```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QDesktopWidget>
#include <QScreen>
#include <QMessageBox>
#include <QMetaEnum>
#include "CustomPlotZoom.h"

MainWindow::MainWindow(QWidget *parent):
  QMainWindow(parent),
  ui(new Ui::MainWindow),
  maxHeightFromInterval(0)
{
  ui->setupUi(this);
  setGeometry(210, 250, 800, 610);
  zoom=new CustomPlotZoom(ui->customPlot);
  click = new SelectEl(ui->customPlot, columnData);

  //Заголовки столбцов
  QStringList horizontalHeader;
  horizontalHeader.append("date");
  horizontalHeader.append("time");
  horizontalHeader.append("username");
  horizontalHeader.append("databasename");
  horizontalHeader.append("host");
  horizontalHeader.append("pid");
  horizontalHeader.append("type");
  horizontalHeader.append("durtime");
  horizontalHeader.append("statement");

  ui->tableWidget->setColumnCount(horizontalHeader.size());

  ui->tableWidget->setHorizontalHeaderLabels(horizontalHeader);

 // const int tab_ab_row = ui->tableWidget->rowCount();
  //вставляем строку
 // ui->tableWidget->insertRow(tab_ab_row);
 // ui->tableWidget->setItem(tab_ab_row,0,new QTableWidgetItem(QString("w")));
 // ui->tableWidget->setItem(tab_ab_row,1,new QTableWidgetItem(QString("waref")));

  ui->tableWidget->resizeRowsToContents();
  //ui->tableWidget->resizeColumnsToContents();
  ui->tableWidget->horizontalHeader()->setDefaultSectionSize((this-
>frameGeometry().width()-4)/horizontalHeader.size());

//  ui->tableWidget->hide();

  File = new QFile(LOG_FILE);
  if (!File->open(QIODevice::ReadOnly | QIODevice::Text))
          return;
  filein = new QTextStream (File);
  setupStatistic(ui->customPlot);

  setWindowTitle("Statistic");

  //QTimer::singleShot(4000, this, SLOT(screenShot()));
}

char MainWindow::StartInitTable()
{
//    QTextStream *f=filein;
    File->close();
    File->open(QIODevice::ReadOnly | QIODevice::Text);
    filein->seek(lastpos);
//    qWarning()<<filein->pos()<<endl;
    QString json=filein->readLine();
//    qWarning()<<json<<filein->pos()<<endl;
    if(json.isEmpty())
```

```
        return 0;
    while(!json.isEmpty())
    {
      QJsonDocument jsonDoc = QJsonDocument::fromJson(json.toUtf8());
      if(jsonDoc.isNull())
          return 0;
      QJsonObject jsonObject = jsonDoc.object();
      BigTable.append(jsonObject.toVariantMap());

      json=filein->readLine();
    }
    lastpos=filein->pos();
    return 1;
}
void MainWindow::FillBars(int i)
{
    for (int j=i;i<BigTable.size();){
      int countInOne=0;
      QDateTime Time;
      if(i<BigTable.size())
        Time=QDateTime(BigTable[i]["messInfo"].toMap()["date"].toDate(),BigTable[i]
["messInfo"].toMap()["time"].toTime());
      for(;i<BigTable.size()&&Time==QDateTime(BigTable[i]["messInfo"].toMap()
["date"].toDate(),BigTable[i]["messInfo"].toMap()["time"].toTime());i++,countInOne++);

      for(;j<i;j++)
          BigTable[j]["column"]=columnData.size();
      ticks<<ticks.size()+0.5;
      labels<<BigTable[i-1]["messInfo"].toMap()["time"].toString();
      columnData<<countInOne;
      if(countInOne>maxHeightFromInterval)
        maxHeightFromInterval=countInOne;
    }
    ui->customPlot->yAxis->setRange(0, maxHeightFromInterval+maxHeightFromInterval/20);
    ui->customPlot->xAxis->setRange(0, columnData.size());
    ui->customPlot->xAxis->setTickVector(ticks);
    ui->customPlot->xAxis->setTickVectorLabels(labels);
    column->setData(ticks, columnData);
}

void MainWindow::setupStatistic(QCustomPlot *customPlot)
{
#if QT_VERSION < QT_VERSION_CHECK(4, 7, 0)
  QMessageBox::critical(this, "", "You're using Qt < 4.7, the realtime data demo needs
functions that are available with Qt 4.7 to work properly");
#endif

  lastpos=0;
  if(StartInitTable()==0)
  {
    QMessageBox::critical(this,"","bad json");
    return;
  }

  column = new QCPBars(customPlot->xAxis, customPlot->yAxis);
  customPlot->addPlottable(column);
  QPen pen;
  pen.setWidthF(1);
  column->setName("column");
  pen.setColor(QColor(255, 131, 0, 80));
  column->setPen(pen);
  column->setBrush(QColor(255, 131, 0, 80));
  column->setWidth(1);

  /*
    // setup legend:
    customPlot->legend->setVisible(true);
    customPlot->axisRect()->insetLayout()->setInsetAlignment(0, Qt::AlignTop|
Qt::AlignHCenter);
```

```
    customPlot->legend->setBrush(QColor(255, 255, 255, 200));
    QPen legendPen;
    legendPen.setColor(QColor(130, 130, 130, 200));
    customPlot->legend->setBorderPen(legendPen);
    QFont legendFont = font();
    legendFont.setPointSize(10);
    customPlot->legend->setFont(legendFont);
  */
  // prepare x axis:
  customPlot->xAxis->setAutoTicks(false);
  customPlot->xAxis->setAutoTickLabels(false);
  customPlot->xAxis->setTickLabelRotation(80);
  customPlot->xAxis->setSubTickCount(0);
  customPlot->xAxis->setTickLength(0, 0);
  customPlot->xAxis->grid()->setVisible(true);
  customPlot->xAxis->setLabel("Time");//TODO

  // prepare y axis:
  customPlot->yAxis->setAutoTickStep(true);
  customPlot->yAxis->setPadding(5); // a bit more space to the left border
  customPlot->yAxis->setLabel("Count");//TODO
  customPlot->yAxis->grid()->setSubGridVisible(true);
  QPen gridPen;
  gridPen.setStyle(Qt::SolidLine);
  gridPen.setColor(QColor(0, 0, 0, 25));
  customPlot->yAxis->grid()->setPen(gridPen);
  gridPen.setStyle(Qt::DotLine);
  customPlot->yAxis->grid()->setSubGridPen(gridPen);

  //must have two changes elements to correct work
  FillBars(0);

  //qWarning();QCPBarData a=column->data()->value(0.5);
  ui->horizontalSlider->setMaximum(columnData.size()-1);
/*
  ui->horizontalScrollBar->setValue(10);
//  ui->horizontalScrollBar->setMaximum(100);
  ui->horizontalScrollBar->setMinimum(ui->customPlot->xAxis->range().center());
  ui->horizontalSlider->setValue(customPlot->xAxis->range().size());
  */
  connect(ui->horizontalScrollBar, SIGNAL(valueChanged(int)), this,
SLOT(horzScrollBarChanged(int)));
//  connect(customPlot->xAxis, SIGNAL(rangeChanged(QCPRange)), this,
SLOT(xAxisChanged(QCPRange)));

  connect(ui->horizontalSlider, SIGNAL(valueChanged(int)), this,
SLOT(horzSliderChanged(int)));

//  QMessageBox::warning(this,"",QString::number(ui->customPlot->xAxis->range().size())+"
"+QString::number(ui->customPlot->xAxis->range().center()));


  //  qWarning()<<BigTable[0]["messInfo"].toMap()["date"];
  /*
    for (float i=0.5;i<BigTable.size()+0.5;i++){
      ticks<<i;
      labels<<BigTable[i-0.5]["messLog"].toMap()["type"].toString();
      columnData<<value0;
    }
    ui->customPlot->xAxis->setTickVector(ticks);
    ui->customPlot->xAxis->setTickVectorLabels(labels);
    column->setData(ticks, columnData);

    qWarning()<<BigTable.size()/ui->customPlot->xAxis->range().size();
    ui->horizontalSlider->setMaximum(BigTable.size()/ui->customPlot->xAxis-
>range().size());
  */
//  customPlot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
```

```
  connect(ui->customPlot, SIGNAL(mousePress(QMouseEvent*)), this->zoom,
SLOT(mousePressEvent(QMouseEvent*)));
  connect(ui->customPlot, SIGNAL(mouseRelease(QMouseEvent*)), this->zoom,
SLOT(mouseReleaseEvent(QMouseEvent*)));
  connect(ui->customPlot, SIGNAL(mouseMove(QMouseEvent*)), this->zoom,
SLOT(mouseMoveEvent(QMouseEvent*)));

  connect(ui->customPlot, SIGNAL(mousePress(QMouseEvent*)), this->click,
SLOT(mousePressEvent(QMouseEvent*)));
  connect(ui->customPlot, SIGNAL(mousePress(QMouseEvent*)), this,
SLOT(mousePressEvent(QMouseEvent*)));
//  return;
  connect(&dataTimer, SIGNAL(timeout()), this, SLOT(UpdateSlot()));
  dataTimer.start(1000); // Interval 0 means to refresh as fast as possible
}

void MainWindow::horzScrollBarChanged(int value)//TODO int may be not enought//
{
  ui->customPlot->xAxis->setRange(value, columnData.size()-ui->horizontalSlider->value(),
Qt::AlignLeft);
  maxHeightFromInterval=0;
  for(int i=value;i<value+ui->customPlot->xAxis->range().size();i++)//TODO too more time
      if(columnData[i]>maxHeightFromInterval)
          maxHeightFromInterval=columnData[i];
  ui->customPlot->yAxis->setRange(0,maxHeightFromInterval+maxHeightFromInterval/20);
//  qWarning()<<maxHeightFromInterval<<ui->customPlot->xAxis->range().size();
  ui->customPlot->replot();
}

void MainWindow::horzSliderChanged(int value)
{
  ui->customPlot->xAxis->setRange(ui->horizontalScrollBar->value(), columnData.size()-
value, Qt::AlignLeft);
  maxHeightFromInterval=0;
  for(int i=ui->horizontalScrollBar->value();i<ui->horizontalScrollBar->value()+ui-
>customPlot->xAxis->range().size()-1;i++)
      if(columnData[i]>maxHeightFromInterval)
          maxHeightFromInterval=columnData[i];
  ui->customPlot->yAxis->setRange(0,maxHeightFromInterval+maxHeightFromInterval/20);
  ui->customPlot->replot();
  ui->horizontalScrollBar->setMaximum(value);
}

void MainWindow::UpdateSlot()
{
  // calculate two new data points:
#if QT_VERSION < QT_VERSION_CHECK(4, 7, 0)
  double key = 0;
#else
  double key = QDateTime::currentDateTime().toMSecsSinceEpoch()/1000.0;
#endif
  static double lastPointKey = 0;
  if (key-lastPointKey > 0.01)
  {
    int lastsize=BigTable.size();
    if(StartInitTable()==0)
        return;
    FillBars(lastsize);
    /*
    for(int i=lastsize;i<BigTable.size();i++){
        qWarning()<<BigTable[i];
        ticks<<(0.5+i);
        labels<<BigTable[i]["messInfo"].toMap()["time"].toString();
        ui->customPlot->xAxis->setTickVector(ticks);
        ui->customPlot->xAxis->setTickVectorLabels(labels);
        columnData<<value0;
        column->setData(ticks, columnData);
        BigTable[i]["column"]=columnData.size();
```

```
        lastPointKey = key;
        if(lastsize>ui->customPlot->xAxis->range().size()&&ui->horizontalScrollBar-
>maximum()>ui->horizontalScrollBar->minimum())
            ui->horizontalScrollBar->setMaximum(lastsize-ui->customPlot->xAxis-
>range().size());
    }
    */
  }
  ui->customPlot->replot();
}

MainWindow::~MainWindow()
{
  delete filein;
  delete File;
  delete ui;
}

void MainWindow::screenShot()
{
#if QT_VERSION < QT_VERSION_CHECK(5, 0, 0)
  QPixmap pm = QPixmap::grabWindow(qApp->desktop()->winId(), this->x()+2, this->y()+2,
this->frameGeometry().width()-4, this->frameGeometry().height()-4);
#else
  QPixmap pm = qApp->primaryScreen()->grabWindow(qApp->desktop()->winId(), this->x()+2,
this->y()+2, this->frameGeometry().width()-4, this->frameGeometry().height()-4);
#endif
  QString fileName = "1.png";
  fileName.replace(" ", "");
  pm.save(fileName);
//  qApp->quit();
}

void MainWindow::mousePressEvent(QMouseEvent * event)
{
    if (event->button() == Qt::LeftButton&&
            event->pos().x()>ui->customPlot->xAxis->coordToPixel(0)&&
            event->pos().x()<ui->customPlot->xAxis->coordToPixel(ui->customPlot->xAxis-
>range().size())&&
            event->pos().y()<ui->customPlot->yAxis->coordToPixel(0)&&
            event->pos().y()>ui->customPlot->yAxis->coordToPixel(columnData[(int)ui-
>customPlot->xAxis->pixelToCoord(event->pos().x())])
            )
    {
        int el = (int)ui->customPlot->xAxis->pixelToCoord(event->pos().x());

        for (;ui->tableWidget->rowCount()>0;){
            ui->tableWidget->removeRow(0);
        }
        if(click->Selected==-1){
//            ui->tableWidget->hide();
            return;
        }
//        ui->tableWidget->show();
        for(int i=0;i<BigTable.size();i++)//TODO
            if(BigTable[i]["column"]==el){
                const int tab_ab_row=ui->tableWidget->rowCount();
                ui->tableWidget->insertRow(tab_ab_row);
                ui->tableWidget->setItem(tab_ab_row,0,new QTableWidgetItem(BigTable[i]
["messInfo"].toMap()["date"].toString()));
                ui->tableWidget->setItem(tab_ab_row,1,new QTableWidgetItem(BigTable[i]
["messInfo"].toMap()["time"].toString()));
                ui->tableWidget->setItem(tab_ab_row,2,new QTableWidgetItem(BigTable[i]
["messInfo"].toMap()["username"].toString()));
                ui->tableWidget->setItem(tab_ab_row,3,new QTableWidgetItem(BigTable[i]
["messInfo"].toMap()["databasename"].toString()));
                ui->tableWidget->setItem(tab_ab_row,4,new QTableWidgetItem(BigTable[i]
["messInfo"].toMap()["host"].toString()));
```

```
                ui->tableWidget->setItem(tab_ab_row,5,new QTableWidgetItem(BigTable[i]
["messInfo"].toMap()["pid"].toString()));
                ui->tableWidget->setItem(tab_ab_row,6,new QTableWidgetItem(BigTable[i]
["messLog"].toMap()["type"].toString()));
                ui->tableWidget->setItem(tab_ab_row,7,new QTableWidgetItem(BigTable[i]
["messLog"].toMap()["duration"].toMap()["time"].toString()));
                ui->tableWidget->setItem(tab_ab_row,8,new QTableWidgetItem(BigTable[i]
["messLog"].toMap()["duration"].toMap()["statement"].toString()));
            }
    }
}
```