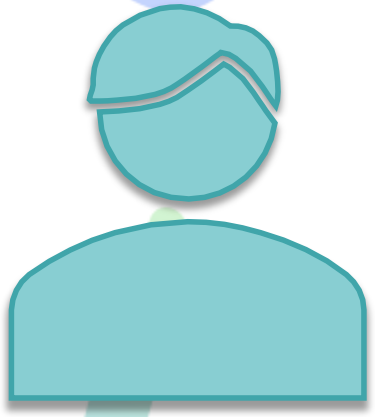
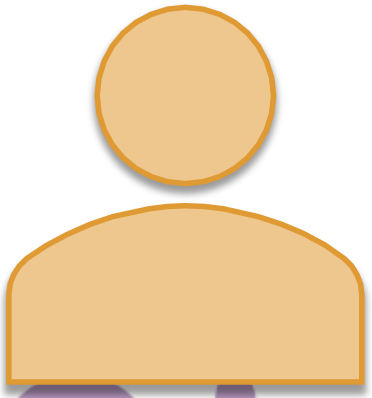

We are NervCode

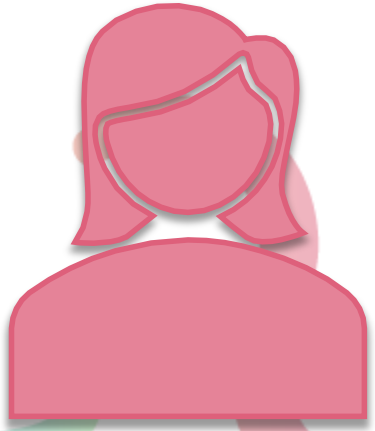
nc



Andres David



Jean Pierre



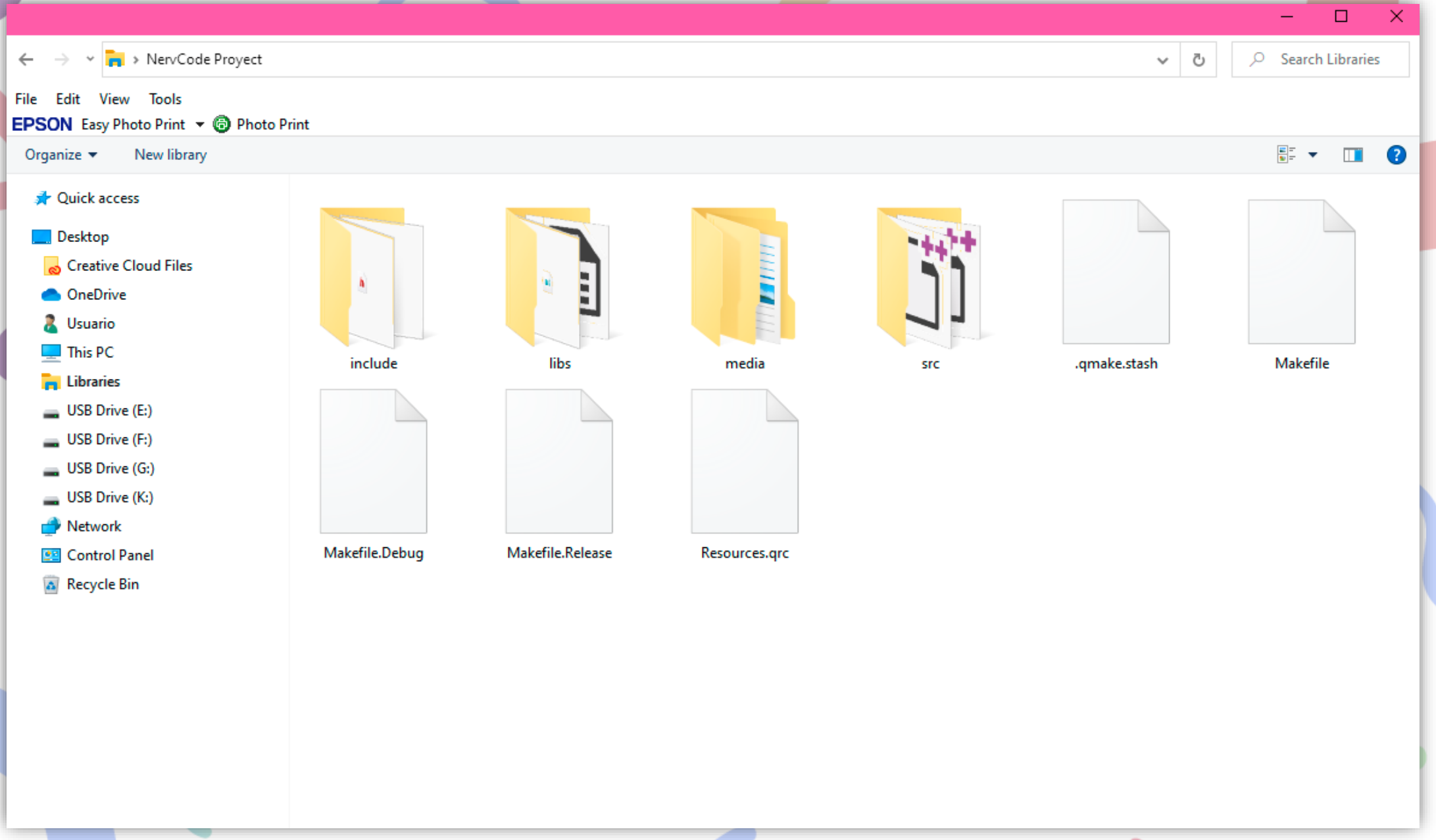
Ingrid

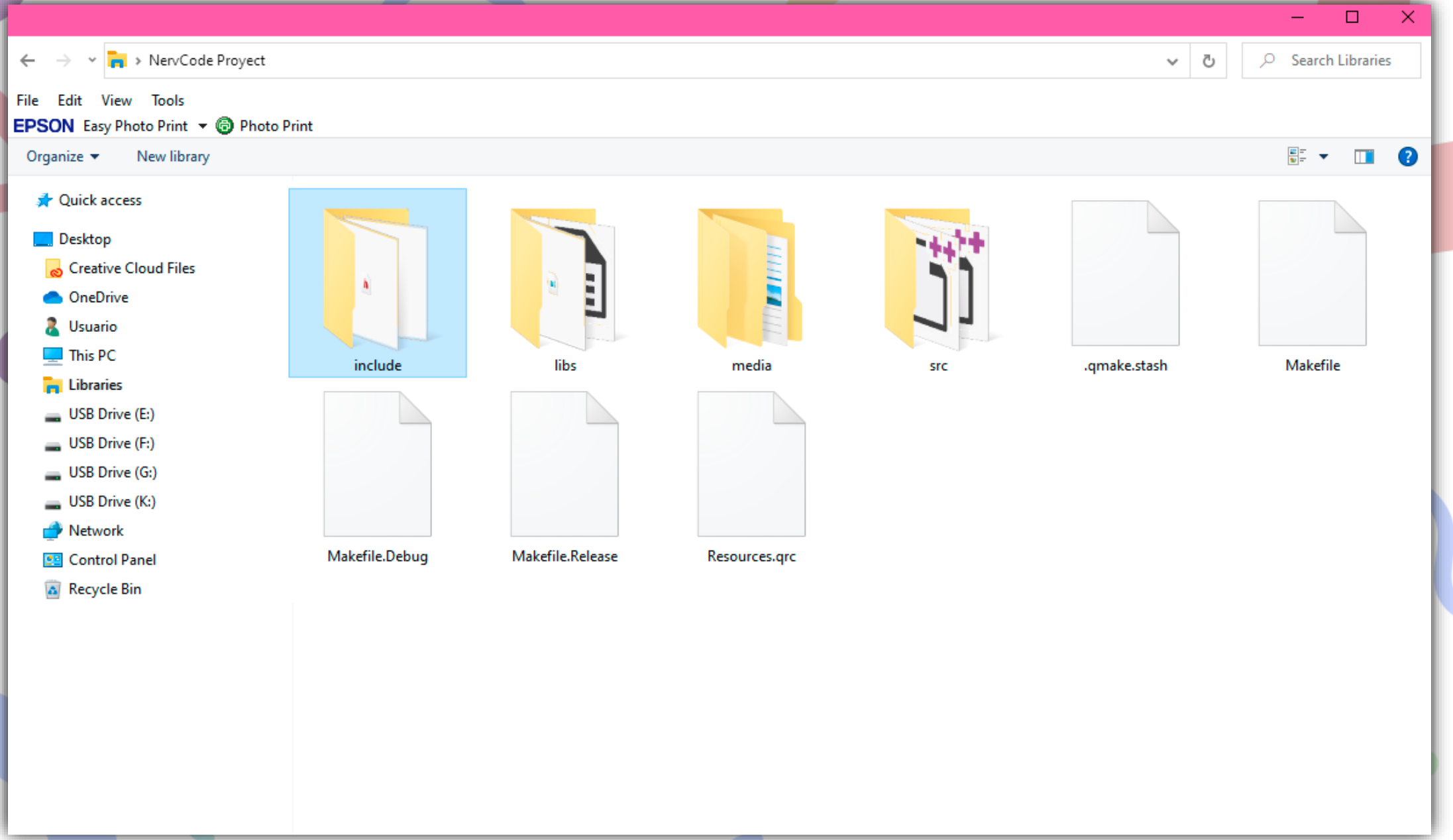
Jhan Alejandro

Proyecto Final IPOO

Panorama

- Elaboración de un programa capaz de aplicar filtros a imágenes .ppm y .pgm .Para la cual se utilizaron las herramientas aprendidas en clase de programación orientada objetos y utilizando QT creator para crear la interfaz.


NervCode Proyect
Search Libraries
File
Edit
View
Tools
EPSON Easy Photo Print
Photo Print
Organize
New library
Quick access
Desktop
Creative Cloud Files
OneDrive
Usuario
This PC
Libraries
USB Drive (E:)
USB Drive (F:)
USB Drive (G:)
USB Drive (K:)
Network
Control Panel
Recycle Bin
include
libs
media
src
.qmake.stash
Makefile
Makefile.Debug
Makefile.Release
Resources.qrc


NervCode Proyect
Search Libraries
File Edit View Tools
EPSON Easy Photo Print
Photo Print
Organize
New library
Quick access
Desktop
Creative Cloud Files
OneDrive
Usuario
This PC
Libraries
USB Drive (E:)
USB Drive (F:)
USB Drive (G:)
USB Drive (K:)
Network
Control Panel
Recycle Bin
include
libs
media
src
.qmake.stash
Makefile
Makefile.Debug
Makefile.Release
Resources.qrc

Herencia multinivel

Class Image

Protected
- int rows;
- int columns;
- int maxRGB
- string name;

Public
- Image()
- Image(string fileName);
- ~Image();
- int getRows();
- int getColumns();
- int getMaxRGB();
- string getName();

Class File : Public Filters

Private
- string name;
- int width;
- int height;
- string header;
- int matrixStart;
- int maxIntensity;

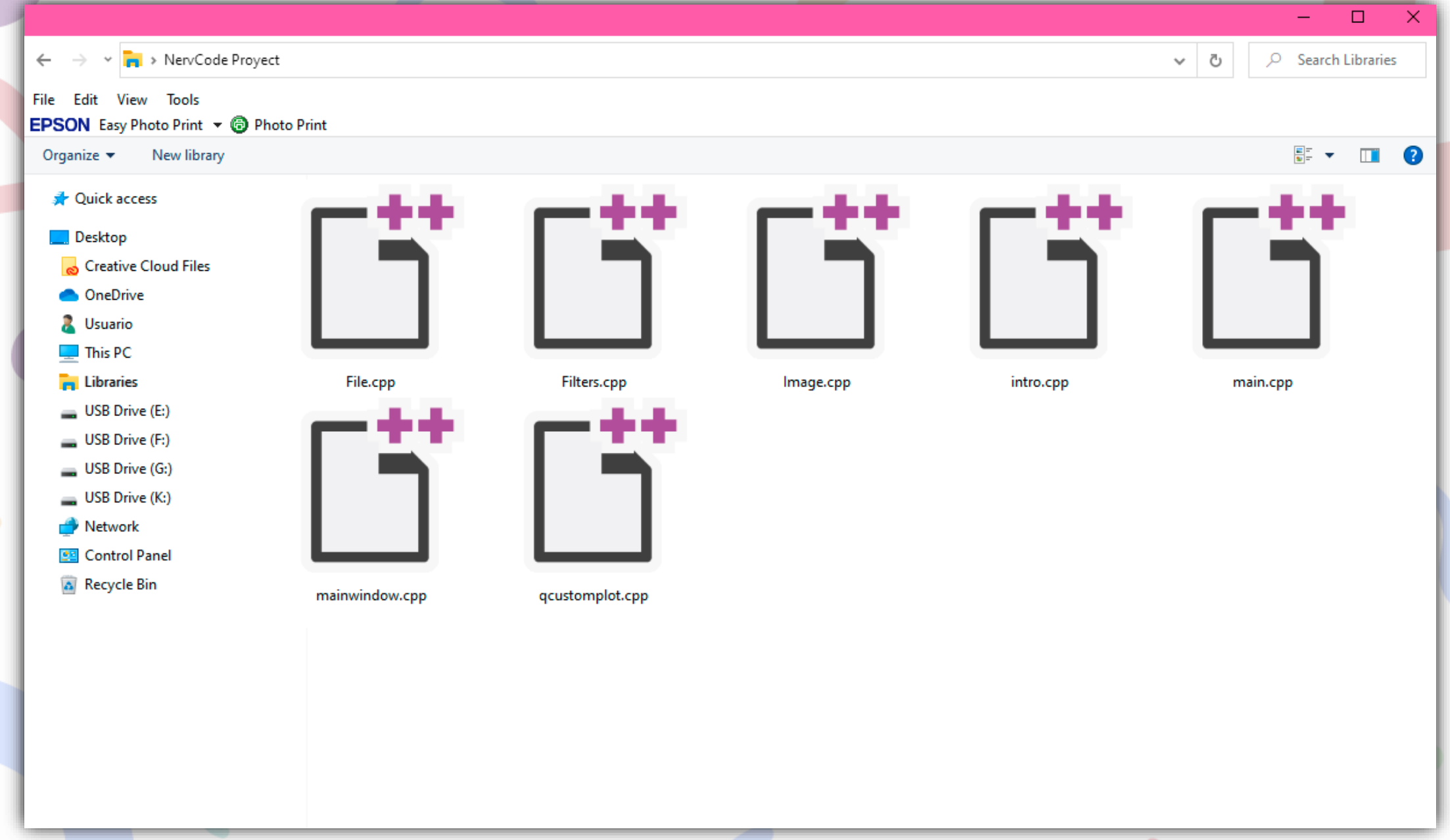
Protected
- vector<vector<int>> picture;
- vector<vector<int>> filtered;
- string extension;

Public
- File();
- File(string fileName);
- ~File();
- void readFile();
- void getFileValues();
- void writeFile(string fileName);
- void convertToVector();
- vector<vector<int>> getPicture();
- void setFiltered(vector<vector<int>> fromFilters);
- string getExtension();
- QVector<double> histogram();
- int getmaxIntensity();

Class Filters : Public File

Private
- string newFileName;
- int rgbCounter;
- int filter;
- vector<vector<int>> image;
- vector<vector<int>> secondImage;
- vector<vector<int>> copyImg;
- int maxRGB;
- bool isGrey;

Public
- Filters();
- Filters(File &obj, int filter, int value=0);
- Filters(File &obj, File &obj2);
- ~Filters();
- vector<vector<int>> getFiltered();
- void normal();
- bool isBlacknWhite();
- void threshold();
- void flip();
- void toGrey();
- void toNegative();
- void toSephia();
- void toTone(int color);
- void toColor();
- void overLay();


NervCode Proyect
Search Libraries
File Edit View Tools
EPSON Easy Photo Print
Photo Print
Organize
New library
Quick access
Desktop
Creative Cloud Files
OneDrive
Usuario
This PC
Libraries
USB Drive (E:)
USB Drive (F:)
USB Drive (G:)
USB Drive (K:)
Network
Control Panel
Recycle Bin
File.cpp
Filters.cpp
Image.cpp
intro.cpp
main.cpp
mainwindow.cpp
qcustomplot.cpp

Images.cpp

```cpp
Image::Image(){
    this->rows = 0;
    this->columns = 0;
    this->maxRGB = 0;
    this->name = " ";
}

Image::Image(string fileName){
    this->rows = 0;
    this->columns = 0;
    this->maxRGB = 0;
    this->name = fileName;
}
```

```cpp
int Image::getMaxRGB()
{
    return maxRGB;
}

string Image::getName()
{
    return this->name;
}
```

```cpp
Image::~Image(){}

int Image::getRows(){
    return rows;
}

int Image::getColumns()
{
    return columns;
}
```

File.cpp

```cpp
void File::readFile()
{
    ifstream imageFile;
    string line;
    imageFile.open(this->name, ios::in);
    if(imageFile.fail())
    {
        cout << "File couldn't be openend" << endl;
        exit(1);
    }
    while(!imageFile.eof())
    {
        getline(imageFile, line);
        cout << line << endl;
    }
    imageFile.close();
}
```

```cpp
void File::getFileValues()
{
    ifstream imageFile;
    string line;
    imageFile.open(this->name, ios::in);
    while(getline(imageFile,line))
    {
        rows++;
        if(rows == 1 && line[0] == 'P')
        {
            this->header = line;
        }
        if(rows == 2 && line[0] != '#' && !line.empty())
        {
            this->width = stod(line.substr(0, line.find(" ")+1));
            this->height = stod(line.substr(line.find(" ")+1,line.length()));
        }
        if(rows == 3)
        {
            if(line.length() <= 3)
            {
                maxRGB = stod(line);
                matrixStart = 3;
            }
            else
            {
                this->width = stod(line.substr(0, line.find(" ")+1));
                this->height = stod(line.substr(line.find(" ")+1,line.length()));
            }
        }
        if(rows == 4 && line.length() <=3)
        {
            maxRGB = stod(line);
            matrixStart = 4;
        }
        int lineLenght = line.length();
        for(int i=0; i<lineLenght; i++)
        {
            if(line[i] == ' ')
            {
                this->columns++;
            }
        }
    }
    this->rows = this->rows - matrixStart;
}
```

```
void File::writeFile(string fileName)
{
    QString tempPath = QDir::tempPath();
    QString extension = QString::fromStdString(this->extension);
    QString temporaryFile = tempPath +  "FilteredImage" + extension;
    ofstream newImageFile;
    if(fileName.length()>25){
        newImageFile.open(fileName, ios::out);
    }else
    {
        newImageFile.open(temporaryFile.toUtf8().constData(), ios::out);
    }
    if(newImageFile.fail())
    {
        cout << "Error could not open archive" <<  endl;
        exit(1);
    }

    newImageFile << this->header << endl;
    newImageFile << "#NervCode" << endl;
    newImageFile << this->width << " " << this->height << endl;
    newImageFile << maxRGB << endl;
    int filterSize = this->filtered.size();
    for(int i=0; i < filterSize; i++)
    {
        int columnSize = this->filtered[i].size();
        for(int j=0; j<columnSize; j++)
        {
            newImageFile << this->filtered[i][j] << " ";
        }
        newImageFile << endl;
    }
    newImageFile.close();
}
```

```
void File::convertToVector()
{
    ifstream imageFile;
    imageFile.open(this->name, ios::in);
    string line;
    for (int i = 0; i < this->rows + matrixStart; i++)
    {
        vector<int> row;
        string line;
        getline(imageFile, line);
        stringstream spliter(line);
        string columnValue;
        if (i >= 4)
        {
            while (getline(spliter, columnValue, ' '))
            {
                int values = stoi(columnValue);
                row.push_back(values);
            }
            picture.push_back(row);
        }
    }
}
```

```
QVector<double> File::histogram()
{
    int valueIntensity = 0;
    QVector<double> histogram(255);

    while(valueIntensity < 255)
    {
        int counter = 0;
        int imageSize = this->picture.size();
        for(int rows = 0; rows<imageSize; rows++)
        {
            int columnsSize = this->picture[rows].size();
            for(int columns = 0; columns < columnsSize; columns++)
            {
                if(this->picture[rows][columns] == valueIntensity)
                {
                    counter ++;
                }

            }
        }
        if(counter > this->maxIntensity)
        {
            this->maxIntensity = counter;
        }
        valueIntensity = valueIntensity + 1;
        histogram[valueIntensity] = counter;
    }
    return histogram;
```

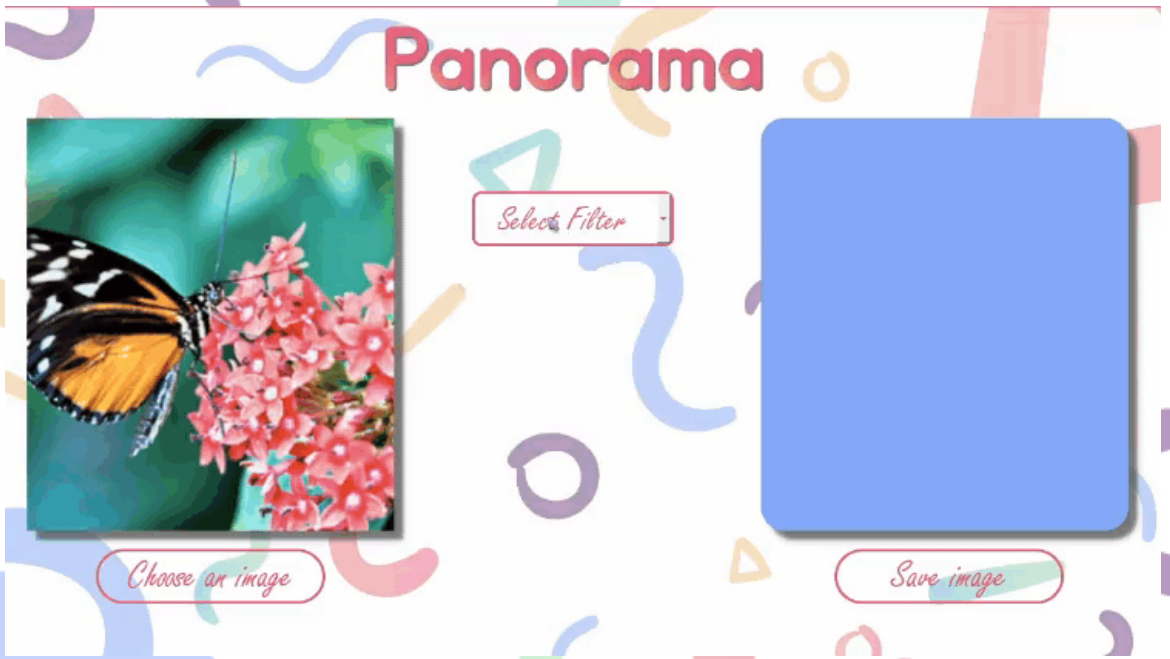
Filters.cpp

```cpp
Filters::Filters(File &obj, int filter, int value)
{
    this->image = obj.getPicture();
    this->copyImg = this->image;
    this->maxRGB = obj.getMaxRGB();
    this->rgbCounter=0;
    this->isGrey = false;
    this->isBlacknWhite();
    switch (filter)
    {
    case 1:
        this->normal();
        break;
    case 2:
        this->toGrey();
        break;
    case 3:
        this->toNegative();
        break;
    case 4:
        this->toSephia();
        break;
    case 5:
        this->toTone(value);
        break;
    case 6:
        this->flip();
        break;
    case 7:
        this->toColor();
        break;
    case 8:
        this->threshold();
        break;
    default:
        break;
    }
}
```

Black and White

```
void Filters::toGrey()
{
    int imageSize = this->image.size();
    for(int i=0; i<imageSize; i++)
    {
        int columnsSize =this->image[i].size();
        for(int j=0; j<columnsSize; j++)
        {
            int red = this->image[i][j];
            int green = this->image[i][j+1];
            int blue = this->image[i][j+2];
            int gray = 0.2989 * red + 0.5870 * green + 0.1140 * blue;
            this->image[i][j] = gray;
            this->image[i][j+1] = gray;
            this->image[i][j+2] = gray;
            j = j + 2;
        }
    }
}
```

Sepia




```
void Filters::toSepia()
{
    int imageSize = this->image.size();
    for(int i=0; i<imageSize; i++)
    {
        int columnsSize =this->image[i].size();
        for(int j=0; j<columnsSize; j++)
        {
            int red = this->image[i][j];
            int green = this->image[i][j+1];
            int blue = this->image[i][j+2];
            int tr = 0.393 *red+ 0.769 *green+ 0.189 *blue;
            if (tr >= maxRGB)
            {
                tr = maxRGB;
            }
            int tg = 0.349 *red+ 0.686 *green+ 0.168 *blue;
            if (tg >= maxRGB)
            {
                tg = maxRGB;
            }
            int tb = 0.272 *red+ 0.534 *green+ 0.131 *blue;
            if (tb >= maxRGB)
            {
                tb = maxRGB;
            }
            this->image[i][j] = tr;
            this->image[i][j+1] = tg;
            this->image[i][j+2] = tb;
            j = j + 2;
        }
    }
}
```

```
void Filters::toNegative(){
    int imageSize = this->image.size();
    for(int i=0; i<imageSize; i++)
    {
        int columnsSize =this->image[i].size();
        for(int j=0; j<columnsSize; j++)
        {
            if(this->image[i][j] == 0)
            {
                this->image[i][j] = maxRGB;
            }
            else if(this->image[i][j] == maxRGB)
            {
                this->image[i][j] = 0;
            }
            else
            {
                int negative = maxRGB - this->image[i][j];
                this->image[i][j] = negative;
            }
        }
    }
}
```

Negative

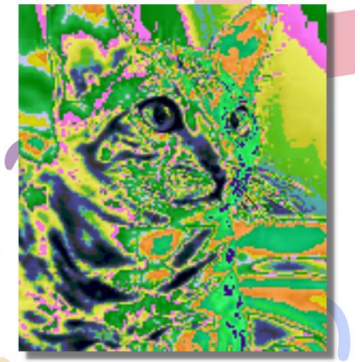
Tone
Tone
Tone

```cpp
void Filters::toTone(int color)
{

  vector<vector<int>> colorTone = {{0,0,0},{91, -99, -14}, {-115, -86, -23},{-72,75,-77}};
  int imageSize = this->image.size();
  for (int i = 0; i < imageSize; i++)
  {
    int columnsSize =this->image[i].size();
    for (int j = 0; j < columnsSize; j++)
    {
      int red = this->image[i][j];
      int green = this->image[i][j + 1];
      int blue = this->image[i][j + 2];

      int toneR = colorTone[color][0] + red;
      int toneG = colorTone[color][1] + green;
      int toneB = colorTone[color][2] + blue;

      if (toneR <= 0)
        toneR = 0;
      if (toneR >= 255)
        toneR = 255;
      if (toneG <= 0)
        toneG = 0;
      if (toneG >= 255)
        toneG = 255;
      if (toneB <= 0)
        toneB = 0;
      if (toneB >= 255)
        toneB = 255;
      this->image[i][j] = toneR;
      this->image[i][j+1] = toneG;
      this->image[i][j+2] = toneB;
      j = j + 2;
    }
  }
}
```

To Color

Panorama

ToColor



Choose an image

Filter applied!

Save image

```
void Filters::toColor(){
    if(this->isGrey){
        int imageSize = this->image.size();
        for (int rows = 0; rows < imageSize; rows++)
        {
            int columnsSize =this->image[rows].size();
            for (int columns = 0; columns < columnsSize; columns++)
            {
                int r = copyImg[rows][columns]/50;
                int g = copyImg[rows][columns + 1]/40;
                int b = copyImg[rows][columns + 2]/30;
                this->image[rows][columns] = copyImg[rows][columns]*r;
                this->image[rows][columns+1] = copyImg[rows][columns+1]*g;
                this->image[rows][columns+2] = copyImg[rows][columns+2]*b;
                columns = columns+2;
            }
        }
    }
    else
    {

    }
}
```

```
void Filters::threshold()
{
    int largest = 0;
    int imageRows = this->image.size();
    for(int rows = 0; rows < imageRows; rows++)
    {
        int imageColumns = this->image[rows].size();
        for(int columns = 0; columns < imageColumns; columns++)
        {
            int red = this->image[rows][columns];
            int green = this->image[rows][columns+1];
            int blue = this->image[rows][columns+2];

            if(red > green && red > blue){
                largest = red;
            }else if(green > red && green > blue){
                largest = green;

            }else if(blue > red && blue > green){
                largest = blue;

            }else if(red == green || red == blue || green == blue){
                largest = red;
            }

            if(largest>127)
            {
                this->image[rows][columns] = 255;
                this->image[rows][columns+1] = 255;
                this->image[rows][columns+2] = 255;
            }
            else
            {
                this->image[rows][columns] = 0;
                this->image[rows][columns+1] = 0;
                this->image[rows][columns+2] = 0;
            }

            columns = columns + 2;
        }
    }
}
```

Threshold

Panorama

Flip

Choose an image

Filter applied!

Save image

Flip

```
void Filters::flip()
{
    int imageRows = this->image.size();
    for(int rows = 0; rows < imageRows; rows++)
    {
        int imageColumns =  this->image[rows].size();
        for(int columns = 0; columns < imageColumns; columns++ )
        {
            int red = copyImg[rows][imageColumns - 3 - columns];
            int green = copyImg[rows][imageColumns - 2 - columns];
            int blue = copyImg[rows][imageColumns - 1 - columns];
            this->image[imageRows-1-rows][columns] = red;
            this->image[imageRows-1-rows][columns+1] = green;
            this->image[imageRows-1-rows][columns+2] = blue;
            columns = columns + 2;
        }
    }
}
```

```
void Filters::overLay(){

    int imageRows = this->image.size();
    for(int rows = 0; rows < imageRows; rows++)
    {
        int secondImgRows = this->secondImage.size();
        if(rows > secondImgRows)
        {
            this->secondImage.push_back(this->image[rows]);
        }
        else
        {

            int imageColumns = this->image[rows].size();
            int secondImgColumns = this->secondImage[rows].size();
            for(int columns = 0; columns < imageColumns; columns++)
            {
                if(columns > secondImgColumns)
                {
                    int greenValue = 0;
                    if(rgbCounter == 0) greenValue = 0;
                    if(rgbCounter == 1) greenValue = 255;
                    if(rgbCounter == 2)
                    {
                        greenValue = 0;
                        rgbCounter = 0;
                    }
                    this->secondImage[rows].push_back(greenValue);
                }
                rgbCounter++;
            }
        }
    }
```

Overlay


```
    for(int rows = 0; rows < imageRows; rows++)
    {
        int imageColumns = this->secondImage[rows].size();
        for(int columns = 0; columns<imageColumns; columns++)
        {
            int red = this->secondImage[rows][columns] ;
            int green = this->secondImage[rows][columns+1];
            int blue = this->secondImage[rows][columns+2];
            if((red != 0 && green != 255 && blue!= 0) || (red == 255 && green == 255 && blue == 255))
            {
                this->image[rows][columns] = this->secondImage[rows][columns];
                this->image[rows][columns+1] = this->secondImage[rows][columns+1];
                this->image[rows][columns+2] = this->secondImage[rows][columns+2];
            }else{

            }
            columns = columns+2;
        }
    }
}
```

Histogram

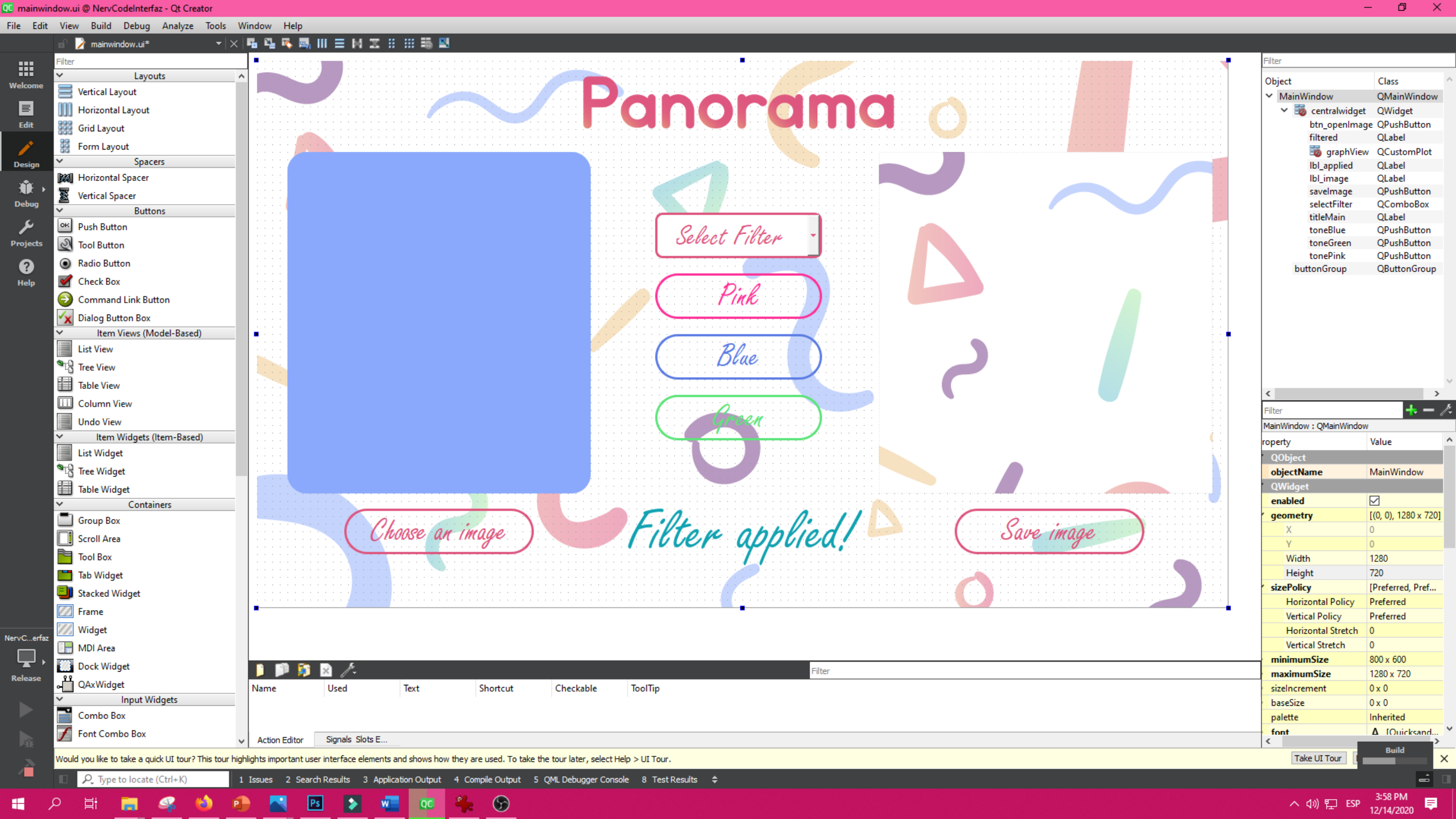
Para la realización de Histogram utilizamos la librería QCustomPlot.

```
void Filters::overLay(){

    int imageRows = this->image.size();
    for(int rows = 0; rows < imageRows; rows++)
    {
        int secondImgRows = this->secondImage.size();
        if(rows > secondImgRows)
        {
            this->secondImage.push_back(this->image[rows]);
        }
        else
        {

            int imageColumns = this->image[rows].size();
            int secondImgColumns = this->secondImage[rows].size();
            for(int columns = 0; columns < imageColumns; columns++)
            {
                if(columns > secondImgColumns)
                {
                    int greenValue = 0;
                    if(rgbCounter == 0) greenValue = 0;
                    if(rgbCounter == 1) greenValue = 255;
                    if(rgbCounter == 2)
                    {
                        greenValue = 0;
                        rgbCounter = 0;
                    }
                    this->secondImage[rows].push_back(greenValue);
                }
                rgbCounter++;
            }
        }
    }
```

```
    for(int rows = 0; rows < imageRows; rows++)
    {
        int imageColumns = this->secondImage[rows].size();
        for(int columns = 0; columns<imageColumns; columns++)
        {
            int red = this->secondImage[rows][columns] ;
            int green = this->secondImage[rows][columns+1];
            int blue = this->secondImage[rows][columns+2];
            if((red != 0 && green != 255 && blue!= 0) || (red == 255 && green == 255 && blue == 255))
            {
                this->image[rows][columns] = this->secondImage[rows][columns];
                this->image[rows][columns+1] = this->secondImage[rows][columns+1];
                this->image[rows][columns+2] = this->secondImage[rows][columns+2];
            }else{

            }
            columns = columns+2;
        }
    }
}
```

```
else if(arg1 == "Overlay")
{
    //Only show add second image and not tone bottons
    ui->openImage2->setVisible(true);

    QGraphicsOpacityEffect *effBtnImage2 = new QGraphicsOpacityEffect(ui->openImage2);
    ui->openImage2->setGraphicsEffect(effBtnImage2);

    QPropertyAnimation *animationAppearPink = new QPropertyAnimation(effBtnImage2, "opacity");
    animationAppearPink->setDuration(500);
    animationAppearPink->setStartValue(0.0);
    animationAppearPink->setEndValue(1.0);
    animationAppearPink->start();

    ui->tonePink->setVisible(false);
    ui->toneBlue->setVisible(false);
    ui->toneGreen->setVisible(false);
    ui->graphView->setVisible(false);
    ui->filtered->setVisible(true);
    ui->saveImage->setVisible(true);
    if(secondImg != " ")
    {
        string secondImgDir = secondImg.toUtf8().constData();
        File *secondImg = new File(secondImgDir);
        Filters apply(*newImage,*secondImg);
        newImage->setFiltered(apply.getFiltered());
        if(save == 1){
            newImage->writeFile(saveImgDir);
            QFile file(temporaryFilePath);
            file.remove();
        }else{
            newImage->writeFile(temporaryFile.toUtf8().constData());
        }
    }
}
```

Qt (pronunciado “cute” en ingles) es un conjunto de herramientas de widgets gratuito y de código abierto para crear interfaces gráficas de usuario.


mainwindow.ui @ NervCodeInterfaz - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
mainwindow.ui*
Welcome
Edit
Design
Debug
Projects
Help
NervC...erfaz
Release
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MDI Area
Dock Widget
QAxWidget
Input Widgets
Combo Box
Font Combo Box
Panorama
Select Filter
Pink
Blue
Green
Choose an image
Filter applied!
Save image
Name
Used
Text
Shortcut
Checkable
ToolTip
Action Editor
Signals Slots E...
Object
Class
MainWindow
QMainWindow
centralwidget
QWidget
btn_openImage
QPushButton
filtered
QLabel
graphView
QCustomPlot
lbl_applied
QLabel
lbl_image
QLabel
saveImage
QPushButton
selectFilter
QComboBox
titleMain
QLabel
toneBlue
QPushButton
toneGreen
QPushButton
tonePink
QPushButton
buttonGroup
QButtonGroup
MainWindow : QMainWindow
Property
Value
QObject
objectName
MainWindow
QWidget
enabled
geometry
[(0, 0), 1280 x 720]
X
0
Y
0
Width
1280
Height
720
sizePolicy
[Preferred, Pref...
Horizontal Policy
Preferred
Vertical Policy
Preferred
Horizontal Stretch
0
Vertical Stretch
0
minimumSize
800 x 600
maximumSize
1280 x 720
sizeIncrement
0 x 0
baseSize
0 x 0
palette
Inherited
font
[Quicksand...
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Build
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
5 QML Debugger Console
8 Test Results
ESP
3:58 PM
12/14/2020

mainwindow.cpp @ NervCodeInterfaz - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects | mainwindow.cpp | MainWindow::on_selectFilter_activated(const QString &, int) -> void | Windows (CRLF) | Line: 361, Col: 10

Welcome
Edit
Design
Debug
Projects
Help

NervCodeInterfaz
- NervCodeInterfaz.pro
- Headers
  - File.h
  - Filters.h
  - Image.h
  - intro.h
  - mainwindow.h
  - qcustomplot.h
- Sources
  - File.cpp
  - Filters.cpp
  - Image.cpp
  - intro.cpp
  - main.cpp
  - mainwindow.cpp
  - qcustomplot.cpp
- Forms
  - intro.ui
  - mainwindow.ui
- Resources
  - Resources.qrc
- Other files

Open Documents
- File.cpp
- File.h
- Filters.cpp
- Filters.h
- Image.cpp
- mainwindow.cpp
- mainwindow.ui
- qcustomplot.cpp

NervC...erfaz
Release

```
133
134
135
136 void MainWindow::on_selectFilter_activated(const QString &arg1,int save)
137 {
138     QImage image;
139     QString tempPath = QDir::tempPath();
140     string imageDir = filename.toUtf8().constData();
141     QString extension = QString::fromStdString(imageDir.substr(imageDir.find("."), imageDir.length()));
142     QString temporaryFile = "FilteredImage"+extension;
143     QString temporaryFilePath = tempPath +  temporaryFile;
144     File *newImage = new File(imageDir);
145     lastFilter = arg1;
146     if(arg1 == "Select Filter")
147     {
148         ui->tonePink->setVisible(false);
149         ui->toneBlue->setVisible(false);
150         ui->toneGreen->setVisible(false);
151         ui->openImage2->setVisible(false);
152         ui->graphView->setVisible(false);
153         ui->filtered->setVisible(true);
154         ui->saveImage->setVisible(true);
155         Filters apply(*newImage,1);
156         newImage->setFiltered(apply.getFiltered());
157         if(save == 1){
158             newImage->writeFile(saveImgDir);
159             QFile file(temporaryFilePath);
160             file.remove();
161         }else{
162             newImage->writeFile(temporaryFile.toUtf8().constData());
163         }
164     }
165     if(arg1 == "BlacknWhite")
166     {
167         ui->tonePink->setVisible(false);
168         ui->toneBlue->setVisible(false);
169         ui->toneGreen->setVisible(false);
170         ui->openImage2->setVisible(false);
171         ui->graphView->setVisible(false);
172         ui->lbl_applied->setVisible(true);
173         ui->lbl_applied->setText("Filter applied!");
174         ui->filtered->setVisible(true);
175         ui->saveImage->setVisible(true);
176         Filters apply(*newImage,2);
177         newImage->setFiltered(apply.getFiltered());
178         if(save == 1){
179             newImage->writeFile(saveImgDir);
180             QFile file(temporaryFilePath);
```

Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour. Take UI Tour | Do Not Show Again

Type to locate (Ctrl+K) | 1 Issues | 2 Search Results | 3 Application Output | 4 Compile Output | 5 QML Debugger Console | 8 Test Results

ESP 4:02 PM 12/14/2020

¡Probemos el programa!