1. **Задание**

   1. Студенту предлагается выбрать предметную область (либо из предложенных вариантов, либо предложить свою) и согласовать ее с преподавателем (преподаватель фиксирует у себя тему) с тем, чтобы в дальнейшем использовать в курсовом проекте по объектно-ориентированному программированию. Не разрешается выбирать одну и ту же предметную область внутри одной группы.

   2. Проанализировать предметную область, выделить сущности предметной области. Сущностей должно быть не менее 3-х (в курсовом проекте – не менее 5- 6 сущностей). Сформулировать функционал, который Вы планируете разработать.

   В предметной области должна быть реализована многозначная зависимость.

   3. На основе сущностей предметной области спроектировать иерархии (или совокупности) связанных классов. Для каждого класса в этом разделе должны быть указаны имена полей (с указанием семантического смысла для каждого поля) и перечислены методы класса (т.е. хедеры классов).

   4. В работе рекомендуется использовать не менее двух паттернов. Паттерны должны быть разных типов: порождающие (кроме Сингелтона), поведенческие, структурные. Студент должен обосновать использование тех или иных паттернов. Результатом проектирования является представленная диаграмма классов (или несколько диаграмм), а также должна быть диаграмма классов для каждого используемого паттерна (с именами классов студента)

   5. Собственно, разработка программы. На основе перечисленного функционала (из пункта 2) формируете структуру меню и уделяете внимание каждому пункту.

2. **Постановка задачи**

   Для выполнения лабораторной работы была выбрана предметная область “Ремонтная мастерская”

   Были выделены следующие 3 сущности:

   - Клиент
   - Мастер
   - Заказ

   Функционал, который следует разработать для данной области:

   - Обработка клиента:
     - Добавление клиента;

- Удаление клиента.
- Обработка мастеров:
  - Добавление мастера;
  - Удаление мастера.
- Обработка заказов:
  - Добавление заказа;
  - Удаление заказа;
  - Выполнение заказа.

3. **Разработка классов**

Класс клиент:

```
class Client : public QObject
{
   Q_OBJECT
public:
   explicit Client(QObject *parent = nullptr, QString FIO = "", QString Adress="",
QString PhoneNumber="");
   static int PhoneNumberToRegNumber(QString);

   int GetRegNumber();
   QString GetFIO();
   QString GetAdress();
   QString GetPhoneNumber();

private:
   int RegNumber;
   QString FIO;
   QString Adress;
   QString PhoneNumber;

signals:
};
```

Класс содержит поля:

- RegNumber – уникальный номер клиента, формируемый на основе номера телефона PhoneNumber
- FIO – ФИО клиента

- Adress – Адрес Клиента
- PhoneNumber – Номер телефона клиента

Класс мастер:

```
class Worker : public QObject
{
   Q_OBJECT
public:
   explicit Worker(QObject *parent = nullptr, QString FIO="", QString Position="");
   static QString FIOnPosToRegCode(QString, QString);

   void SetOrder(int);

   QString GetRegCode();
   QString GetFIO();
   QString GetPosition();
   bool IsBusy();
   QString GetOrderStatus();

private:
   QString RegCode;
   QString FIO;
   QString Position;
   bool MasterBusy;
   int OrderNumber;
};
```

Класс содержит поля:

- RegCode – уникальный номер мастера, формируемый на основе первых букв ФИО
- FIO – ФИО мастера
- Position – Должность Мастера
- MasterBusy – Занятость мастера(true = занят)
- OrderNumber – Номер заказа, которым занят мастер(-1 значит нет заказа)

Класс заказ:

```
class Order : public QObject
{
    Q_OBJECT
public:
    enum class RepairStatus
    {
        InProcess,
        Completed
    };

    explicit Order(QObject *parent = nullptr, int ClientRegNumber = 0,
                   QString MasterRegCode = "", QString ThingToRepair = "");
    void SetOrderNumber(int);
    void SetRepairStatus(RepairStatus);

    int GetClientRegNumber();
    QString GetMasterRegCode();
    QString GetThingToRepair();
    QString GetStatusOfRepair();

private:
    int ClientRegNumber;
    QString MasterRegCode;
    QString ThingToRepair;
    RepairStatus StatusOfRepair;
    int OrderNumber;
};
```

Класс содержит поля:

- ClientRegNumber – код клиента, который отдал вещь в ремонт
- MasterRegCode – код мастера, который чинит вещь
- ThingToRepair – Название вещи, которую чинят
- StatusOfRepair – Статус починки(В процессе/Выполнен)
- OrderNumber – Номер заказа

В лабораторной работе используется два паттерна проектирования: Структурный Фасад и Поведенческий Наблюдатель. Ниже на рисунках 1 и 2-6 соответственно указаны UML диаграммы паттернов.

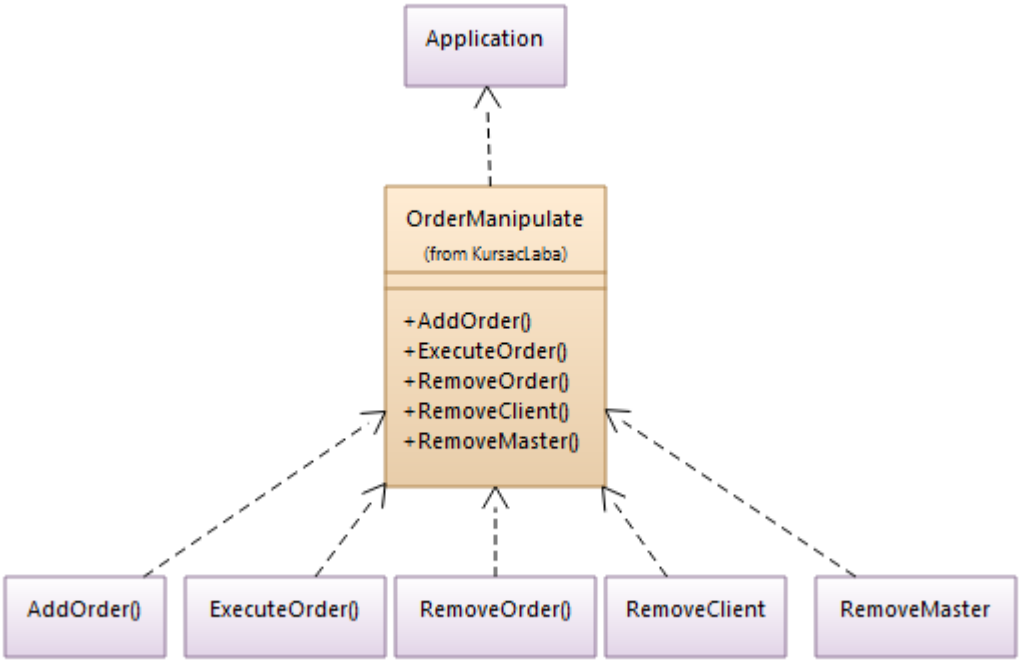


Рисунок 1 – Паттерн Фасад

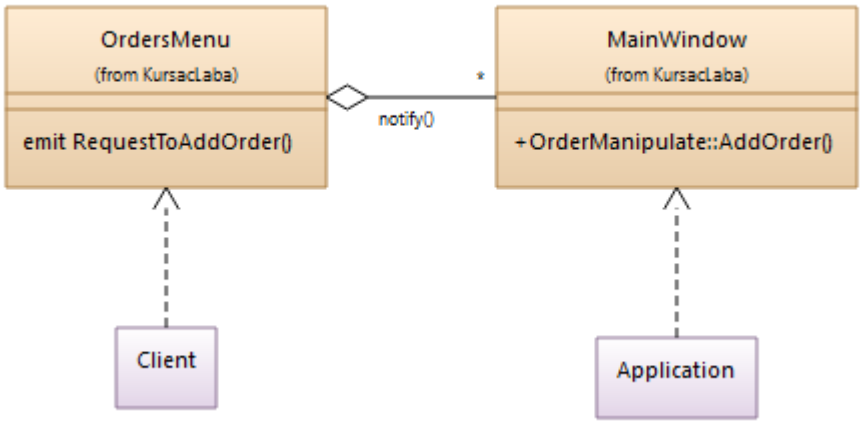


Рисунок 2 – Паттерн Наблюдатель

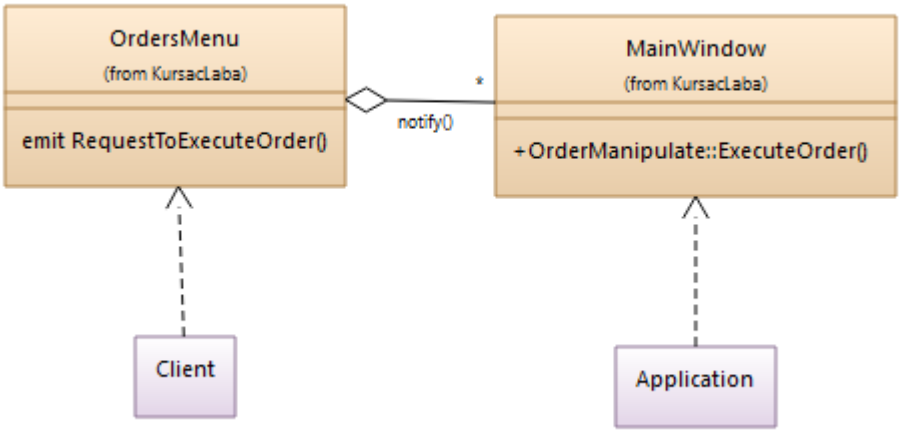


Рисунок 3 – Паттерн Наблюдатель

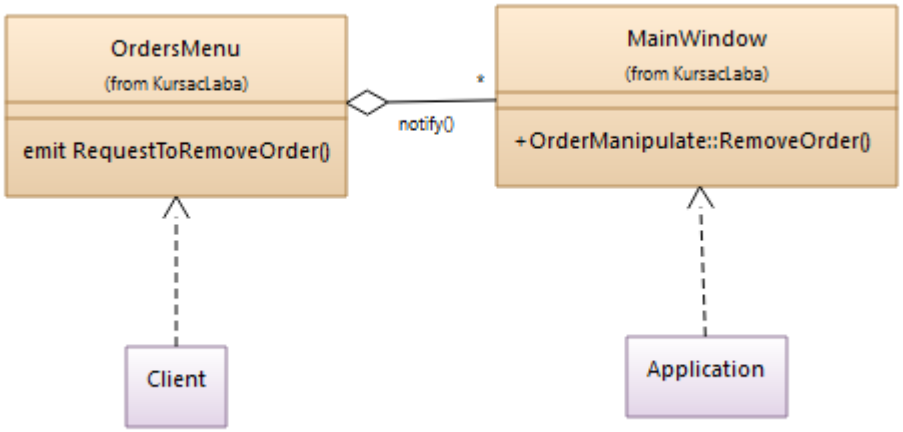


Рисунок 4 – Паттерн Наблюдатель

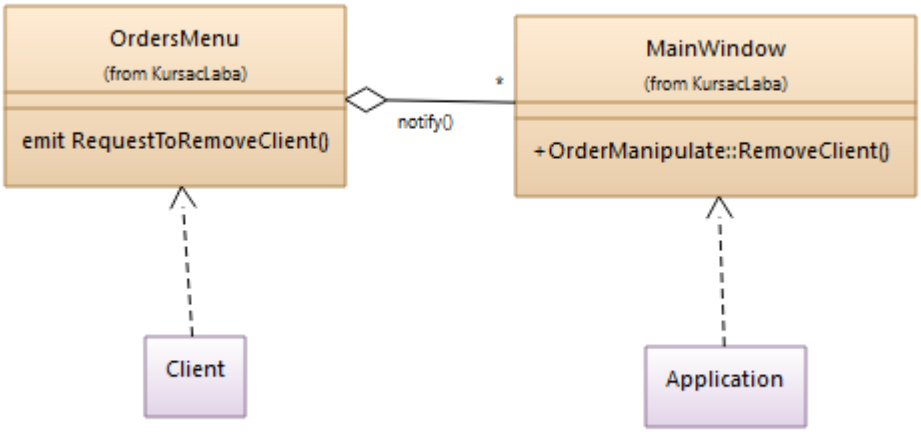


Рисунок 5 – Паттерн Наблюдатель

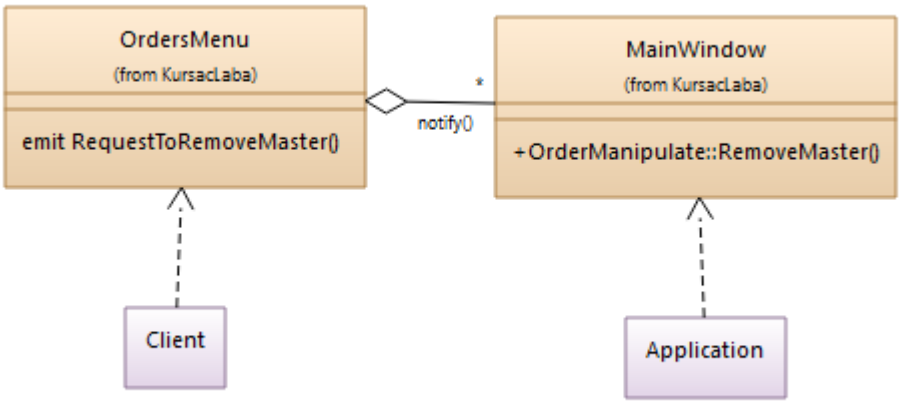


Рисунок 6 – Паттерн Наблюдатель

На рисунке 7 изображена диаграмма всего приложения

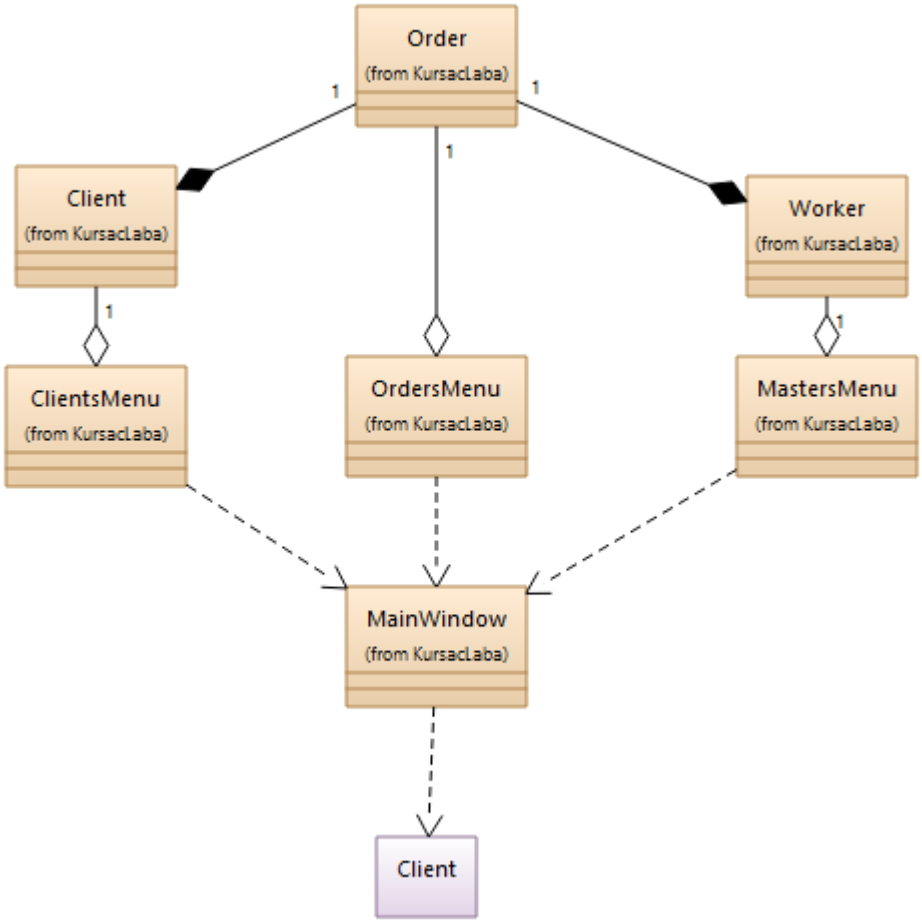


Рисунок 7 – Диаграмма приложения

4. Форма программы

В приложении присутствует 4 формы, 3 из которых представляют из себя разные списки сущностей. На рисунке 8 представлено основное меню, на рисунке 9 меню управление клиента, на рисунке 10 управление заказами, на рисунке 11 управление мастерами.

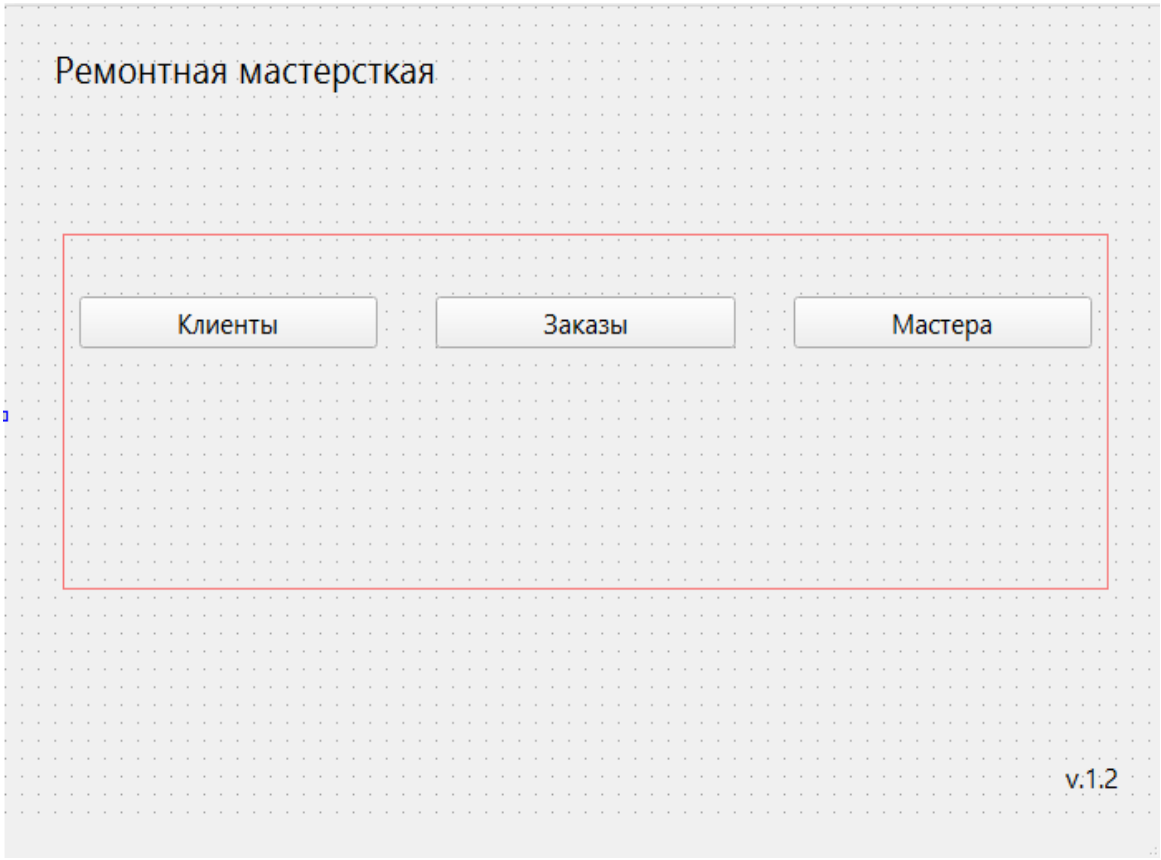


Рисунок 9 – Главное меню

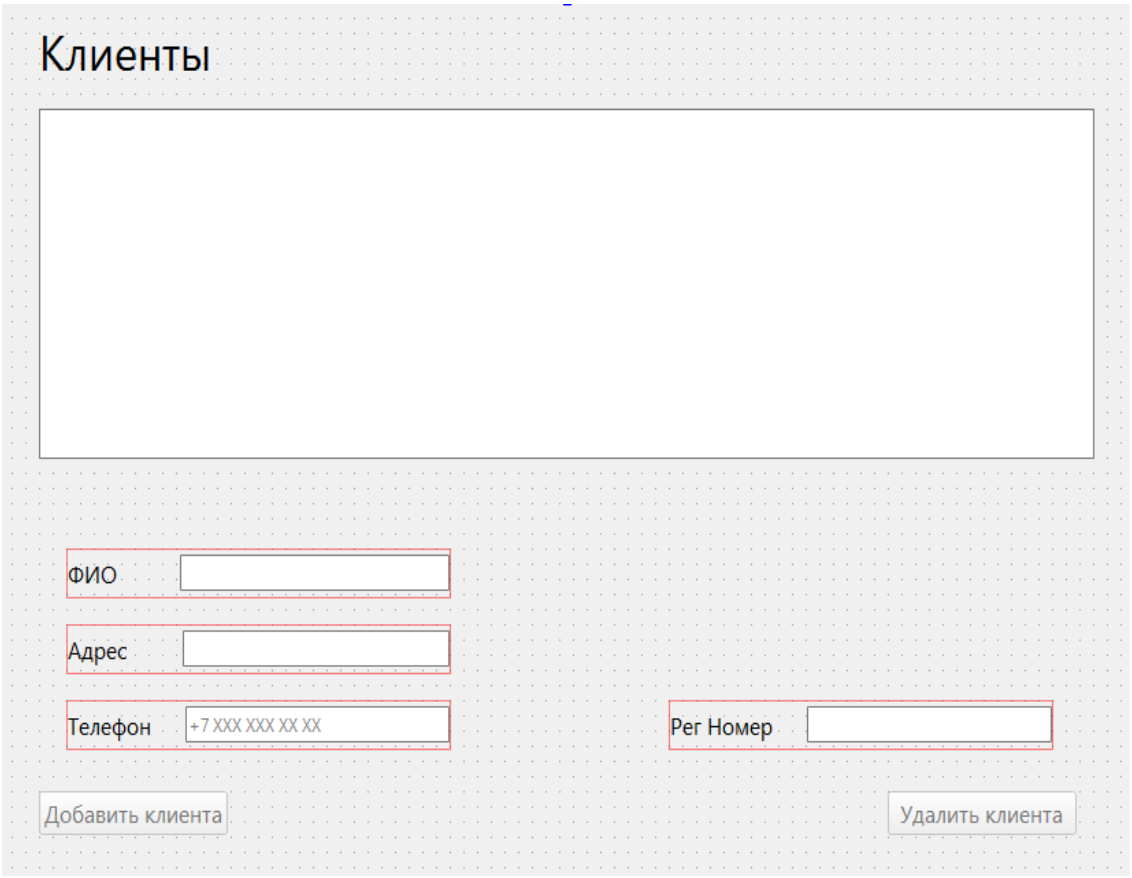


Рисунок 10 – Меню клиенты

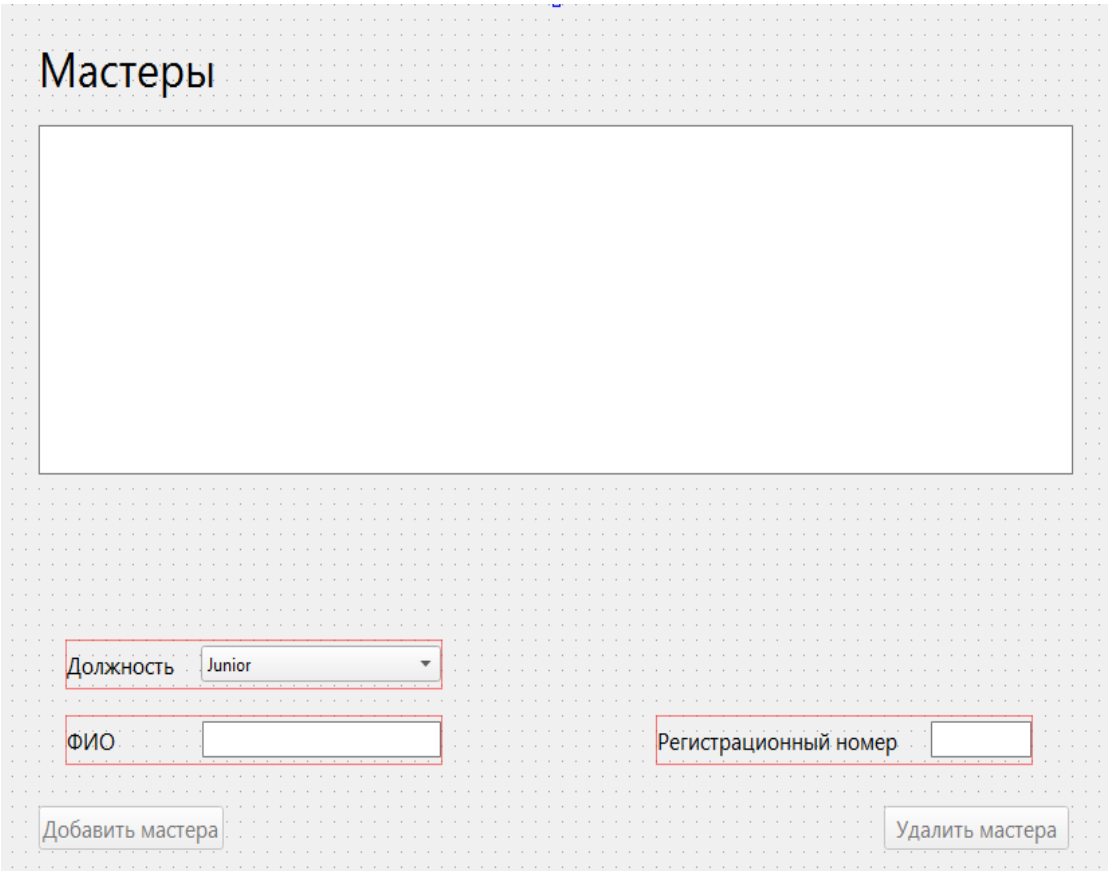


Рисунок 11 – Меню заказы

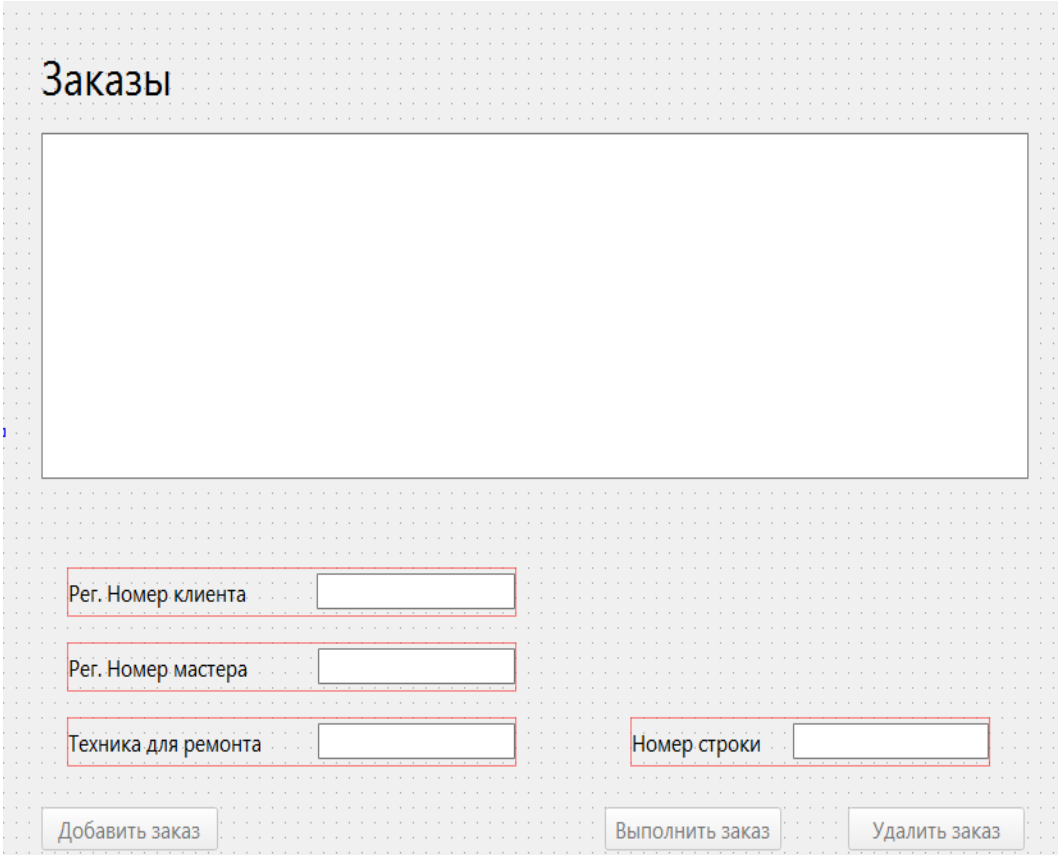


Рисунок 12 – Меню мастеров

5. **Описание формы**

Форма главное меню mainwindow

Qlabel – MainTitle – Главное название

Qlabel – SubTitle – Надпись версии

QHBoxLayout – MainButtons – Кнопки меню

- QPushButton – ClientsButton – Меню клиенты
- QPushButton – MastersButton – Меню мастеры
- QPushButton - OrdersButton – Меню заказы

Форма меню клиенты clientsmenu

MainTitle – Qlabel – Основное название

QTableView – ClientTable – Таблица вывода

QPushButton – AddClient – Кнопка добавить клиента

QPushButton – RemoveClient - Кнопка удалить клиента

QHBoxLayout – AdressLayout – Группа элементов ввода адреса

- QLineEdit – AdressEdit – Ввод адреса
- Qlabel – AdressLabel – Название ввода адреса

QHBoxLayout – FIOLayout - Группа элементов ввода фио

- QLineEdit – FIOEdit – Ввод ФИО
- Qlabel – FIOLabel – Название ввода ФИО

QHBoxLayout – PhoneLayout - Группа элементов ввода телефона

- QLineEdit – PhoneEdit – Ввод телефона
- Qlabel – PhoneLabel – Название ввода телефона

QHBoxLayout – RegLayout - Группа элементов ввода рег. номера

- QLineEdit – RegEdit – Ввод рег. номера
- Qlabel – RegLabel – Название ввода рег. номера

Форма меню клиенты mastersmenu

MainTitle – Qlabel – Основное название

QTableView – MastersTable – Таблица вывода

QPushButton – AddMaster – Кнопка добавить мастера

QPushButton – RemoveMaster - Кнопка удалить мастера

QHBoxLayout – FIOLayout - Группа элементов ввода фио

- QLineEdit – FIOEdit – Ввод ФИО

- Qlabel – FIOLabel – Название ввода ФИО

QHBoxLayout – PositionLayout - Группа элементов выбора должности

- QComboBox– PositionCombo – Выбор должности
- Qlabel – PositionLabel – Название выбора должности

QHBoxLayout – RegLayout - Группа элементов ввода рег. номера

- QLineEdit – RegEdit – Ввод рег. номера
- Qlabel – RegLabel – Название ввода рег. номера

Форма меню клиенты ordersmenu

MainTitle – Qlabel – Основное название

QTableView – OrdersTable – Таблица вывода

QPushButton – AddOrder – Кнопка добавить заказ

QPushButton – RemoveOrder - Кнопка удалить заказ

QPushButton – ExecuteOrder - Кнопка выполнить заказ

QHBoxLayout – RegNumClientLayout - Группа элементов ввода рег. Номера клиента

- QLineEdit – RegNumClientEdit – Ввод рег. номера
- Qlabel – RegNumClientLabel – Название ввода рег. номера

QHBoxLayout – RegNumMasterLayout - Группа элементов ввода рег. Номера мастера

- QLineEdit – RegNumMasterEdit – Ввод рег. номера
- Qlabel – RegNumMasterLabel – Название ввода рег. номера

QHBoxLayout – ThingLayout - Группа элементов ввода вещи

- QLineEdit – ThingEdit – Ввод названия вещи
- Qlabel – ThingLabel – Название ввода вещи

QHBoxLayout – NumOfRowLayout - Группа элементов ввода номера строки

- QLineEdit – NumOfRowEdit – Ввод номера строки
- Qlabel – NumOfRowLabel – Название ввода номера строки

6. Листинг программы

Main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
```

```
{
    QApplication a(argc, argv);
    MainWindow w;

    w.show();
    return a.exec();
}
```

Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "clientsmenu.h"
#include "ordermanipulate.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    CM = new ClientsMenu(this);
    OM = new OrdersMenu(this);
    MM = new MastersMenu(this);

    connect(ui->ClientsButton, &QPushButton::clicked, this,
&MainWindow::MainClientsClicked);
    connect(ui->OrdersButton, &QPushButton::clicked, this,
&MainWindow::MainOrdersClicked);
    connect(ui->MastersButton, &QPushButton::clicked, this,
&MainWindow::MainMastersClicked);

    connect(OM, &OrdersMenu::RequestToAddOrder, this, &MainWindow::AddOrder);
    connect(OM, &OrdersMenu::RequestToExecuteOrder, this,
&MainWindow::ExecuteOrder);
    connect(OM, &OrdersMenu::RequestToRemoveOrder, this,
&MainWindow::RemoveOrder);
```

```
    connect(CM, &ClientsMenu::RequestToRemoveClient, this,
&MainWindow::RemoveClient);
    connect(MM, &MastersMenu::RequestToRemoveMaster, this,
&MainWindow::RemoveMaster);
}

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

void MainWindow::AddOrder(Order* OrderInfo)
{
    OrderManipulate::AddOrder(OrderInfo, this);
}

void MainWindow::ExecuteOrder(int OrderNumber)
{
    OrderManipulate::ExecuteOrder(OrderNumber, this);
}

void MainWindow::RemoveOrder(int OrderNumber)
{
    OrderManipulate::RemoveOrder(OrderNumber, this);
}

void MainWindow::RemoveClient(int RegNumber)
{
    OrderManipulate::RemoveClient(RegNumber, this);
}

void MainWindow::RemoveMaster(QString RegCode)
{
    OrderManipulate::RemoveMaster(RegCode, this);
}
```

```
ClientsMenu* MainWindow::GetClientsMenu()
{
   return this->CM;
}

OrdersMenu* MainWindow::GetOrdersMenu()
{
   return this->OM;
}

MastersMenu* MainWindow::GetMastersMenu()
{
   return this->MM;
}

void MainWindow::MainClientsClicked()
{
   CM->show();
}

void MainWindow::MainOrdersClicked()
{
   OM->show();
}

void MainWindow::MainMastersClicked()
{
   MM->show();
}
```

Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
```

```
#include <QMainWindow>
#include "clientsmenu.h"
#include "mastersmenu.h"
#include "ordersmenu.h"

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    ClientsMenu* GetClientsMenu();
    OrdersMenu* GetOrdersMenu();
    MastersMenu* GetMastersMenu();

private:
    Ui::MainWindow* ui;
    ClientsMenu* CM;
    OrdersMenu* OM;
    MastersMenu* MM;

private slots:
    void AddOrder(Order*);
    void ExecuteOrder(int);
    void RemoveOrder(int);
    void RemoveClient(int);
    void RemoveMaster(QString);
```

```
    void MainClientsClicked();
    void MainOrdersClicked();
    void MainMastersClicked();
};
#endif // MAINWINDOW_H
```

Client.cpp

```
#include "client.h"


Client::Client(QObject *parent, QString FIO, QString Adress, QString PhoneNumber)
    : QObject{parent}
{
    this->FIO = FIO;
    this->Adress = Adress;
    this->PhoneNumber = PhoneNumber;

    this->RegNumber = Client::PhoneNumberToRegNumber(PhoneNumber);
}

QString Client::GetFIO()
{
    return this->FIO;
}

QString Client::GetAdress()
{
    return this->Adress;
}

QString Client::GetPhoneNumber()
{
    return this->PhoneNumber;
}

int Client::GetRegNumber()
```

```
{
    return this->RegNumber;
}

int Client::PhoneNumberToRegNumber(QString PhoneNumber)
{
    QString NumberToSum = PhoneNumber.replace( " ", "" ).remove('+');
    int OutSum = 0;
    for (auto number : std::as_const(NumberToSum) )
    {
        OutSum += pow(number.digitValue(), 2);
    }
    return OutSum;
}
```

Client.h

```
#ifndef CLIENT_H
#define CLIENT_H

#include <QObject>

class Client : public QObject
{
    Q_OBJECT
public:
    explicit Client(QObject *parent = nullptr, QString FIO = "", QString Adress="",
QString PhoneNumber="");
    static int PhoneNumberToRegNumber(QString);

    int GetRegNumber();
    QString GetFIO();
    QString GetAdress();
    QString GetPhoneNumber();

private:
    int RegNumber;
```

```
    QString FIO;
    QString Adress;
    QString PhoneNumber;

signals:
};

#endif // CLIENT_H
```

Order.cpp

```
#include "order.h"

Order::Order(QObject *parent, int ClientRegNumber,
             QString MasterRegCode, QString ThingToRepair)
    : QObject{parent}
{
    this->ClientRegNumber = ClientRegNumber;
    this->MasterRegCode = MasterRegCode;
    this->ThingToRepair = ThingToRepair;
    this->StatusOfRepair = RepairStatus::InProcess;
    this->OrderNumber = -1;
}

void Order::SetOrderNumber(int OrderNum)
{
    this->OrderNumber = OrderNum;
}

void Order::SetRepairStatus(RepairStatus Status)
{
    this->StatusOfRepair = Status;
}

int Order::GetClientRegNumber()
{
    return this->ClientRegNumber;
```

```
}

QString Order::GetMasterRegCode()
{
   return this->MasterRegCode;
}

QString Order::GetThingToRepair()
{
   return this->ThingToRepair;
}

QString Order::GetStatusOfRepair()
{
   switch (this->StatusOfRepair)
   {
   case RepairStatus::InProcess:
      return "В процессе";
      break;
   case RepairStatus::Completed:
      return "Выполнен";
      break;
   }
   return "";
}
```

Order.h

```
#ifndef ORDER_H
#define ORDER_H

#include <QObject>

class Order : public QObject
{
   Q_OBJECT
public:
```

```
    enum class RepairStatus
    {
        InProcess,
        Completed
    };

    explicit Order(QObject *parent = nullptr, int ClientRegNumber = 0,
                   QString MasterRegCode = "", QString ThingToRepair = "");
    void SetOrderNumber(int);
    void SetRepairStatus(RepairStatus);

    int GetClientRegNumber();
    QString GetMasterRegCode();
    QString GetThingToRepair();
    QString GetStatusOfRepair();

private:
    int ClientRegNumber;
    QString MasterRegCode;
    QString ThingToRepair;
    RepairStatus StatusOfRepair;
    int OrderNumber;
};

#endif // ORDER_H
```

Worker.cpp

```
#include "worker.h"
#include "qdebug.h"

Worker::Worker(QObject *parent, QString FIO, QString Position)
    : QObject{parent}
{
    this->FIO = FIO;
    this->Position = Position;
    this->MasterBusy = false;
```

```
    this->OrderNumber = -1;

    this->RegCode = Worker::FIOnPosToRegCode(FIO, Position);
}

void Worker::SetOrder(int OrderNumber)
{
    this->OrderNumber = OrderNumber;
    if (OrderNumber == -1)
    {
        this->MasterBusy = false;
    }
    else
    {
        this->MasterBusy = true;
    }
}

QString Worker::GetRegCode()
{
    return this->RegCode;
}

QString Worker::GetFIO()
{
    return this->FIO;
}

QString Worker::GetPosition()
{
    return this->Position;
}

bool Worker::IsBusy()
{
```

```
    return this->MasterBusy;
}

QString Worker::GetOrderStatus()
{
    if ((this->OrderNumber == -1) && not(this->MasterBusy))
    {
        return "Нет";
    }
    return "Есть";
}

QString Worker::FIOnPosToRegCode(QString FIO, QString Position)
{
    QString RegCode = "";
    if (Position == "Intern")
    {
        RegCode += "IN-";
    }
    else if (Position == "Junior")
    {
        RegCode += "JN-";
    }
    else if (Position == "Middle")
    {
        RegCode += "ML-";
    }
    else if (Position == "Senior")
    {
        RegCode += "SN-";
    }
    else
    {
        RegCode += "OT-";
    }
```

```
    int tempCode = 0;
    FIO = FIO.toUpper();
    foreach (auto str, FIO.split(" ")){
        tempCode += int(str[0].unicode());
    }
    return RegCode + QString::number(tempCode);
}
```

Worker.h

```
#ifndef WORKER_H
#define WORKER_H

#include <QObject>

class Worker : public QObject
{
    Q_OBJECT
public:
    explicit Worker(QObject *parent = nullptr, QString FIO="", QString Position="");
    static QString FIOnPosToRegCode(QString, QString);

    void SetOrder(int);

    QString GetRegCode();
    QString GetFIO();
    QString GetPosition();
    bool IsBusy();
    QString GetOrderStatus();

private:
    QString RegCode;
    QString FIO;
    QString Position;
    bool MasterBusy;
    int OrderNumber;
```

```
};

#endif // WORKER_H
```

Clientsmenu.cpp

```
#include "clientsmenu.h"
#include "client.h"
#include "ui_clientsmenu.h"
#include "ordermanipulate.h"
#include <QStandardItemModel>

ClientsMenu::ClientsMenu(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::ClientsMenu)
{
    ui->setupUi(this);
    this->setWindowFlag(Qt::Window);

    this->NumOfClients = 0;
    auto ClientTable = ui->ClientTable;
    ClientTable->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ClientTable->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

    model = new QStandardItemModel(0, 4);
    SetModelHeader();

    ClientTable->setModel(model);
    ui->PhoneEdit->setInputMask("+7 999 999 99-99;_");
    ui->RegEdit->setValidator(new QIntValidator(0, 810, this));

    AddNewClientToModel(new Client(nullptr, "ОО ВА К", "ул. Гастелло 12344", "+7
946 535 54-35"));
    AddNewClientToModel(new Client(nullptr, "ОО ВА К", "ул. Гастелло 12344", "+7
945 541 54-35"));
    AddNewClientToModel(new Client(nullptr, "ОО ВА К", "ул. Гастелло 12344", "+7
435 435 58-85"));
```

```
    AddNewClientToModel(new Client(nullptr, "ОО ВА К", "ул. Гастелло 12344", "+7
435 795 58-85"));

    connect(ui->AddClient, &QPushButton::clicked, this,
&ClientsMenu::RequestToAddClient);
    connect(ui->RemoveClient, &QPushButton::clicked, this, [this] {emit
RequestToRemoveClient(ui->RegEdit->text().toInt());});
    connect(ui->FIOEdit, &QLineEdit::textChanged, this, &ClientsMenu::InfoChanged);
    connect(ui->AdressEdit, &QLineEdit::textChanged, this,
&ClientsMenu::InfoChanged);
    connect(ui->PhoneEdit, &QLineEdit::textChanged, this, &ClientsMenu::InfoChanged);
    connect(ui->RegEdit, &QLineEdit::textChanged, this, &ClientsMenu::RegChanged);
}

ClientsMenu::~ClientsMenu()
{
    auto ListBegin = ClientsList.begin();
    auto ListEnd = ClientsList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        delete *ListBegin;

    }
    model->clear();
    delete model;
    delete ui;
}

void ClientsMenu::SetModelHeader()
{
    model->setHeaderData(0, Qt::Horizontal, "Регистрационный номер");
    model->setHeaderData(1, Qt::Horizontal, "ФИО");
    model->setHeaderData(2, Qt::Horizontal, "Адрес");
    model->setHeaderData(3, Qt::Horizontal, "Телефон");
}
```

```
void ClientsMenu::ClearInputs()
{
   ui->AddClient->setDisabled(true);
   ui->RemoveClient->setDisabled(true);

   ui->FIOEdit->clear();
   ui->AdressEdit->clear();
   ui->PhoneEdit->clear();
   ui->RegEdit->clear();
}

void ClientsMenu::AddNewClientToModel(Client* ClientInfo)
{
   int IndexForAdd = ++this->NumOfClients-1;
   model->setItem(IndexForAdd, 0, new QStandardItem(QString::number(ClientInfo-
>GetRegNumber())));
   model->setItem(IndexForAdd, 1, new QStandardItem(ClientInfo->GetFIO()));
   model->setItem(IndexForAdd, 2, new QStandardItem(ClientInfo->GetAdress()));
   model->setItem(IndexForAdd, 3, new QStandardItem(ClientInfo-
>GetPhoneNumber()));
   ClientsList.append(ClientInfo);
}

void ClientsMenu::RemoveClientFromModel(int RegNumber)
{
   auto ListBegin = ClientsList.begin();
   auto ListEnd = ClientsList.end();
   QList<Client*>::iterator finded = ListEnd;
   for (;ListBegin<ListEnd; ListBegin++)
   {
      if ((*ListBegin)->GetRegNumber() == RegNumber)
      {
         finded = ListBegin;
         break;
```

```
        }
    }
    int IndexOfFinded = finded-ClientsList.begin();
    if (finded != ListEnd)
    {
        ClientsList.erase(finded);
        model->removeRow(IndexOfFinded);
        this->NumOfClients--;
    }
}

bool ClientsMenu::IsClientInList(int RegNumber)
{
    auto ListBegin = ClientsList.begin();
    auto ListEnd = ClientsList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        if ((*ListBegin)->GetRegNumber() == RegNumber)
        {
            return true;
        }
    }
    return false;
}

void ClientsMenu::RequestToAddClient()
{
    QString PhoneNumber = ui->PhoneEdit->text();
    if (not(IsClientInList(Client::PhoneNumberToRegNumber(PhoneNumber))))
    {
        AddNewClientToModel(new Client(nullptr, ui->FIOEdit->text(), ui->AdressEdit-
>text(), PhoneNumber));
    }
    else
    {
```

```
        OrderManipulate::SendErrorMessage("Такой клиент уже существует.");
        return;
    }
    ClearInputs();
}

void ClientsMenu::InfoChanged()
{
    if (ui->PhoneEdit->hasAcceptableInput() && not(ui->FIOEdit->text().isEmpty()) &&
not(ui->AdressEdit->text().isEmpty()))
    {
        ui->AddClient->setDisabled(false);
    }
    else
    {
        ui->AddClient->setDisabled(true);
    }
}

void ClientsMenu::RegChanged()
{
    if (ui->RegEdit->hasAcceptableInput()  && NumOfClients>0)
    {
        ui->RemoveClient->setDisabled(false);
    }
    else
    {
        ui->RemoveClient->setDisabled(true);
    }
}
```

Clientsmenu.h

```
#ifndef CLIENTSMENU_H
#define CLIENTSMENU_H

#include "client.h"
```

```
#include "qstandarditemmodel.h"
#include <QWidget>

namespace Ui {
class ClientsMenu;
}

class ClientsMenu : public QWidget
{
    Q_OBJECT

public:
    explicit ClientsMenu(QWidget *parent = nullptr);
    ~ClientsMenu();
    bool IsClientInList(int);
    void ClearInputs();
    void RemoveClientFromModel(int);

signals:
    void RequestToRemoveClient(int);

private:
    Ui::ClientsMenu* ui;
    int NumOfClients;
    QList<Client*> ClientsList;
    QStandardItemModel* model;

    void SetModelHeader();
    void AddNewClientToModel(Client*);


private slots:
    void RequestToAddClient();
    void InfoChanged();
    void RegChanged();
```

```
};

#endif // CLIENTSMENU_H
```

Mastersmenu.cpp

```
#include "mastersmenu.h"
#include "ordermanipulate.h"
#include "ui_mastersmenu.h"

MastersMenu::MastersMenu(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::MastersMenu)
{
    ui->setupUi(this);
    this->setWindowFlag(Qt::Window);

    model = new QStandardItemModel(0, 4);
    SetModelHeader();

    this->NumOfMasters = 0;
    auto MasterTable = ui->MastersTable;
    MasterTable->setEditTriggers(QAbstractItemView::NoEditTriggers);
    MasterTable->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

    MasterTable->setModel(model);
    ui->RegEdit->setInputMask(">AA-9000;_");

    AddNewMasterToModel(new Worker(nullptr, "AA eee BB", "Junior"));
    AddNewMasterToModel(new Worker(nullptr, "УУ ввa даку", "Junior"));
    AddNewMasterToModel(new Worker(nullptr, "павп кукпп п", "Senior"));
    AddNewMasterToModel(new Worker(nullptr, "аупау пук", "Middle"));

    connect(ui->AddMaster, &QPushButton::clicked, this,
&MastersMenu::RequestToAddMaster);
    connect(ui->RemoveMaster, &QPushButton::clicked, this, [this] {emit
RequestToRemoveMaster(ui->RegEdit->text());});
```

```
    connect(ui->PositionCombo, &QComboBox::currentTextChanged, this,
&MastersMenu::InfoChanged);
    connect(ui->FIOEdit, &QLineEdit::textChanged, this, &MastersMenu::InfoChanged);
    connect(ui->RegEdit, &QLineEdit::textChanged, this,
&MastersMenu::RegNumberChanged);
}

MastersMenu::~MastersMenu()
{
    auto ListBegin = MasterList.begin();
    auto ListEnd = MasterList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        delete *ListBegin;

    }
    model->clear();
    delete model;
    delete ui;
}

void MastersMenu::SetMasterBusyStatusByRegCode(QString RegCode, int
OrderNumber)
{
    auto ListBegin = MasterList.begin();
    auto ListEnd = MasterList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        if ((*ListBegin)->GetRegCode() == RegCode)
        {
            break;
        }
    }

    (*ListBegin)->SetOrder(OrderNumber);
```

```
    model->setItem(ListBegin-MasterList.begin(), 3, new QStandardItem((*ListBegin)-
>GetOrderStatus()));
}

void MastersMenu::SetModelHeader()
{
    model->setHeaderData(0, Qt::Horizontal, "Регистрационный номер");
    model->setHeaderData(1, Qt::Horizontal, "ФИО");
    model->setHeaderData(2, Qt::Horizontal, "Должность");
    model->setHeaderData(3, Qt::Horizontal, "Заказ");
}

void MastersMenu::ClearInputs()
{
    ui->AddMaster->setDisabled(true);
    ui->RemoveMaster->setDisabled(true);

    ui->FIOEdit->clear();
    ui->PositionCombo->setCurrentIndex(0);
    ui->RegEdit->clear();
}

void MastersMenu::AddNewMasterToModel(Worker* MasterInfo)
{
    int IndexForAdd = ++this->NumOfMasters-1;
    model->setItem(IndexForAdd, 0, new QStandardItem(MasterInfo->GetRegCode()));
    model->setItem(IndexForAdd, 1, new QStandardItem(MasterInfo->GetFIO()));
    model->setItem(IndexForAdd, 2, new QStandardItem(MasterInfo->GetPosition()));
    model->setItem(IndexForAdd, 3, new QStandardItem(MasterInfo->GetOrderStatus()));
    MasterList.append(MasterInfo);
}

void MastersMenu::RemoveMasterFromModel(QString RegCode)
{
    auto ListBegin = MasterList.begin();
```

```
    auto ListEnd = MasterList.end();
    QList<Worker*>::iterator finded = ListEnd;
    for (;ListBegin<ListEnd; ListBegin++)
    {
        if ((*ListBegin)->GetRegCode() == RegCode)
        {
            finded = ListBegin;
            break;
        }
    }
    int IndexOfFinded = finded-MasterList.begin();
    if (finded != ListEnd)
    {
        MasterList.erase(finded);
        model->removeRow(IndexOfFinded);
        this->NumOfMasters--;
    }
}

bool MastersMenu::IsMasterInList(QString RegCode)
{
    auto ListBegin = MasterList.begin();
    auto ListEnd = MasterList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        if ((*ListBegin)->GetRegCode() == RegCode)
        {
            return true;
        }
    }
    return false;
}

Worker* MastersMenu::GetMasterByRegCode(QString RegCode)
{
```

```
    auto ListBegin = MasterList.begin();
    auto ListEnd = MasterList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        if ((*ListBegin)->GetRegCode() == RegCode)
        {
            return *ListBegin;
        }
    }
    return nullptr;
}

void MastersMenu::RequestToAddMaster()
{
    QString FIO = ui->FIOEdit->text();
    QString Position = ui->PositionCombo->currentText();
    QString RegCode = Worker::FIOnPosToRegCode(FIO, Position);
    if (not(IsMasterInList(RegCode)))
    {
        AddNewMasterToModel(new Worker(nullptr, FIO, Position));
    }
    else
    {
        OrderManipulate::SendErrorMessage("Такой мастер уже существует.");
        return;
    }
    ClearInputs();
}

void MastersMenu::InfoChanged()
{
    if (not(ui->PositionCombo->currentText().isEmpty()) && not(ui->FIOEdit-
>text().isEmpty()) &&
        not(ui->FIOEdit->text().size() > 31))
    {
```

```
        ui->AddMaster->setDisabled(false);
    }
    else
    {
        ui->AddMaster->setDisabled(true);
    }
}

void MastersMenu::RegNumberChanged()
{
    if (ui->RegEdit->hasAcceptableInput() && NumOfMasters>0)
    {
        ui->RemoveMaster->setDisabled(false);
    }
    else
    {
        ui->RemoveMaster->setDisabled(true);
    }
}
```

Mastersmenu.h

```
#ifndef MASTERSMENU_H
#define MASTERSMENU_H

#include "qstandarditemmodel.h"
#include "worker.h"
#include <QWidget>

namespace Ui {
class MastersMenu;
}

class MastersMenu : public QWidget
{
    Q_OBJECT
```

```
public:
    explicit MastersMenu(QWidget *parent = nullptr);
    ~MastersMenu();
    bool IsMasterInList(QString);
    void SetMasterBusyStatusByRegCode(QString, int);
    Worker* GetMasterByRegCode(QString);
    void ClearInputs();
    void RemoveMasterFromModel(QString);

signals:
    void RequestToRemoveMaster(QString);

private:
    Ui::MastersMenu* ui;
    int NumOfMasters;
    QList<Worker*> MasterList;
    QStandardItemModel* model;

    void SetModelHeader();
    void AddNewMasterToModel(Worker*);

private slots:
    void InfoChanged();
    void RegNumberChanged();
    void RequestToAddMaster();
};

#endif // MASTERSMENU_H
```

Ordersmenu.cpp

```
#include "ordersmenu.h"
#include "ui_ordersmenu.h"

OrdersMenu::OrdersMenu(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::OrdersMenu)
```

```
{
    ui->setupUi(this);
    this->setWindowFlag(Qt::Window);

    this->NumOfOrders = 0;
    auto OrdersTable = ui->OrdersTable;
    OrdersTable->setEditTriggers(QAbstractItemView::NoEditTriggers);
    OrdersTable->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

    model = new QStandardItemModel(0, 4);
    SetModelHeader();
    OrdersTable->setModel(model);

    ui->RegNumClientEdit->setValidator(new QIntValidator(0, 810, this));
    ui->RegNumMasterEdit->setInputMask(">AA-9000;_");
    auto Validator = new QIntValidator(this);
    Validator->setBottom(1);
    ui->NumOfRowEdit->setValidator(Validator);

    connect(ui->RegNumClientEdit, &QLineEdit::textChanged, this,
&OrdersMenu::InfoChanged);
    connect(ui->RegNumMasterEdit, &QLineEdit::textChanged, this,
&OrdersMenu::InfoChanged);
    connect(ui->ThingEdit, &QLineEdit::textChanged, this, &OrdersMenu::InfoChanged);
    connect(ui->NumOfRowEdit, &QLineEdit::textChanged, this,
&OrdersMenu::RowNumberChanged);

    connect(ui->AddOrder, &QPushButton::clicked, this, [this] {
        emit RequestToAddOrder(new Order(nullptr, ui->RegNumClientEdit->text().toInt(),
ui->RegNumMasterEdit->text(), ui->ThingEdit->text()));});
    connect(ui->ExecuteOrder, &QPushButton::clicked, this, [this] {emit
RequestToExecuteOrder(ui->NumOfRowEdit->text().toInt());});
    connect(ui->RemoveOrder, &QPushButton::clicked, this, [this] {emit
RequestToRemoveOrder(ui->NumOfRowEdit->text().toInt());});
}
```

```
OrdersMenu::~OrdersMenu()
{
    auto ListBegin = OrderList.begin();
    auto ListEnd = OrderList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        delete *ListBegin;

    }
    model->clear();
    delete model;
    delete ui;
}

bool OrdersMenu::HasSomeoneNotcompletedOrder(int RegNumber)
{
    auto ListBegin = OrderList.begin();
    auto ListEnd = OrderList.end();
    for (;ListBegin<ListEnd; ListBegin++)
    {
        if ((*ListBegin)->GetClientRegNumber() == RegNumber && (*ListBegin)-
>GetStatusOfRepair() == "В процессе")
        {
            return true;
        }
    }
    return false;
}

bool OrdersMenu::HasSomeoneNotcompletedOrder(QString RegCode)
{
    auto ListBegin = OrderList.begin();
    auto ListEnd = OrderList.end();
    for (;ListBegin<ListEnd; ListBegin++)
```

```
    {
        if ((*ListBegin)->GetMasterRegCode() == RegCode && (*ListBegin)-
>GetStatusOfRepair() == "В процессе")
        {
            return true;
        }
    }
    return false;
}


int OrdersMenu::GetSizeOfList()
{
    return this->NumOfOrders;
}

QString OrdersMenu::GetMasterRegCodeByValue(int OrderNumber)
{
    auto OrderInfo = OrderList.at(OrderNumber-1);
    return OrderInfo->GetMasterRegCode();
}

QStandardItemModel* OrdersMenu::GetModelOfMenu()
{
    return model;
}

void OrdersMenu::SetModelHeader()
{
    model->setHeaderData(0, Qt::Horizontal, "Номер клиента");
    model->setHeaderData(1, Qt::Horizontal, "Номер мастера");
    model->setHeaderData(2, Qt::Horizontal, "Объект заказа");
    model->setHeaderData(3, Qt::Horizontal, "Статус");
}
```

```
void OrdersMenu::ClearInputs()
{
   ui->AddOrder->setDisabled(true);
   ui->RemoveOrder->setDisabled(true);

   ui->RegNumClientEdit->clear();
   ui->RegNumMasterEdit->clear();
   ui->ThingEdit->clear();
   ui->NumOfRowEdit->clear();
}

void OrdersMenu::AddNewOrderToModel(Order* OrderInfo)
{
   int IndexForAdd = ++this->NumOfOrders-1;
   model->setItem(IndexForAdd, 0, new QStandardItem(QString::number(OrderInfo-
>GetClientRegNumber())));
   model->setItem(IndexForAdd, 1, new QStandardItem(OrderInfo-
>GetMasterRegCode()));
   model->setItem(IndexForAdd, 2, new QStandardItem(OrderInfo-
>GetThingToRepair()));
   model->setItem(IndexForAdd, 3, new QStandardItem(OrderInfo-
>GetStatusOfRepair()));
   OrderList.append(OrderInfo);
}

void OrdersMenu::RemoveOrder(int OrderNumber)
{
   OrderList.removeAt(OrderNumber-1);
   model->removeRow(OrderNumber-1);
   this->NumOfOrders--;
}

void OrdersMenu::SetOrderAsCompleted(int OrderNumber)
{
```

```
    auto OrderInfo = OrderList.at(OrderNumber-1);
    OrderInfo->SetOrderNumber(-1);
    OrderInfo->SetRepairStatus(Order::RepairStatus::Completed);
    model->setItem(OrderNumber-1, 3, new QStandardItem(OrderInfo-
>GetStatusOfRepair()));
}

void OrdersMenu::InfoChanged()
{
    if (ui->RegNumClientEdit->hasAcceptableInput() && ui->RegNumMasterEdit-
>hasAcceptableInput() &&
        not(ui->ThingEdit->text().isEmpty()))
    {
        ui->AddOrder->setDisabled(false);
    }
    else
    {
        ui->AddOrder->setDisabled(true);
    }
}

void OrdersMenu::RowNumberChanged()
{
    if (ui->NumOfRowEdit->hasAcceptableInput() && NumOfOrders>0)
    {
        ui->RemoveOrder->setDisabled(false);
        ui->ExecuteOrder->setDisabled(false);
    }
    else
    {
        ui->RemoveOrder->setDisabled(true);
        ui->ExecuteOrder->setDisabled(true);
    }
}
```

Ordersmenu.h

```
#ifndef ORDERSMENU_H
#define ORDERSMENU_H


#include "order.h"
#include "qstandarditemmodel.h"
#include <QWidget>

namespace Ui {
class OrdersMenu;
}

class OrdersMenu : public QWidget
{
    Q_OBJECT

public:
    explicit OrdersMenu(QWidget *parent = nullptr);
    ~OrdersMenu();

    void AddNewOrderToModel(Order*);
    void RemoveOrder(int);
    void SetOrderAsCompleted(int);
    int GetSizeOfList();
    QString GetMasterRegCodeByValue(int);
    QStandardItemModel* GetModelOfMenu();
    bool HasSomeoneNotcompletedOrder(int);
    bool HasSomeoneNotcompletedOrder(QString);
    void ClearInputs();

signals:
    void RequestToAddOrder(Order*);
    void RequestToExecuteOrder(int);
    void RequestToRemoveOrder(int);
```

```
private:
    Ui::OrdersMenu* ui;
    int NumOfOrders;
    QList<Order*> OrderList;
    QStandardItemModel* model;

    void SetModelHeader();

private slots:
    void InfoChanged();
    void RowNumberChanged();
};

#endif // ORDERSMENU_H
```

Ordermanipulate.cpp

```
#include "ordermanipulate.h"
#include "qmessagebox.h"

OrderManipulate::OrderManipulate(QWidget *parent)
    : QWidget{parent}
{}

void OrderManipulate::SendErrorMessage(QString ErrorMessage)
{
    QMessageBox msgBox;
    msgBox.setText(ErrorMessage);
    msgBox.setIcon(QMessageBox::Critical);
    msgBox.exec();
}

void OrderManipulate::AddOrder(Order* OrderInfo, MainWindow* MainInfo)
{
    QString MasterRegCode = OrderInfo->GetMasterRegCode();
    if (not(MainInfo->GetClientsMenu()->IsClientInList(OrderInfo-
>GetClientRegNumber())))
```

```
    {
        OrderManipulate::SendErrorMessage("Такого клиента не существует.");

        delete OrderInfo;
        return;
    }
    if (not(MainInfo->GetMastersMenu()->IsMasterInList(MasterRegCode)))
    {
        OrderManipulate::SendErrorMessage("Такого мастера не существует.");

        delete OrderInfo;
        return;
    }
    if (MainInfo->GetMastersMenu()->GetMasterByRegCode(MasterRegCode)-
>IsBusy())
    {
        OrderManipulate::SendErrorMessage("Мастер уже занят.");

        delete OrderInfo;
        return;
    }
    MainInfo->GetOrdersMenu()->AddNewOrderToModel(OrderInfo);
    MainInfo->GetMastersMenu()->SetMasterBusyStatusByRegCode(MasterRegCode,
                                        MainInfo->GetOrdersMenu()->GetSizeOfList());
    MainInfo->GetOrdersMenu()->ClearInputs();
}

void OrderManipulate::ExecuteOrder(int OrderNumber, MainWindow* MainInfo)
{

    if (OrderNumber > MainInfo->GetOrdersMenu()->GetSizeOfList())
    {
        OrderManipulate::SendErrorMessage("Такого заказа не существует.");

        return;
```

```
    }
    if (MainInfo->GetOrdersMenu()->GetModelOfMenu()->item(OrderNumber-1, 3)-
>text() == "Выполнен")
    {
        OrderManipulate::SendErrorMessage("Данный заказ уже выполнен.");

        return;
    }

    MainInfo->GetMastersMenu()->SetMasterBusyStatusByRegCode(MainInfo-
>GetOrdersMenu()->GetMasterRegCodeByValue(OrderNumber), -1);
    MainInfo->GetOrdersMenu()->SetOrderAsCompleted(OrderNumber);
    MainInfo->GetOrdersMenu()->ClearInputs();
}

void OrderManipulate::RemoveOrder(int OrderNumber, MainWindow* MainInfo)
{
    if (OrderNumber > MainInfo->GetOrdersMenu()->GetSizeOfList())
    {
        OrderManipulate::SendErrorMessage("Такого заказа не существует.");

        return;
    }
    if (MainInfo->GetOrdersMenu()->GetModelOfMenu()->item(OrderNumber-1, 3)-
>text() == "В процессе")
    {
        OrderManipulate::SendErrorMessage("Выполните заказ перед удалением.");

        return;
    }
    MainInfo->GetOrdersMenu()->RemoveOrder(OrderNumber);
    MainInfo->GetOrdersMenu()->ClearInputs();
}

void OrderManipulate::RemoveClient(int RegNumber, MainWindow* MainInfo)
```

```
{
    if (not(MainInfo->GetClientsMenu()->IsClientInList(RegNumber)))
    {
        OrderManipulate::SendErrorMessage("Такого клиента не существует.");

        return;
    }

    if (MainInfo->GetOrdersMenu()->HasSomeoneNotcompletedOrder(RegNumber))
    {
        OrderManipulate::SendErrorMessage("У клиента есть не выполненный заказ.");

        return;
    }

    MainInfo->GetClientsMenu()->RemoveClientFromModel(RegNumber);
    MainInfo->GetClientsMenu()->ClearInputs();
}

void OrderManipulate::RemoveMaster(QString RegCode, MainWindow* MainInfo)
{
    if (not(MainInfo->GetMastersMenu()->IsMasterInList(RegCode)))
    {
        OrderManipulate::SendErrorMessage("Такого мастера не существует.");

        return;
    }
    if (MainInfo->GetOrdersMenu()->HasSomeoneNotcompletedOrder(RegCode))
    {
        OrderManipulate::SendErrorMessage("У мастера есть не выполненный заказ.");

        return;
    }
    MainInfo->GetMastersMenu()->RemoveMasterFromModel(RegCode);
    MainInfo->GetMastersMenu()->ClearInputs();
```

```
}
```

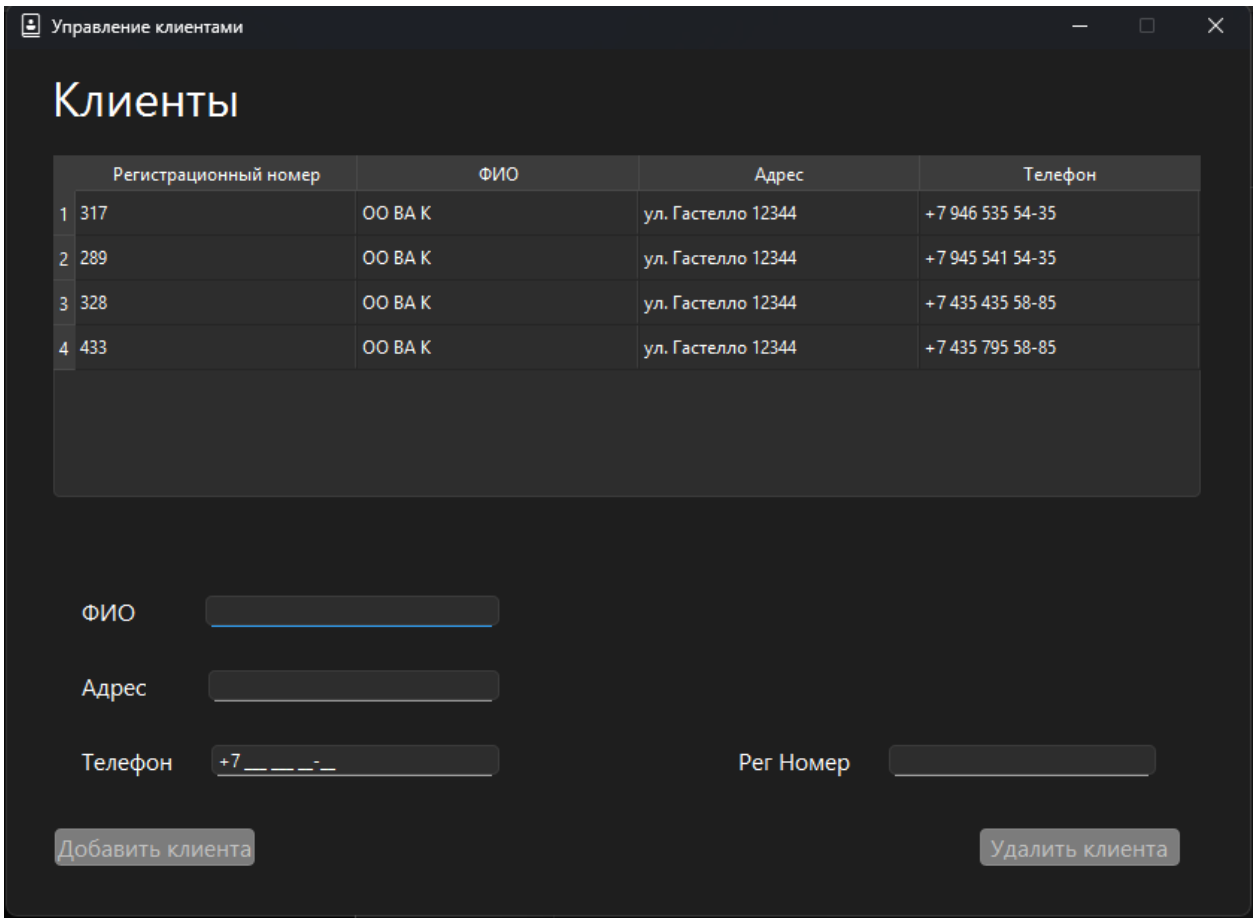
Ordermanipulate.h

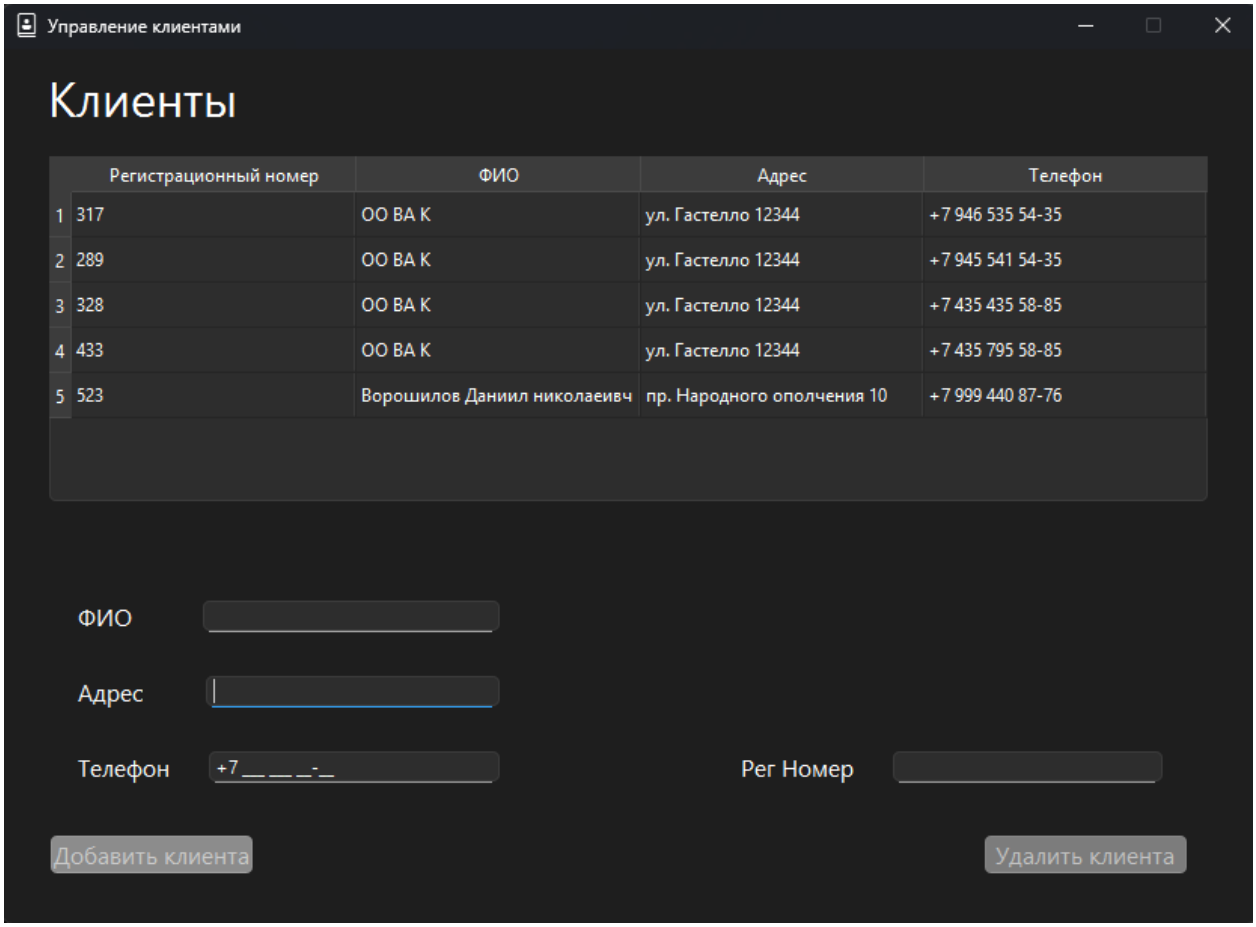
```
#ifndef ORDERMANIPULATE_H
#define ORDERMANIPULATE_H

#include "mainwindow.h"
#include "order.h"
#include <QWidget>

class OrderManipulate : public QWidget
{
   Q_OBJECT
public:
   explicit OrderManipulate(QWidget *parent = nullptr);
   static void SendErrorMessage(QString);
   static void AddOrder(Order*, MainWindow*);
   static void ExecuteOrder(int, MainWindow*);
   static void RemoveOrder(int, MainWindow*);
   static void RemoveClient(int, MainWindow*);
   static void RemoveMaster(QString, MainWindow*);

signals:
};

#endif // ORDERMANIPULATE_H
```

7. Работа программы

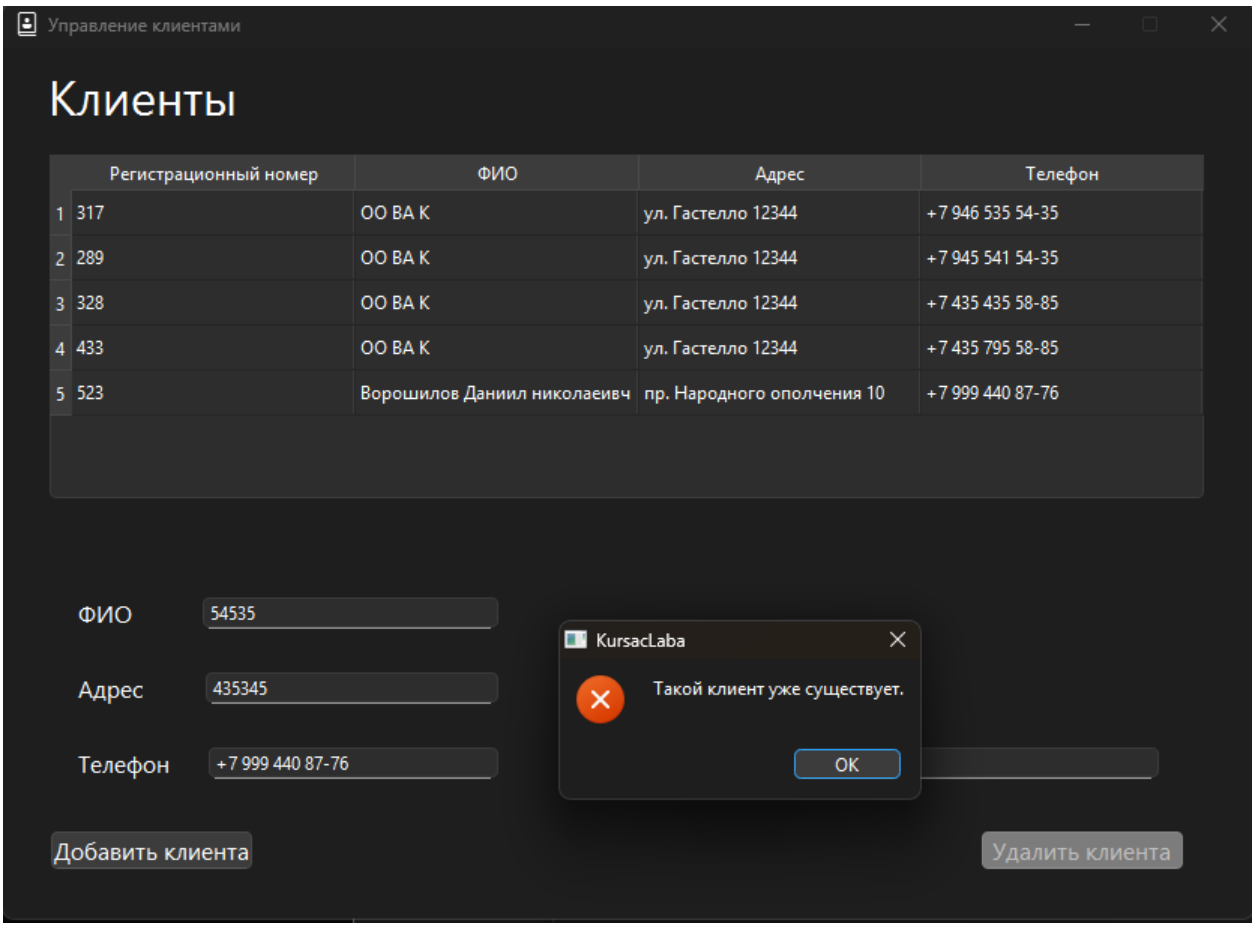
   На рисунках ниже представлены всевозможные обработки приложения

   Добавление клиента до/после:

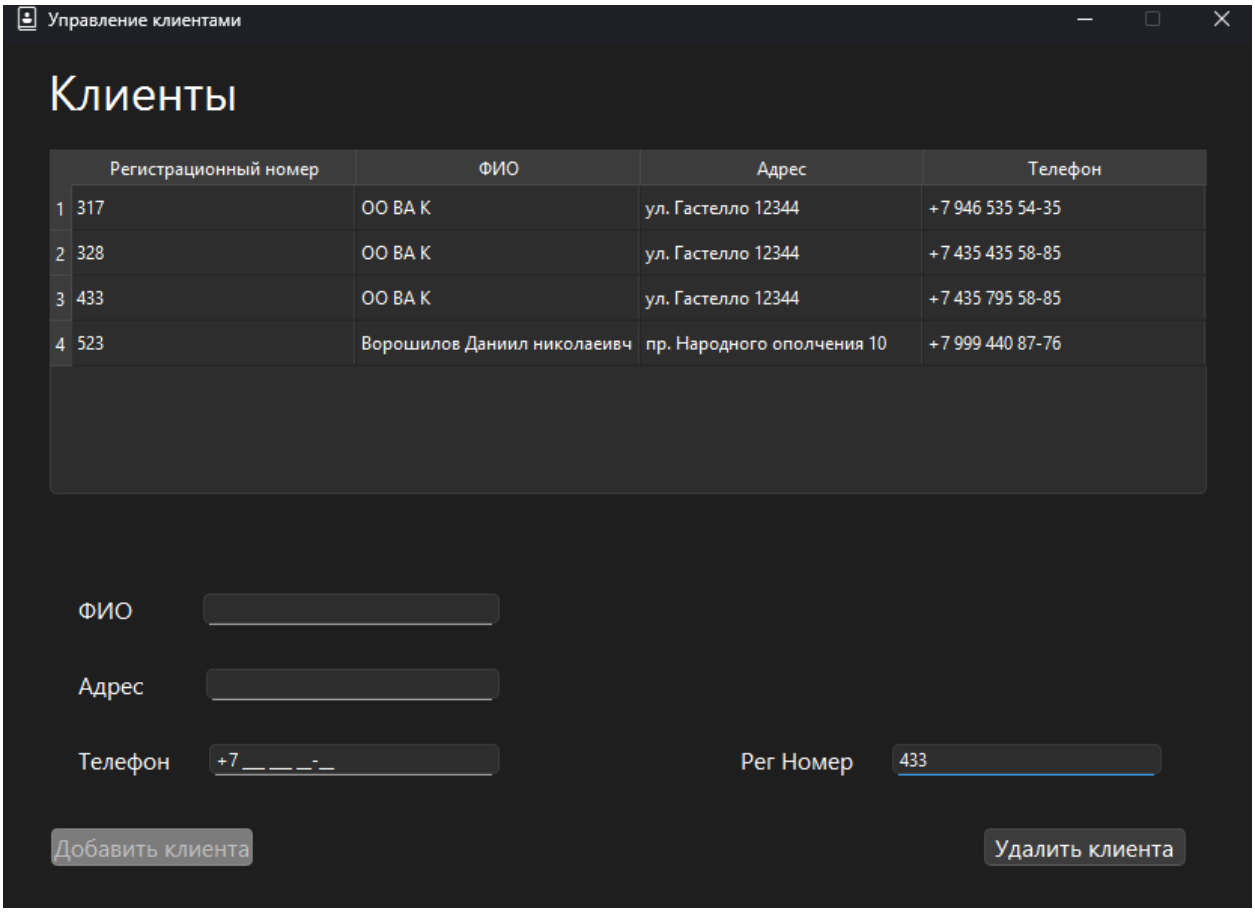


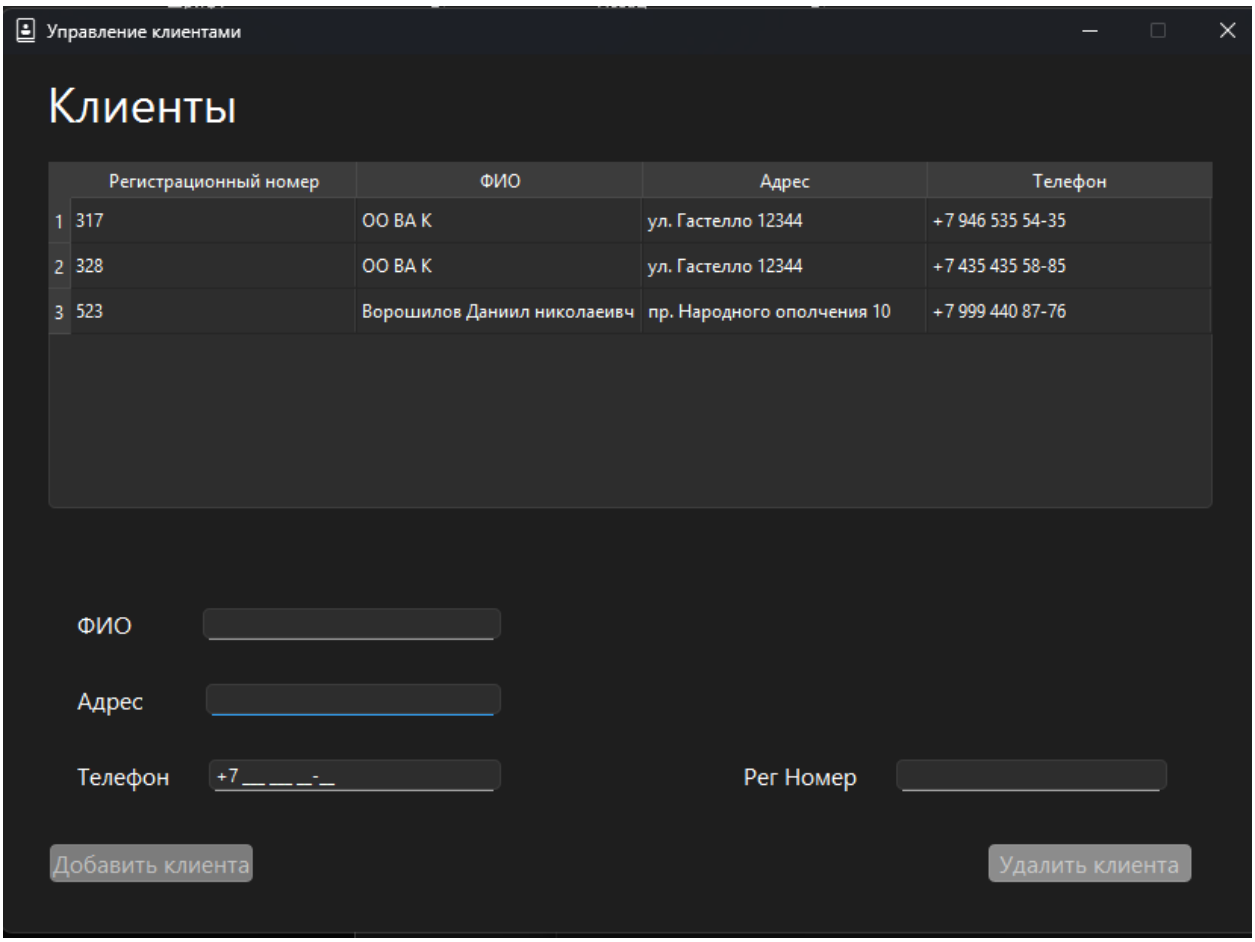


Добавление клиента по существующему номеру:

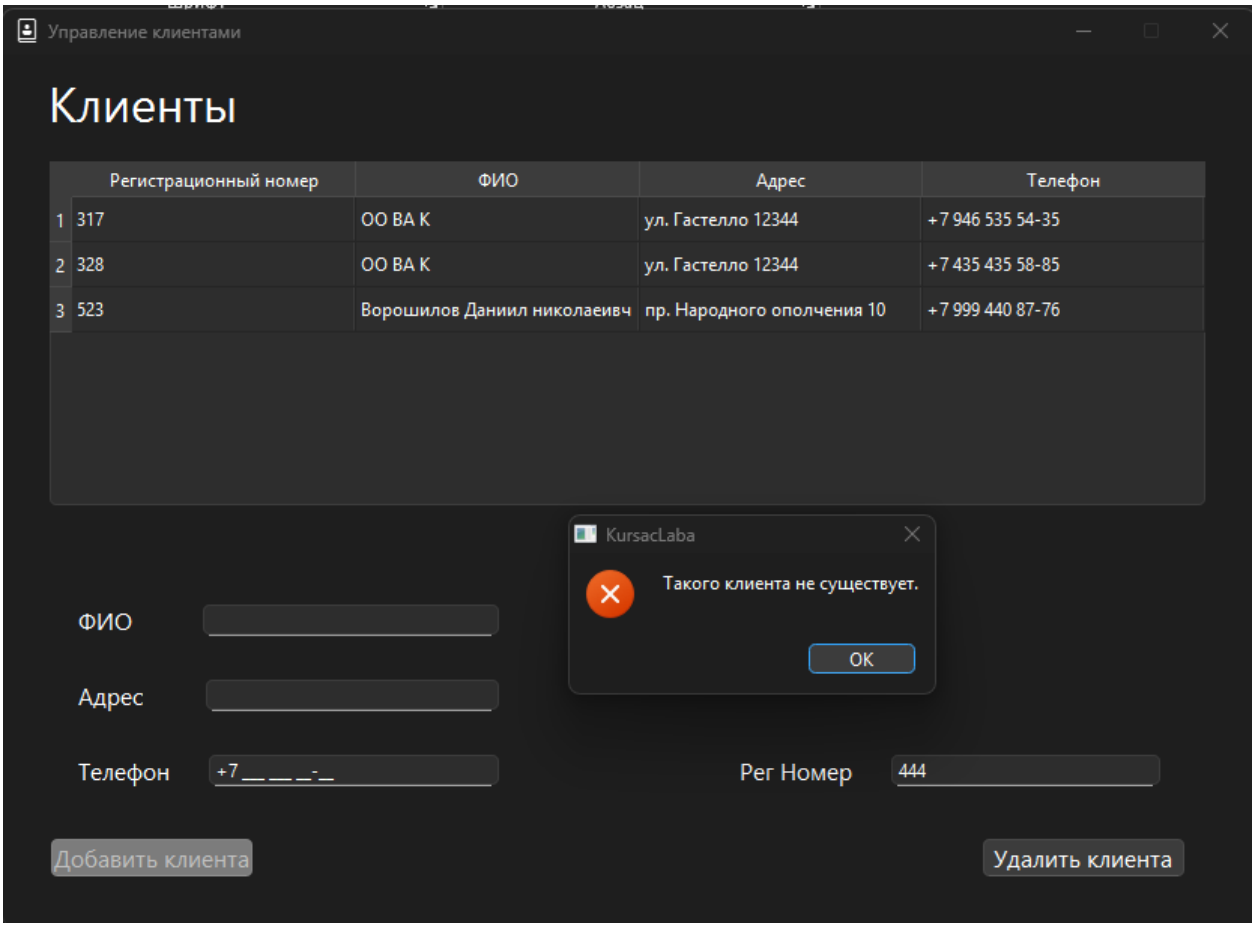


Удаление существующего клиента до/после:

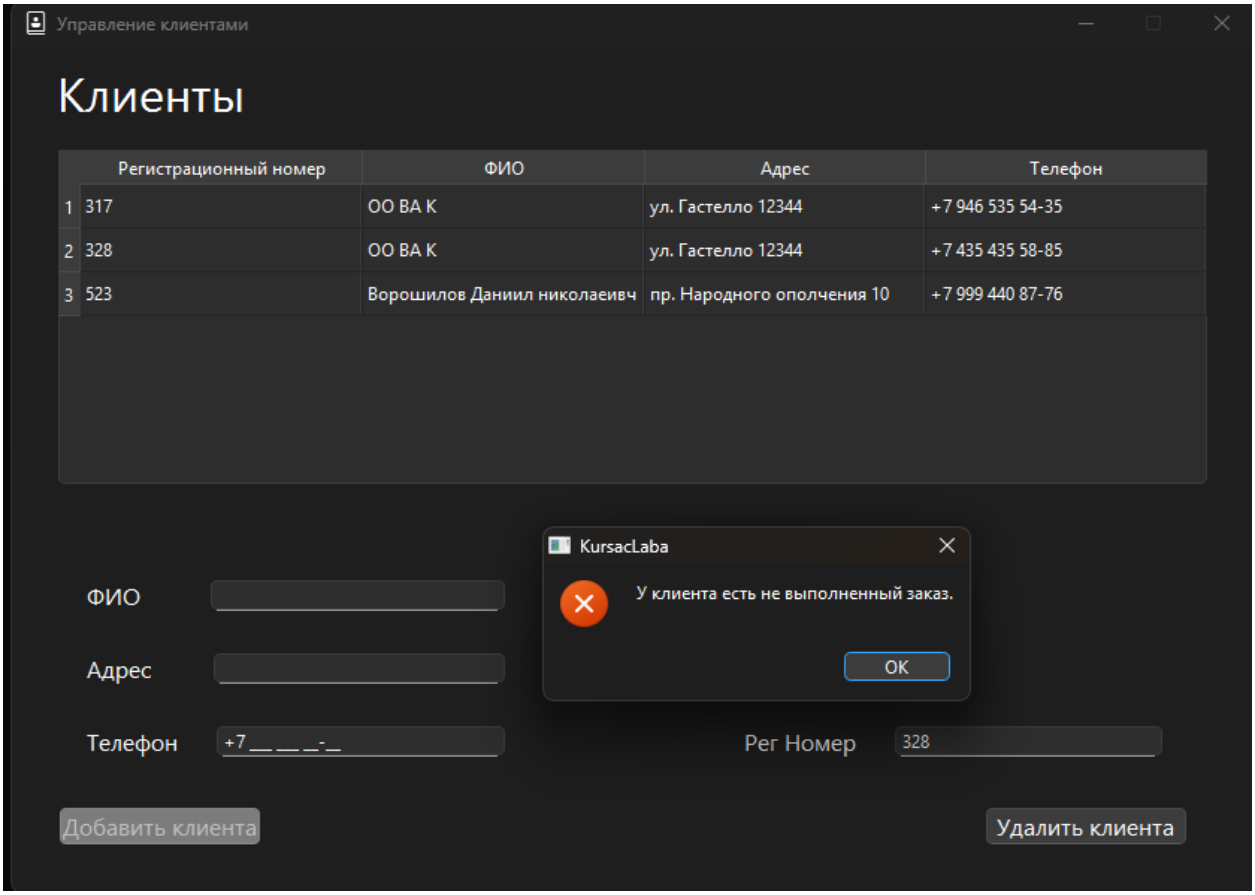




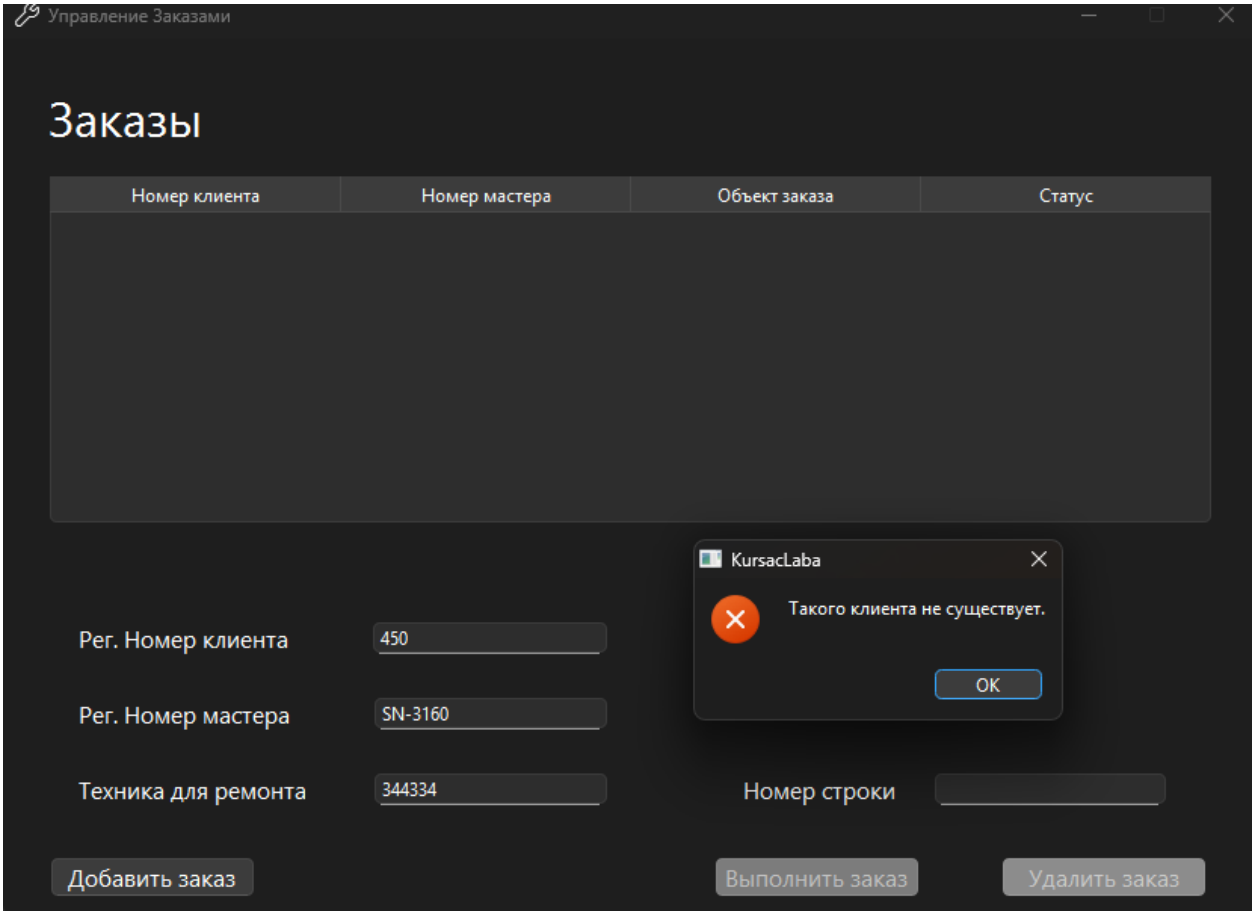
Удаление несуществующего клиента:



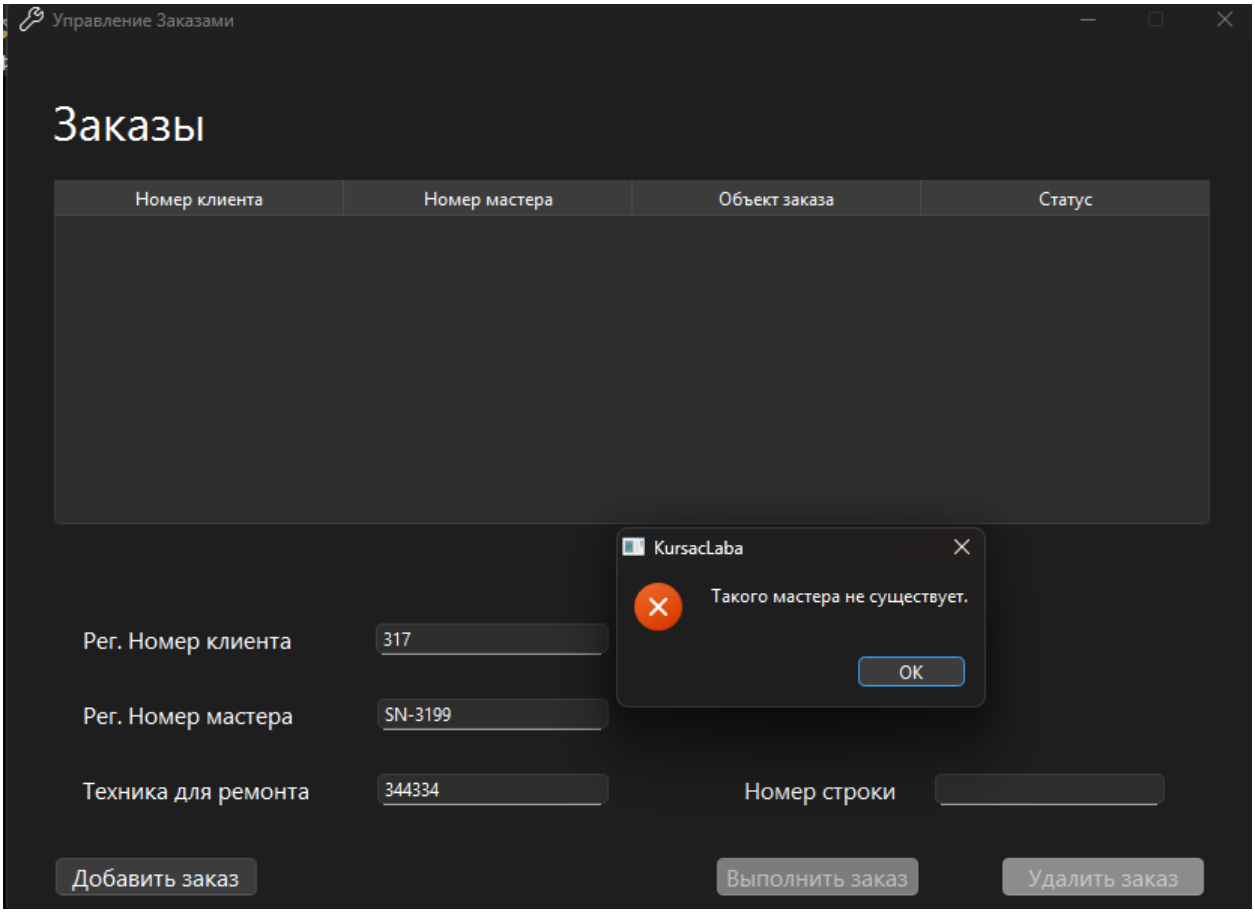
Удаление клиента с невыполненным заказом:



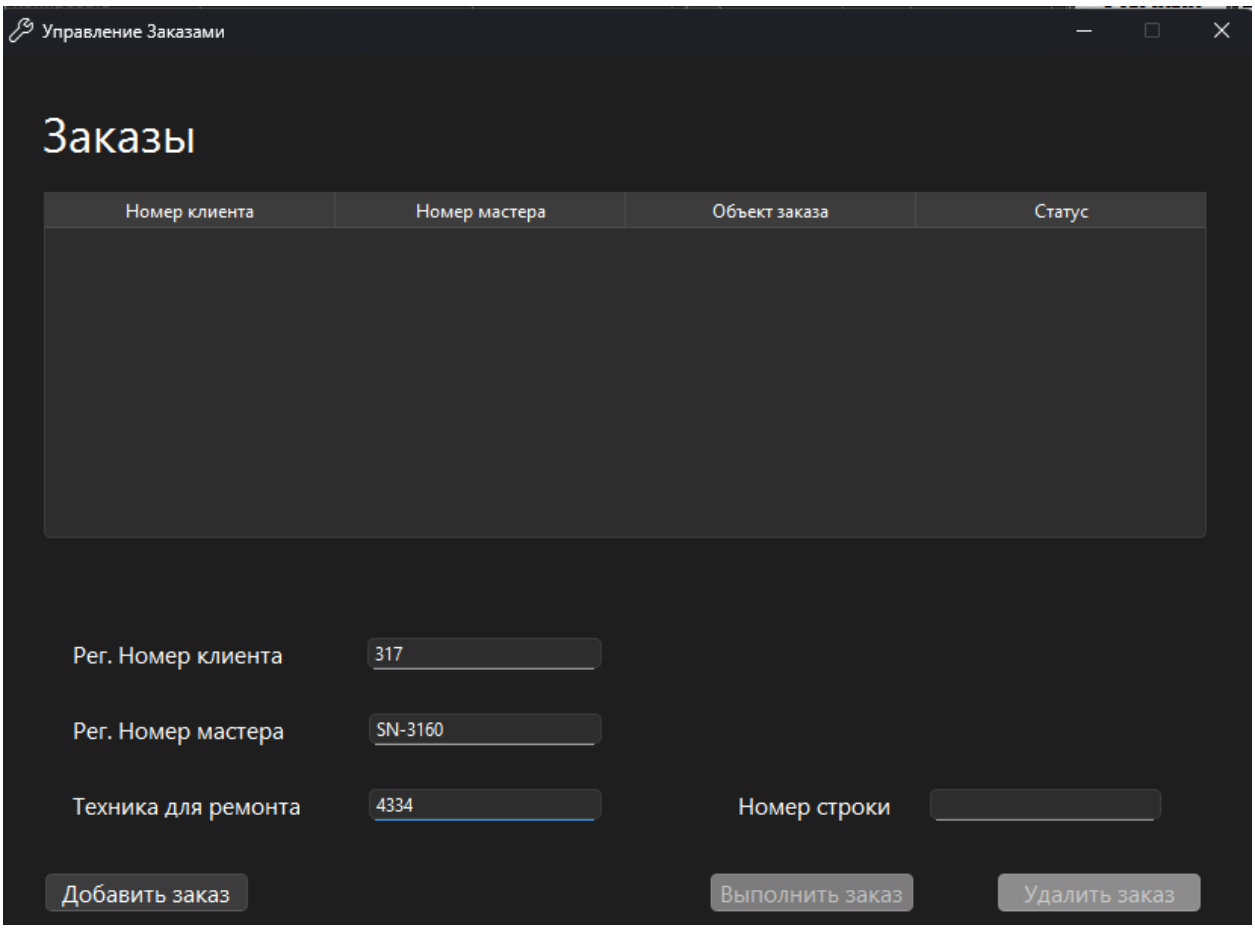
Добавление заказа с несуществующим клиентом:

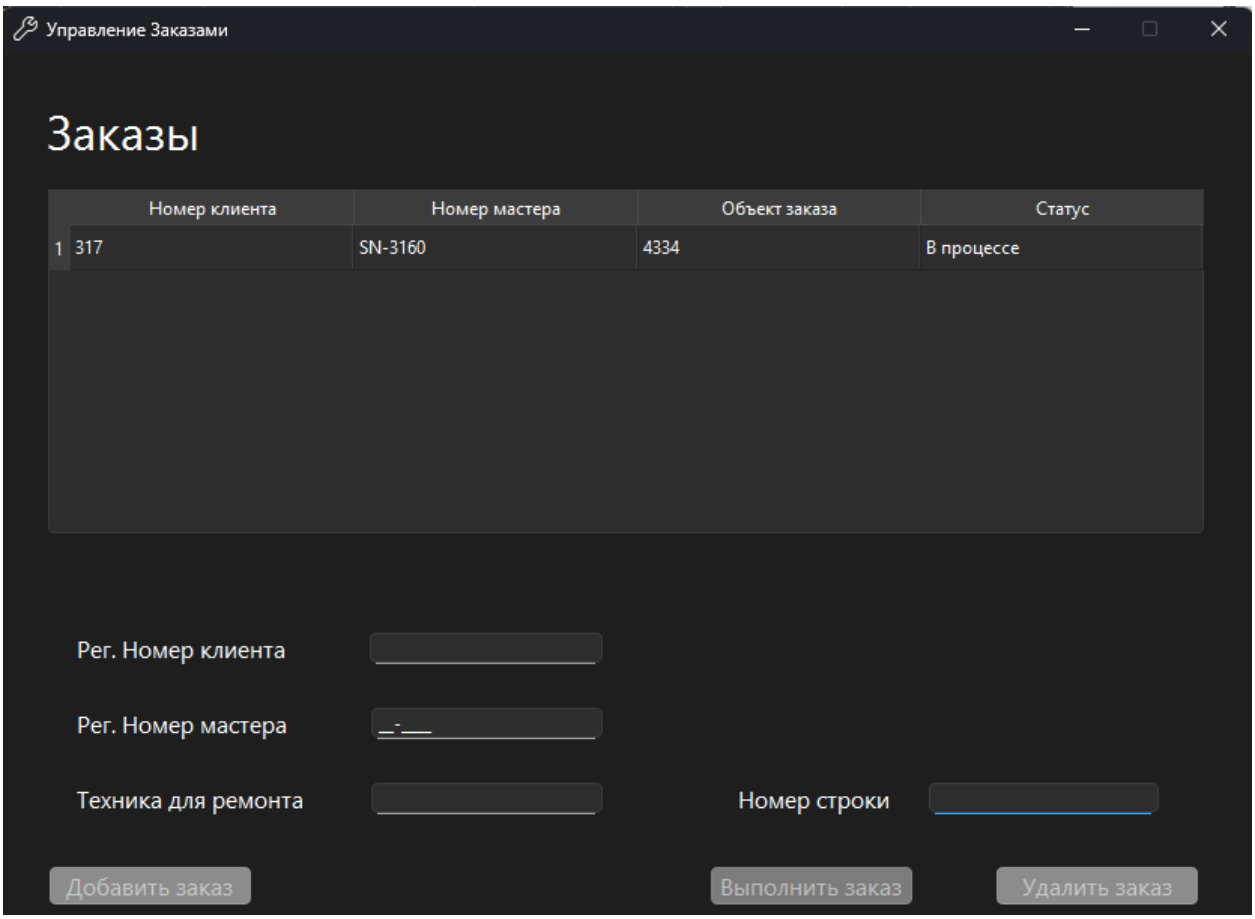


Добавление заказа с несуществующим мастером:

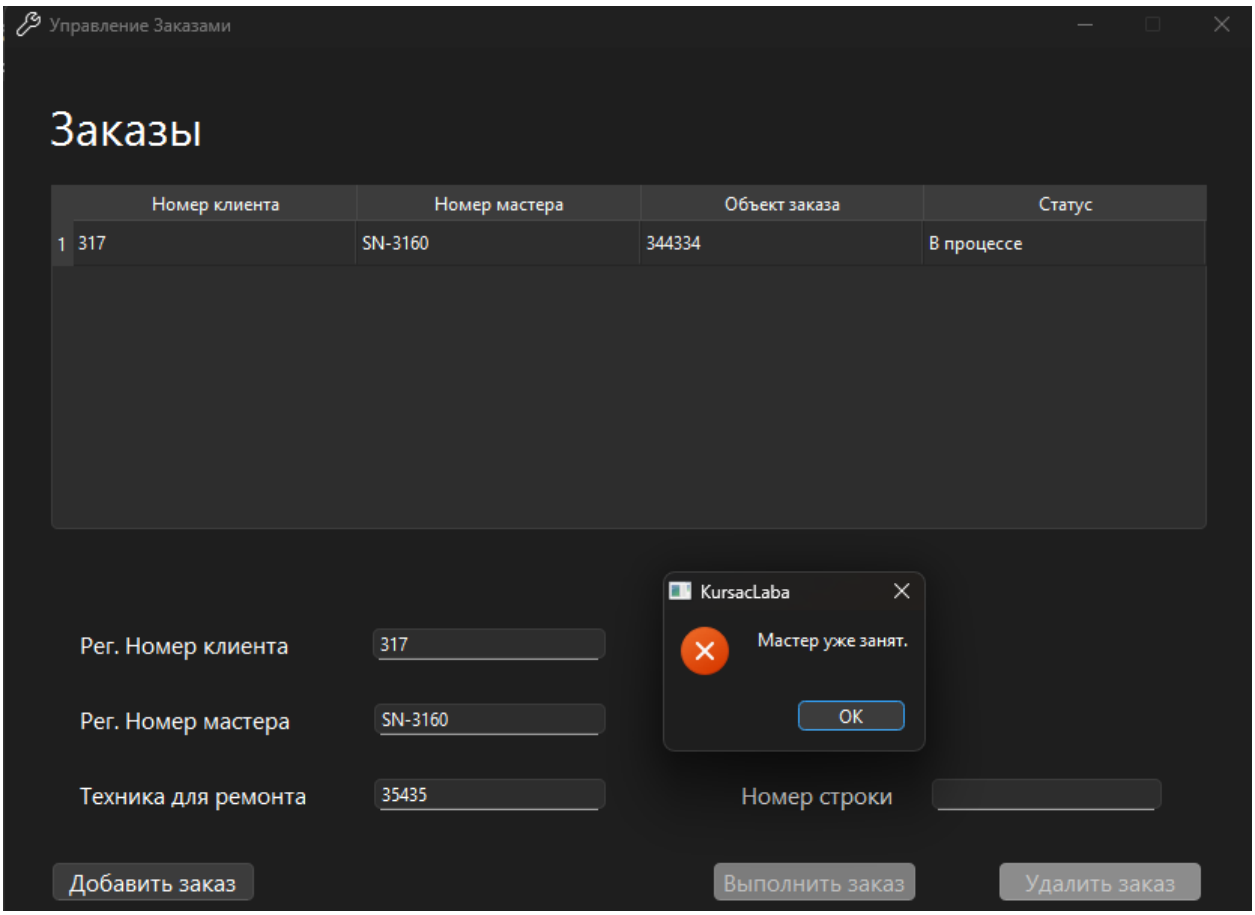


Добавление заказа до/после:

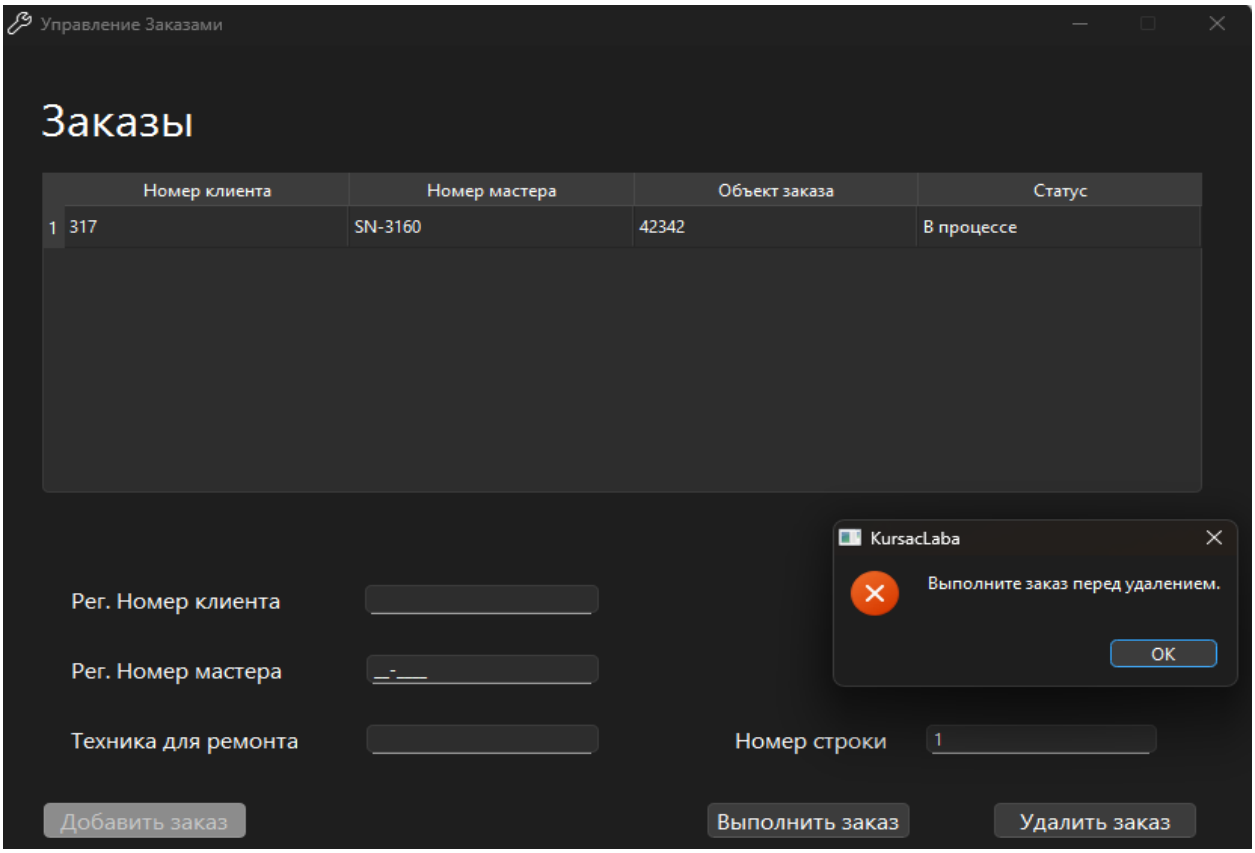




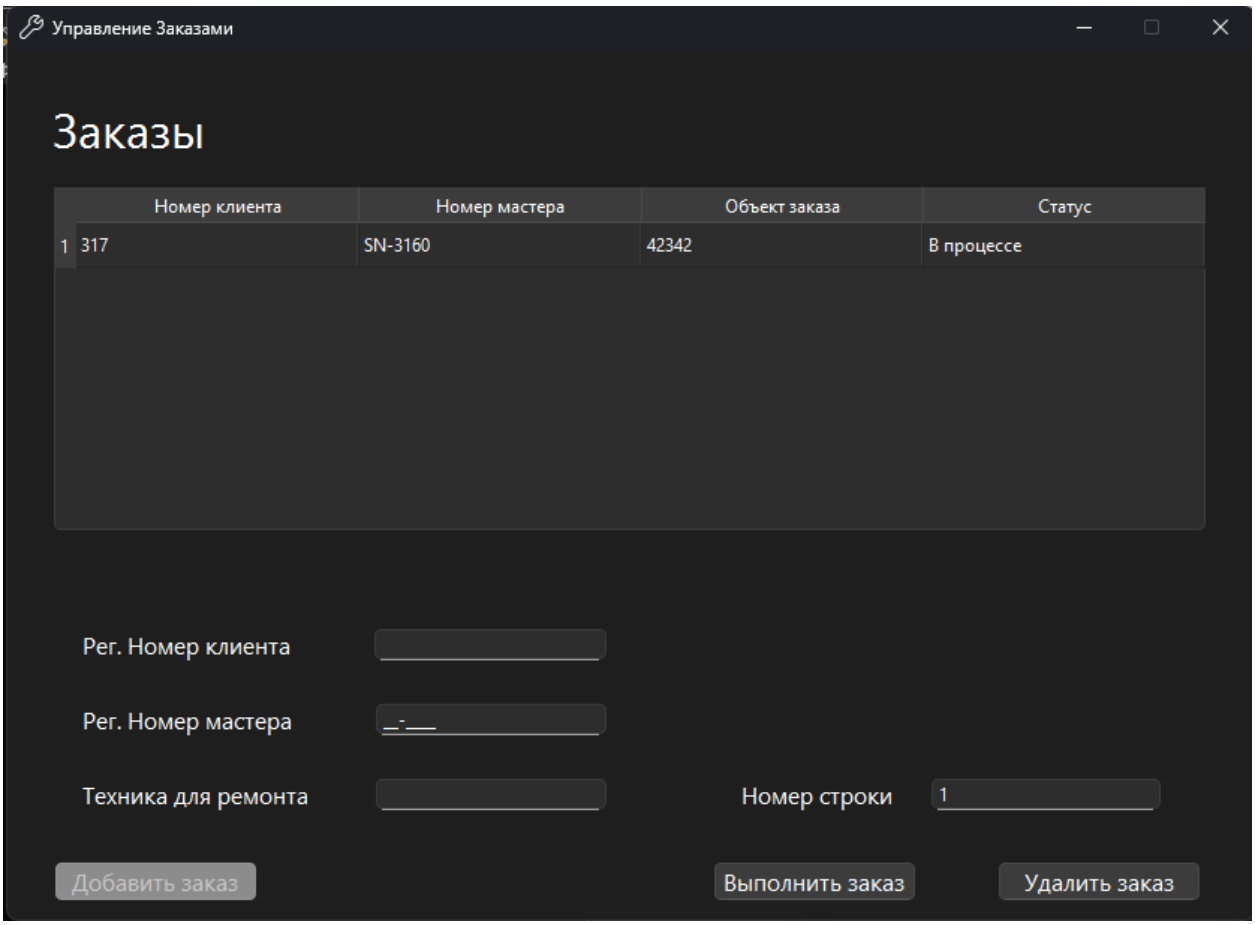
Добавление заказа, если у мастера уже есть заказ:

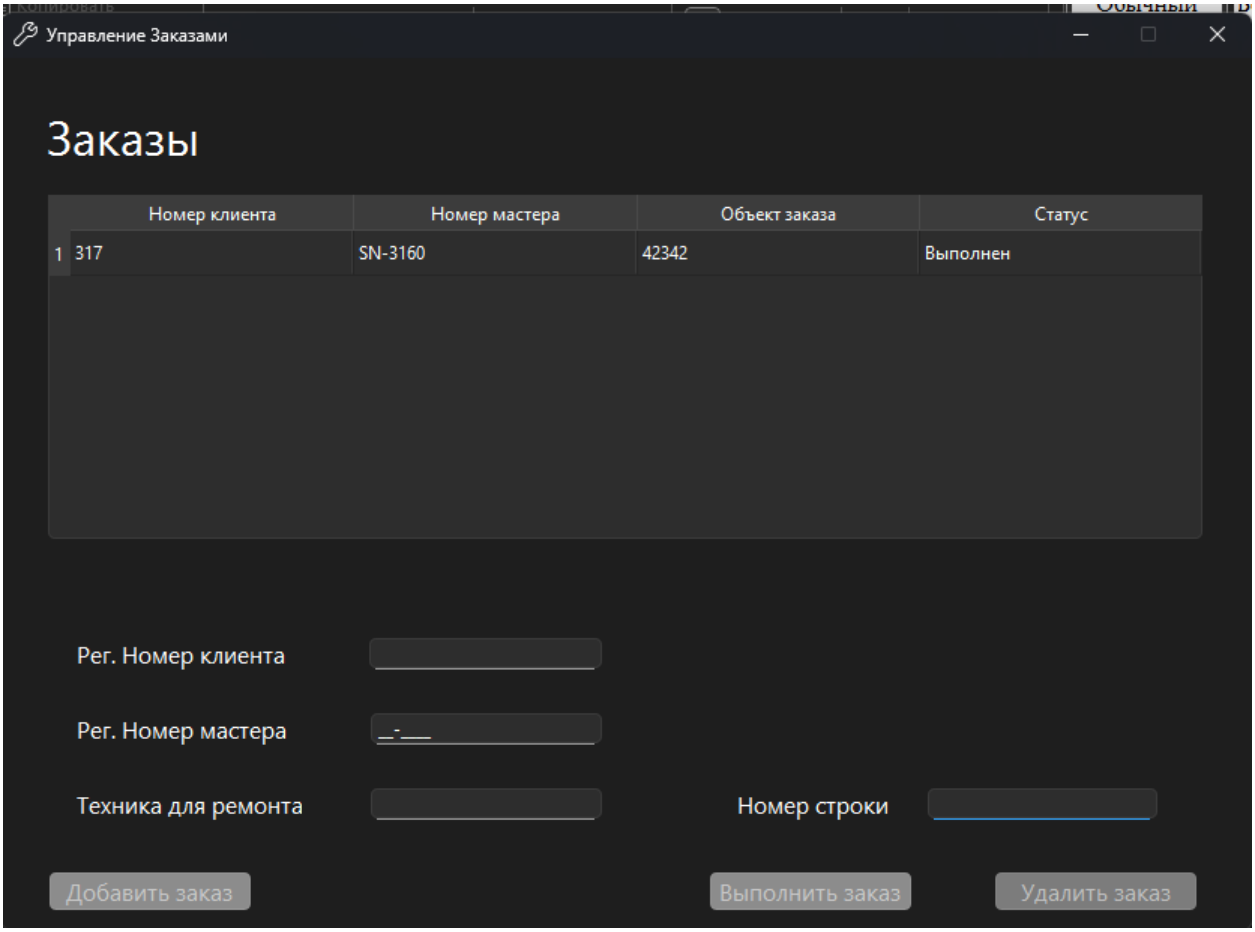


Удаление невыполненного заказа:

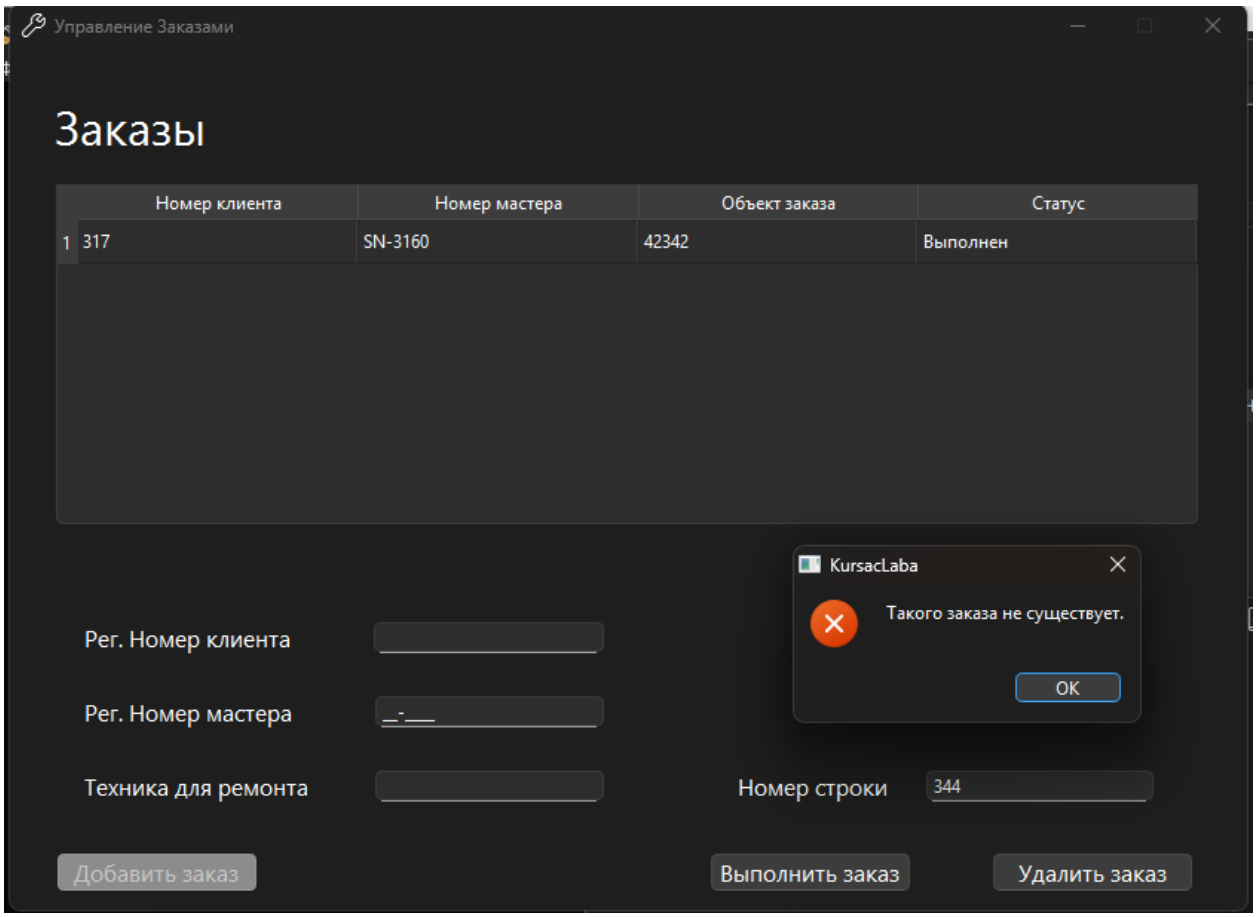


Выполнение заказа до/после:

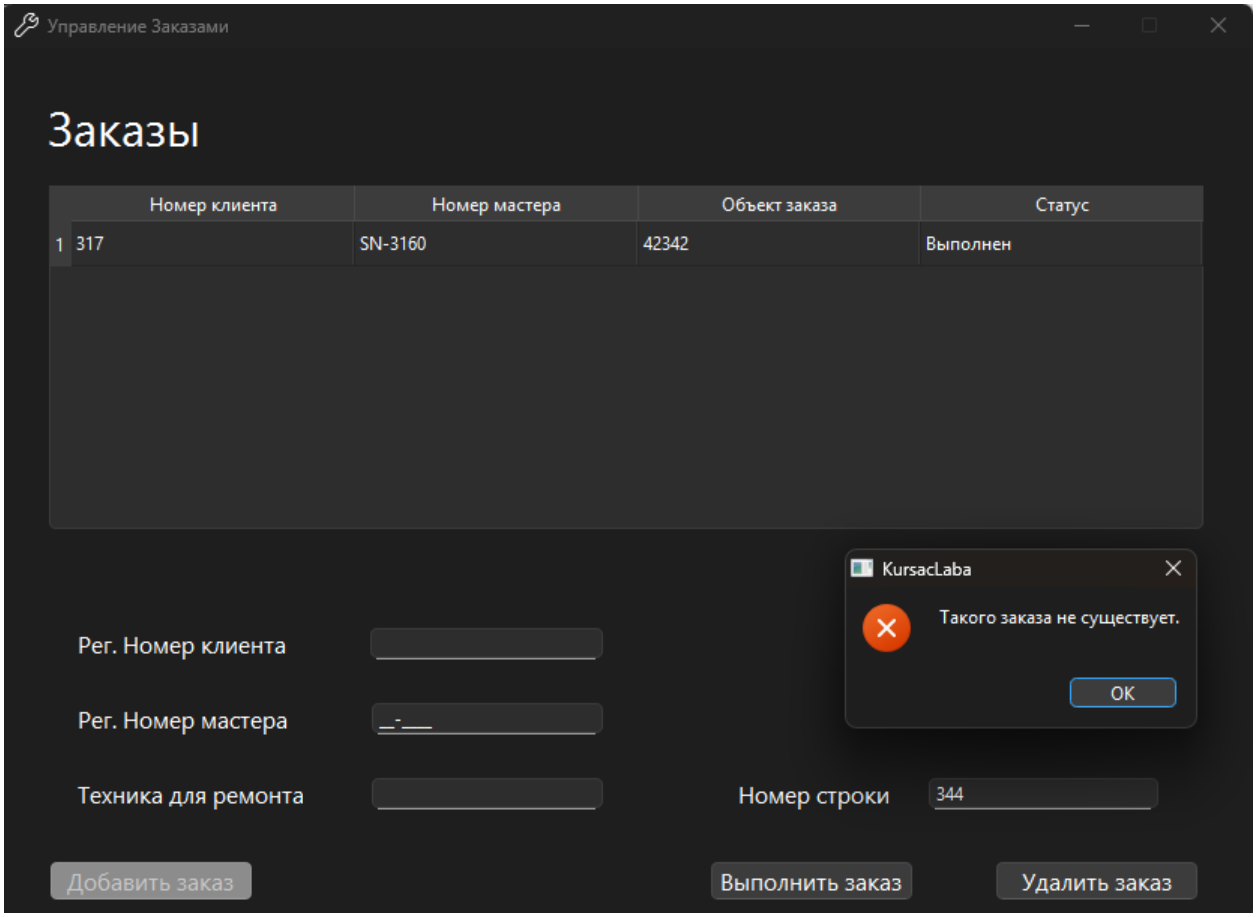




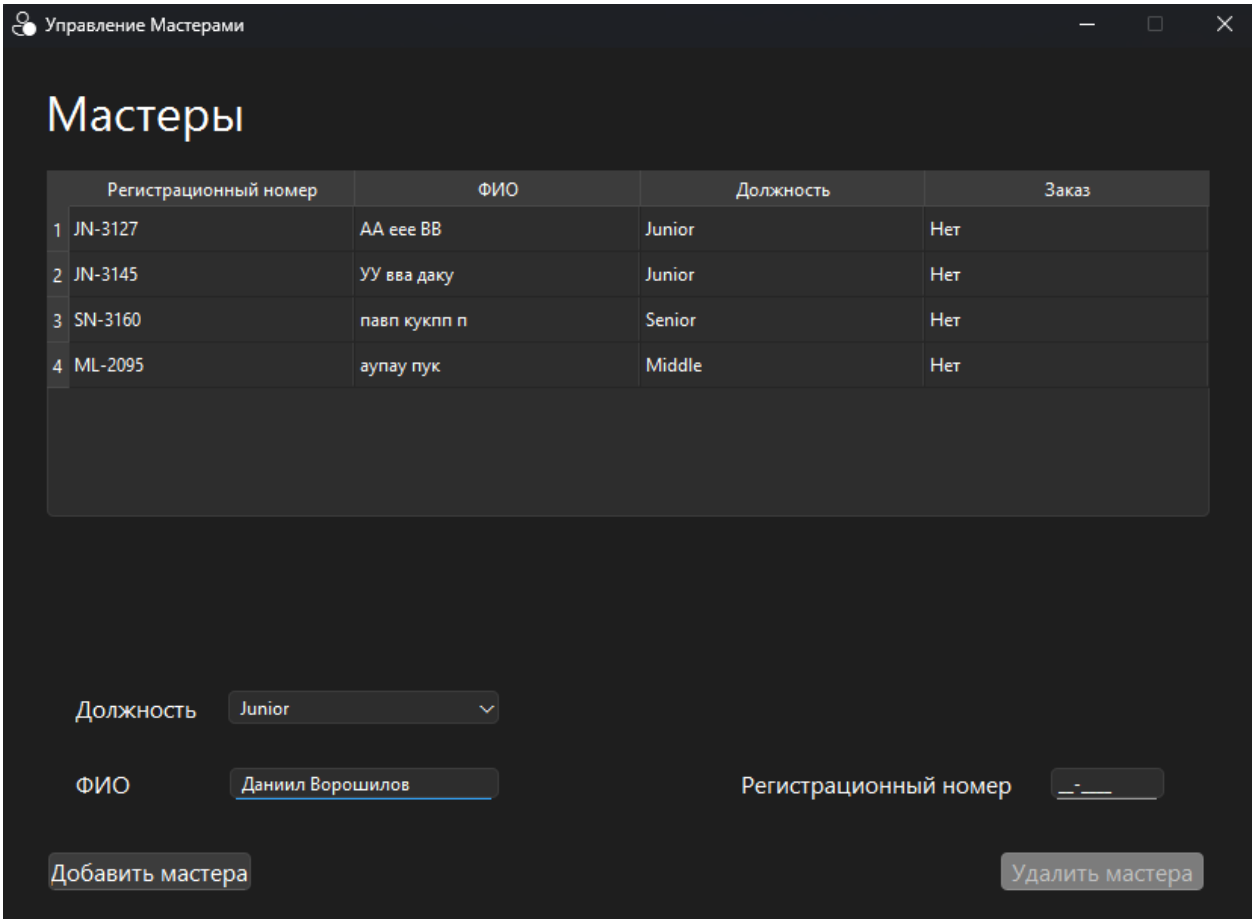
Выполнение несуществующего заказа:

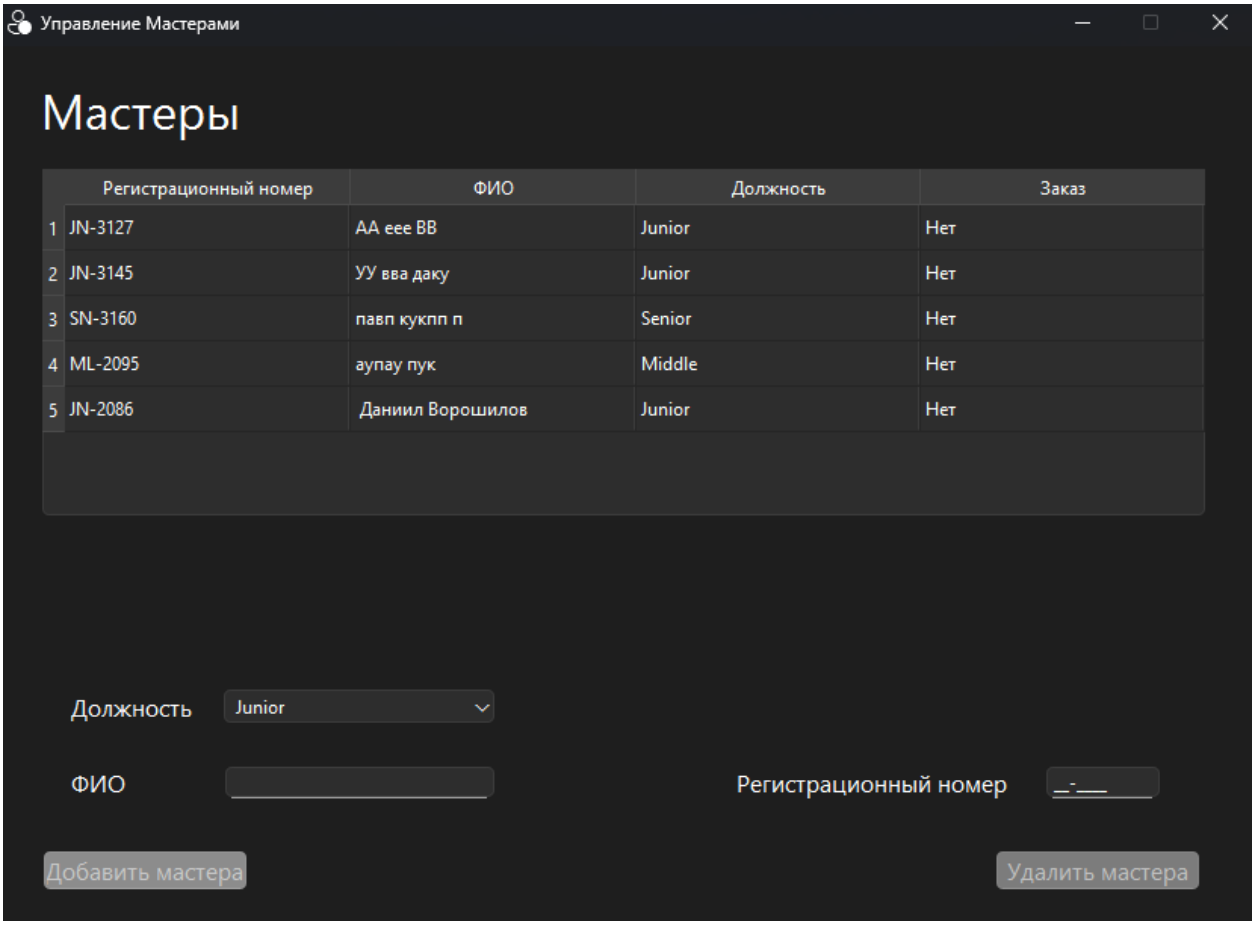


Удаление несуществующего заказа:

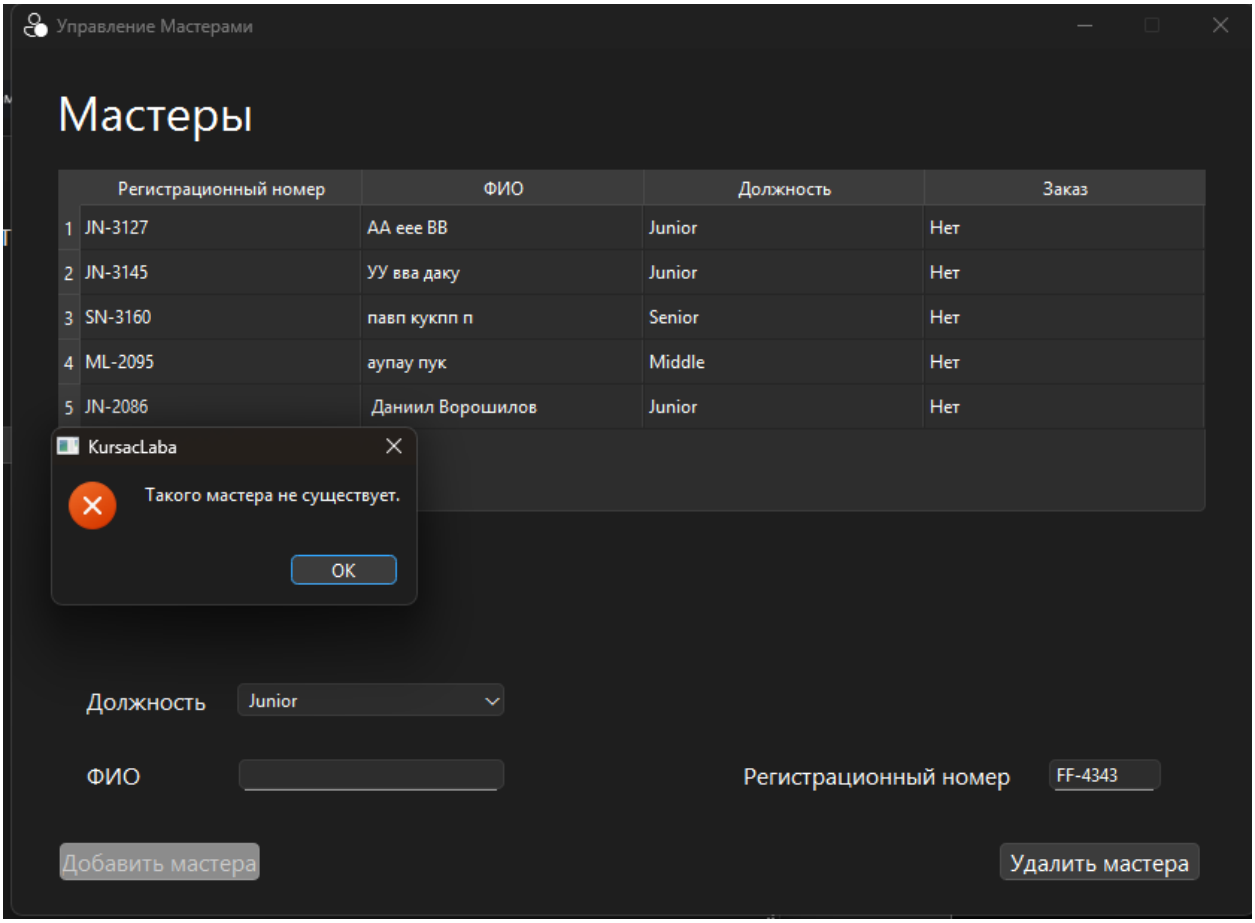


Добавление мастера до/после:

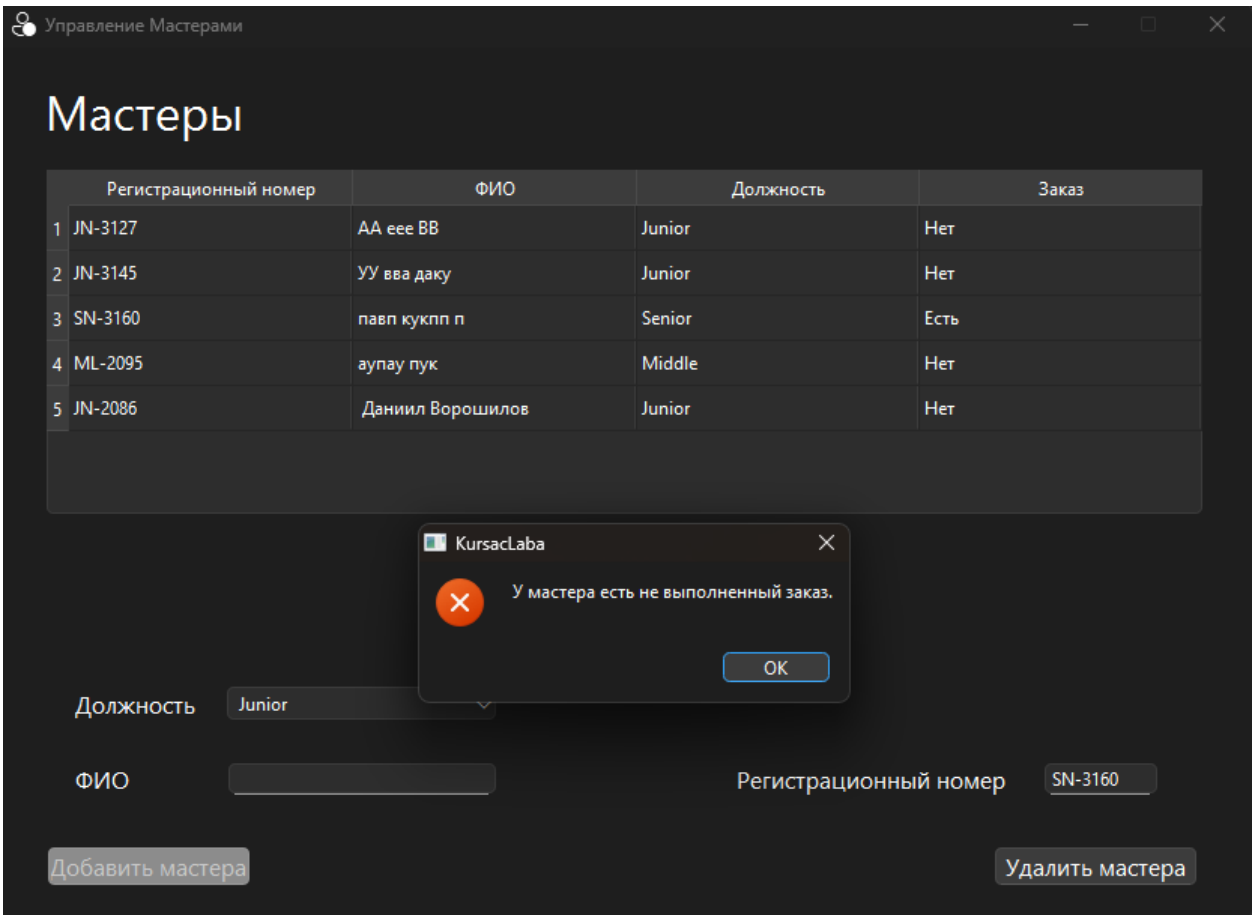




Удаление несуществующего мастера:



Удаление мастера с заказом:



8. Выводы

   В процессе лабораторной работы были получены практические навыки использования паттернов проектирования при разработке приложения в выбранной предметной области “Ремонтная мастерская”