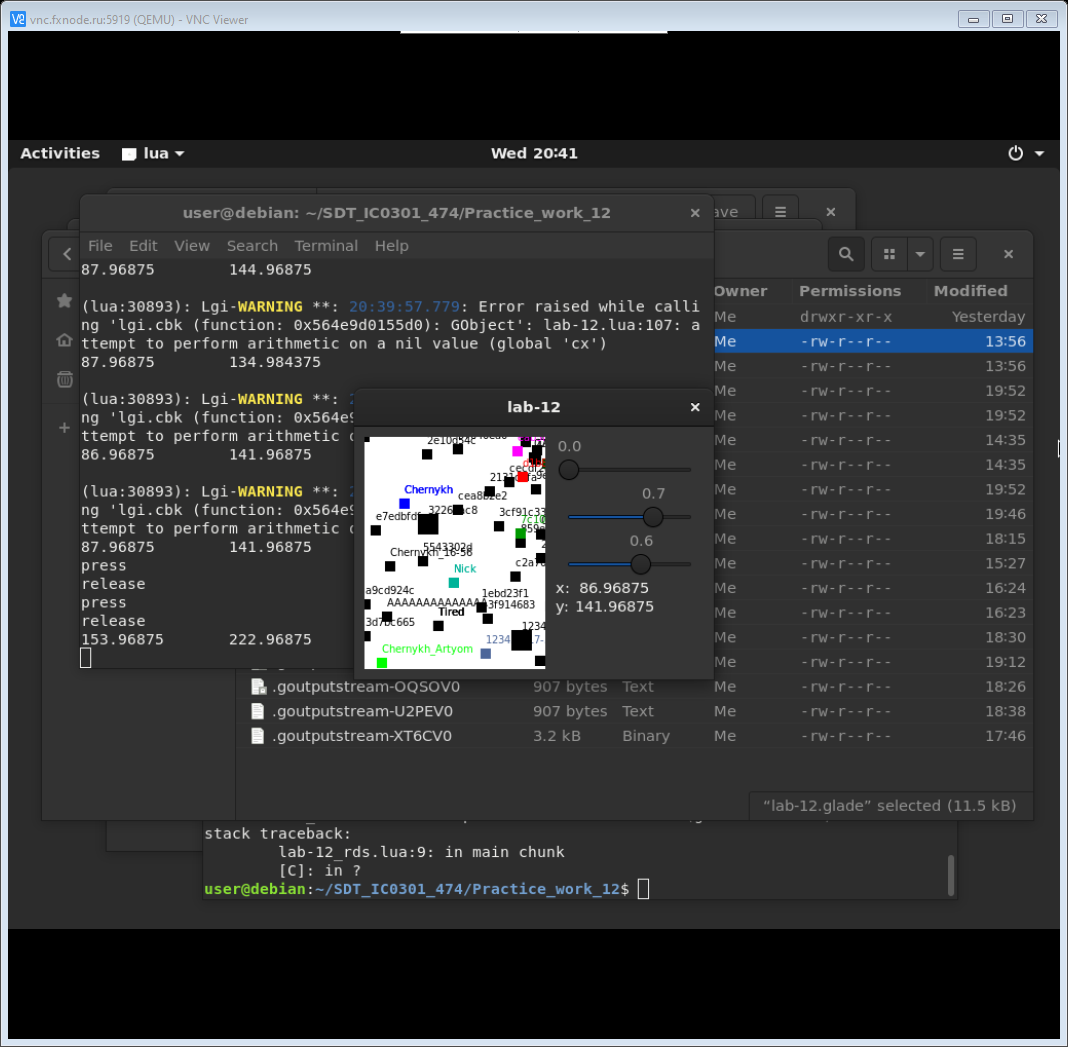
1. Пример работы приложения, если пользователь введёт ник рисунках 2, 3.



Черных Артём

Рисунок 2 – Пользователь согласился вводить свой ник

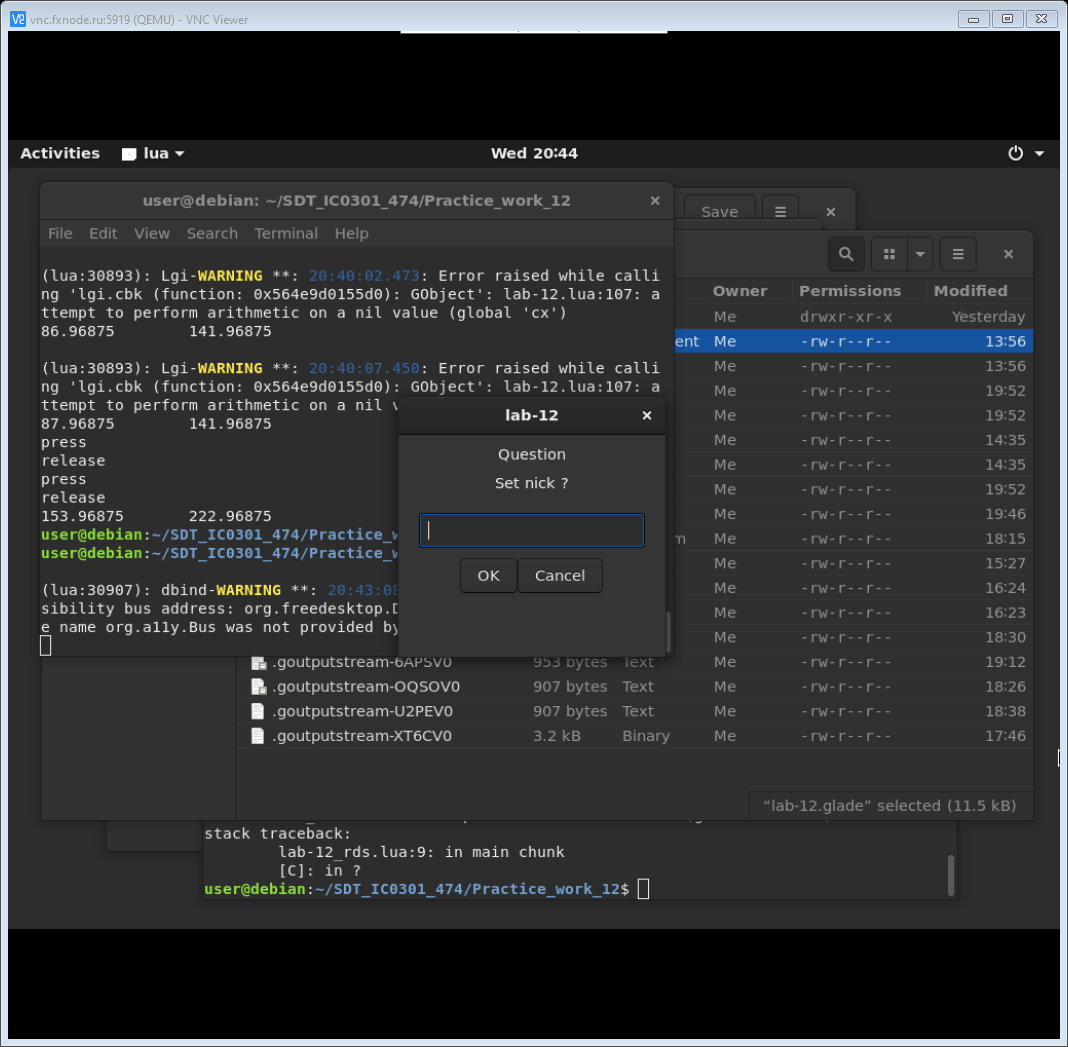
Основная часть приложения:



Черных Артём

Рисунок 3 – Ник пользователя

1. Пример работы приложения, если пользователь не вводит ник на рисунках 4, 5.

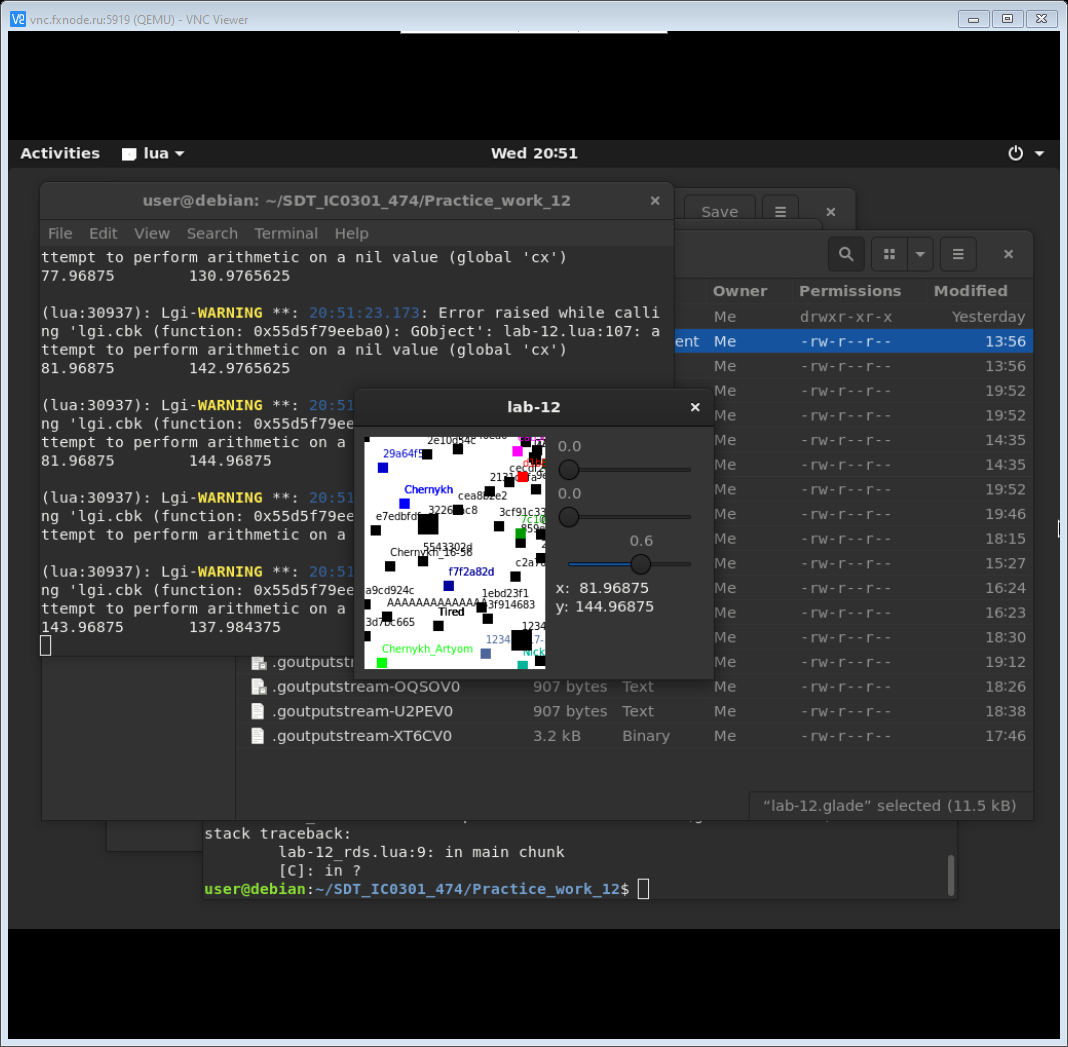


Черных Артём

Черных Артём

Рисунок 4 – Пользователь отказался вводить свой ник

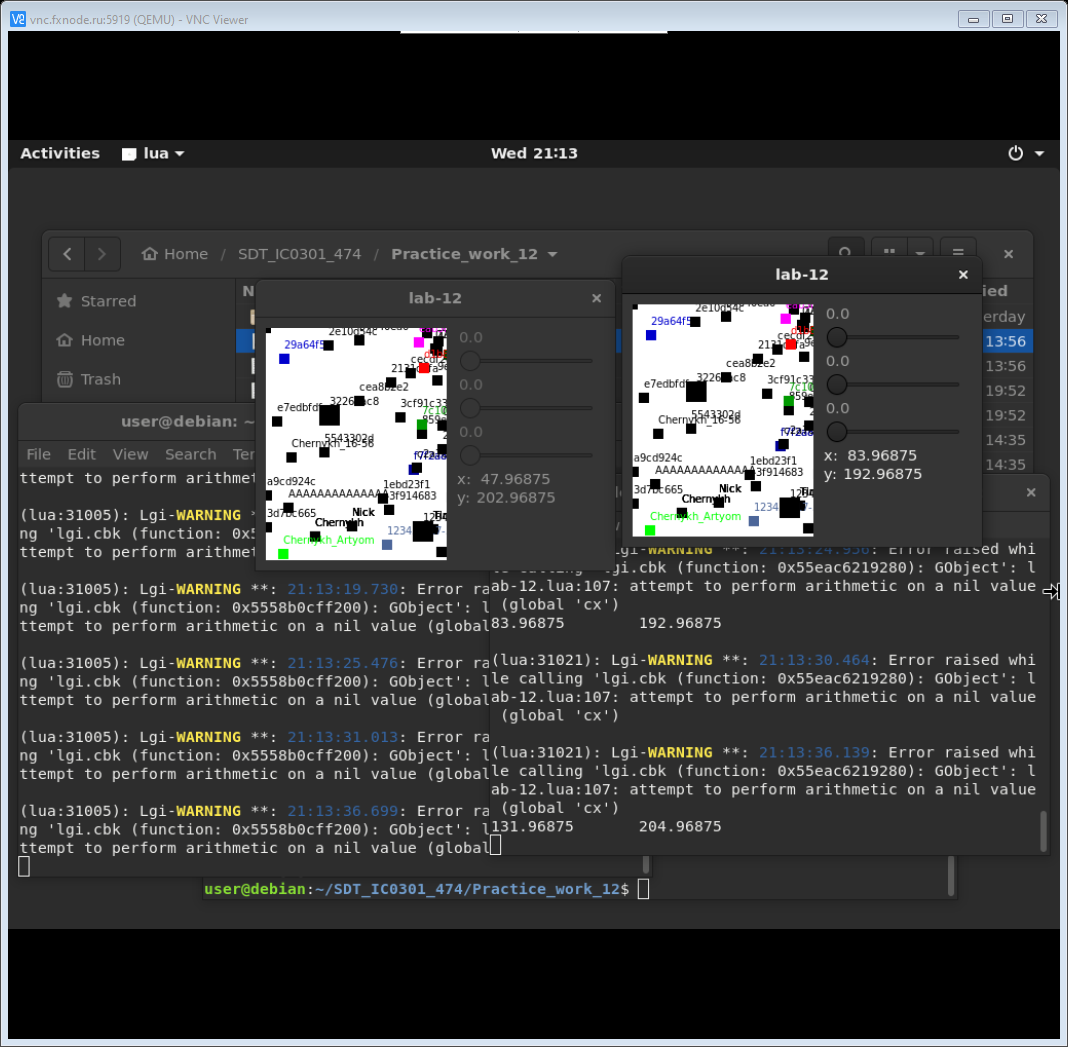
Основная часть приложения:



Черных Артём

Рисунок 5 – Случайный идентификатор

1. Если запущено одновременно несколько пользователей на рисунке 6.



Черных Артём

Рисунок 6 – Несколько пользователей

**Дополнительно**

1. Процедура выполняется на сервере Redis. Которая удаляет ключи пользователей из хранилища не менявших значения своих ключей достаточно долгое время, таким образом исчезнет информация как из базы данных, так и у других подключённых клиентов. Выполнение приложения с помощью функции redis.call( ) для доступа к хранилищу ключей.

Файл «lab-12\_rds.lua» (К сожалению приложени это не работает, хотя первая строчка в которой ошибка работает в Redis)

local ttl = 3-- client timeout

--local cli = redis.call('LRANGE', 'clients', 0, -1)-- get a list of clients

local cli = redis.call('LRANGE', 'clients', 0, -1)

local now = tonumber(redis.call('TIME')[1])-- get current time

for i = 1, #cli do

local ts = redis.call('GET', cli[i] .. '-ts')

local rem = true

if ts ~= false then-- key exists

if now < ts + ttl then rem = false end-- keep alive

end

if rem == true then

redis.call('LREM', 'clients', 0, cli[i])-- remove from cliets list

redis.call('DEL', cli[i] .. '-ts')-- remove keys

redis.call('DEL', cli[i] .. '-x')

redis.call('DEL', cli[i] .. '-y')

redis.call('DEL', cli[i] .. '-btn')

redis.call('DEL', cli[i] .. '-sr')

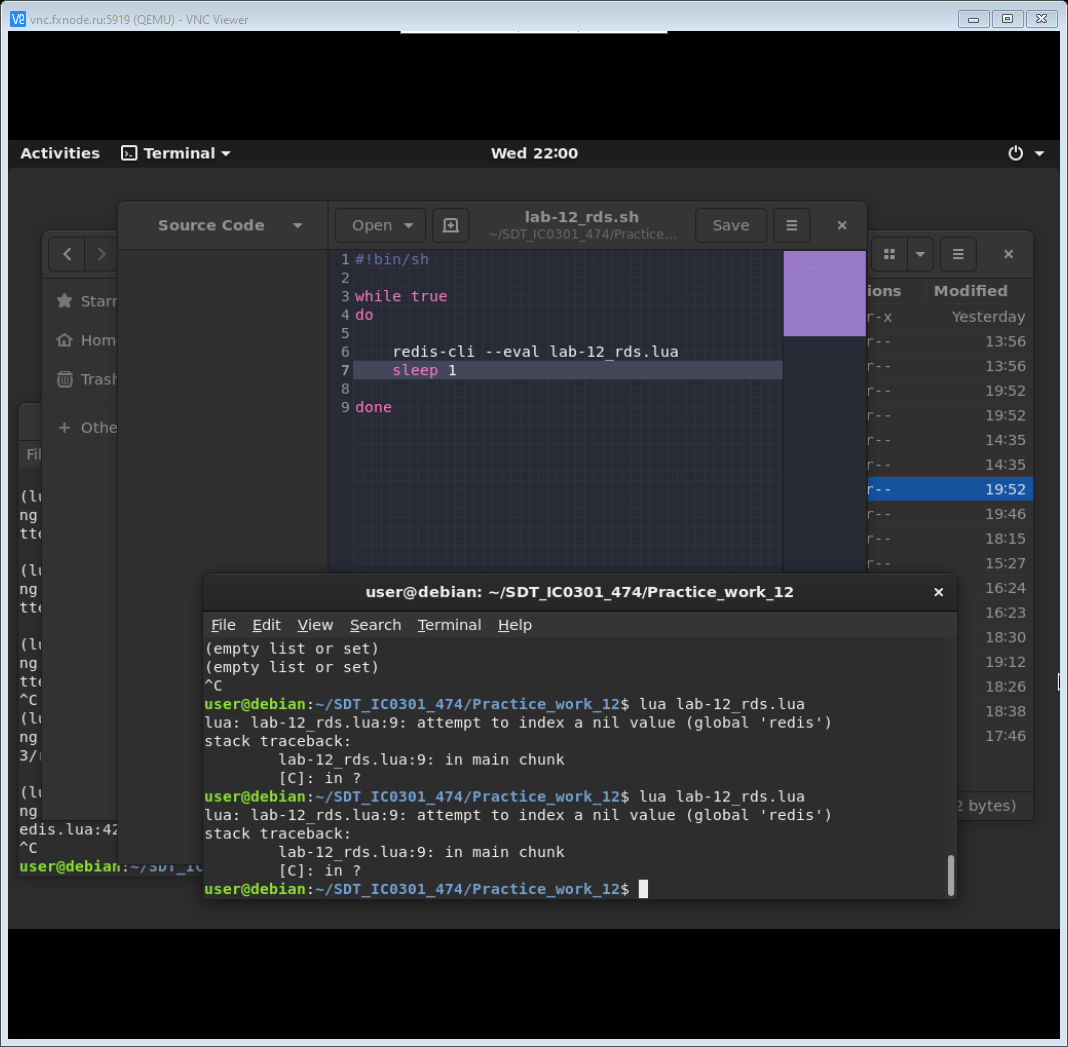
redis.call('DEL', cli[i] .. '-sg')

redis.call('DEL', cli[i] .. '-sb')

end

end

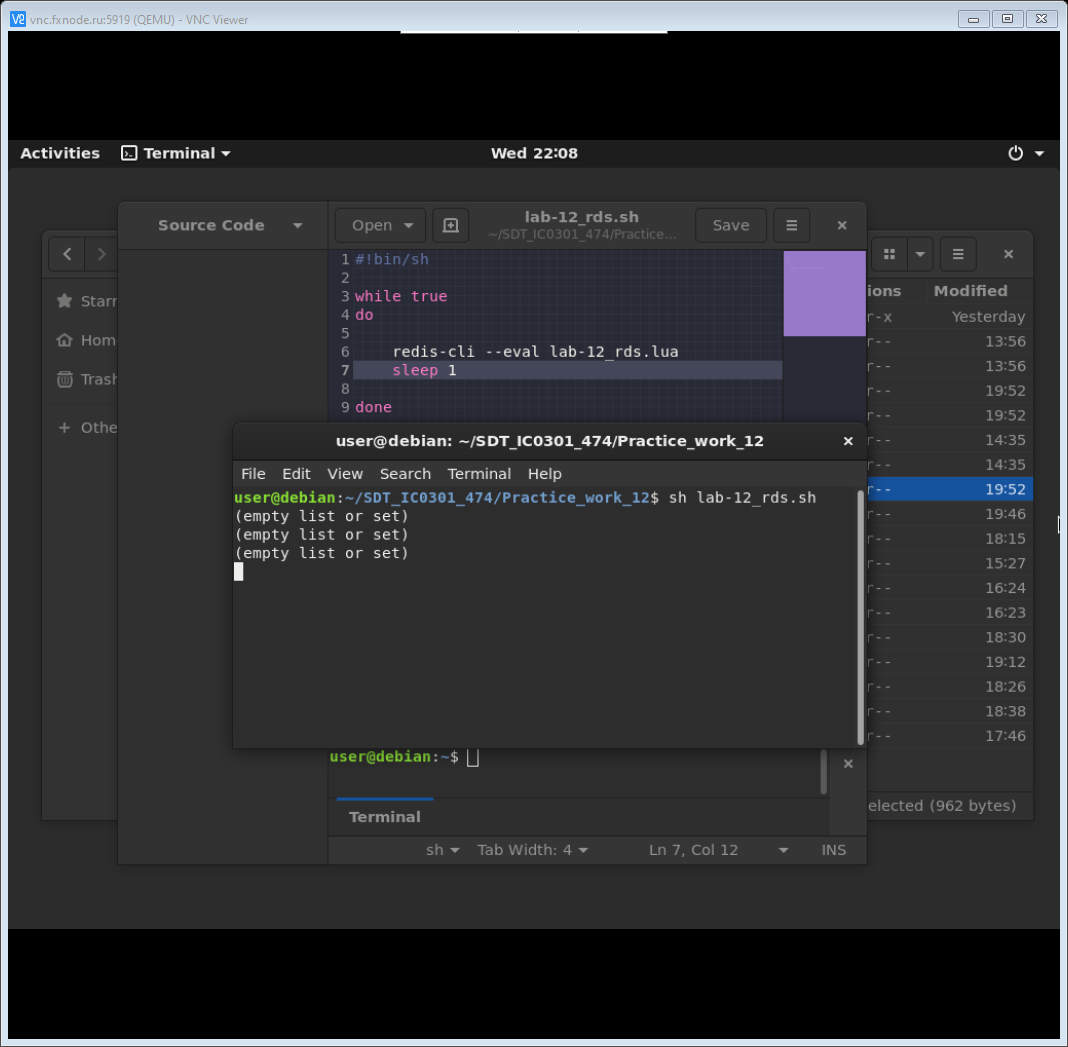
return cliПример работы процедуры (Ошибка при работе процедуры) на рисунке 7.



Черных Артём

Рисунок 7 – Запуск файла «lab-12\_rds.lua»

1. Переодический запуск сценария внутри сервера Redis следующийм shell-скриптом на рисунке 8.



Черных Артём

Рисунок 8 – Сценарий «lab-12\_rds.sh» успешно запускается