Задание 1. Создание информационной программной системы с графическим интерфейсом на C++

Задание: Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

База данных (файл) магазина хозяйственных товаров содержит сведения о поступлении товаров: наименование, дата поступления, количество штук, страна-производитель. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Определить, есть ли в магазине указанный товар производства данной страны.
2. Показать количество товара каждого наименования в магазине.
3. Определить товары (наименование, страна), поступившие в указанный период.
4. Построить график поступления заданного товара по датам.

Выполнение:

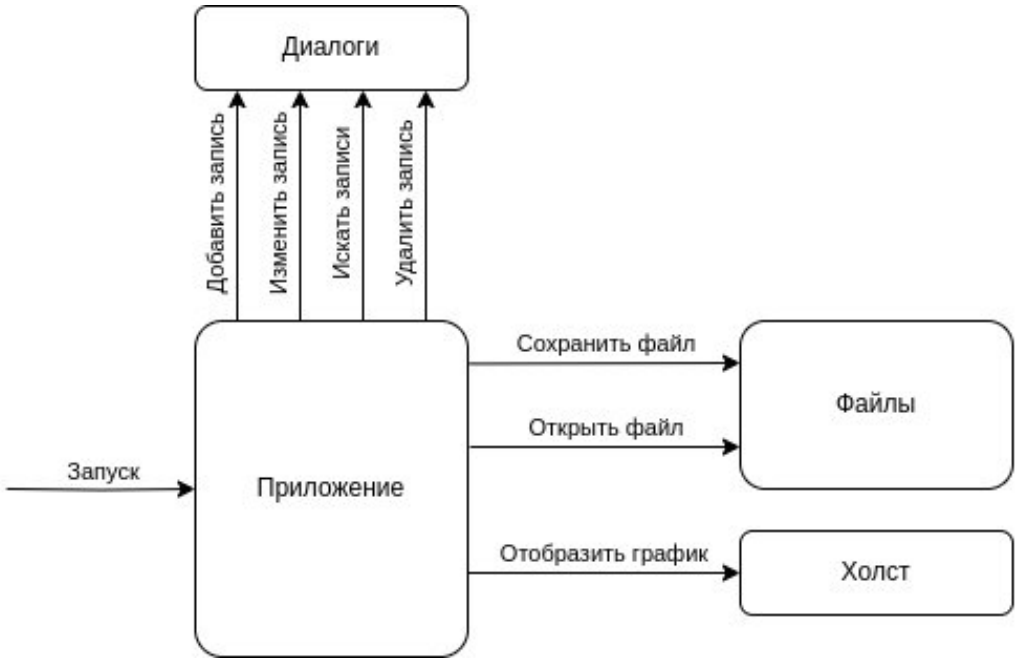

Рисунок 1 — Объектная декомпозиция

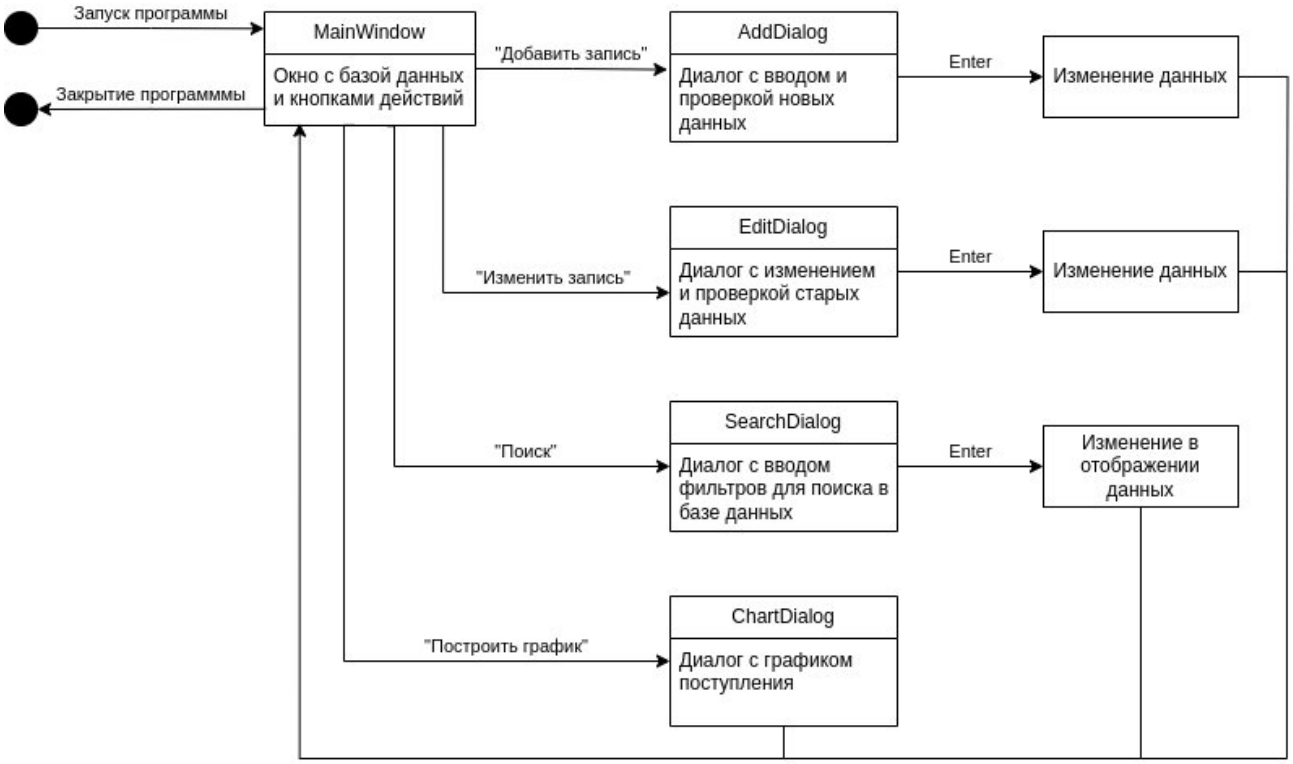

Рисунок 2 — Диаграмма классов интерфейса

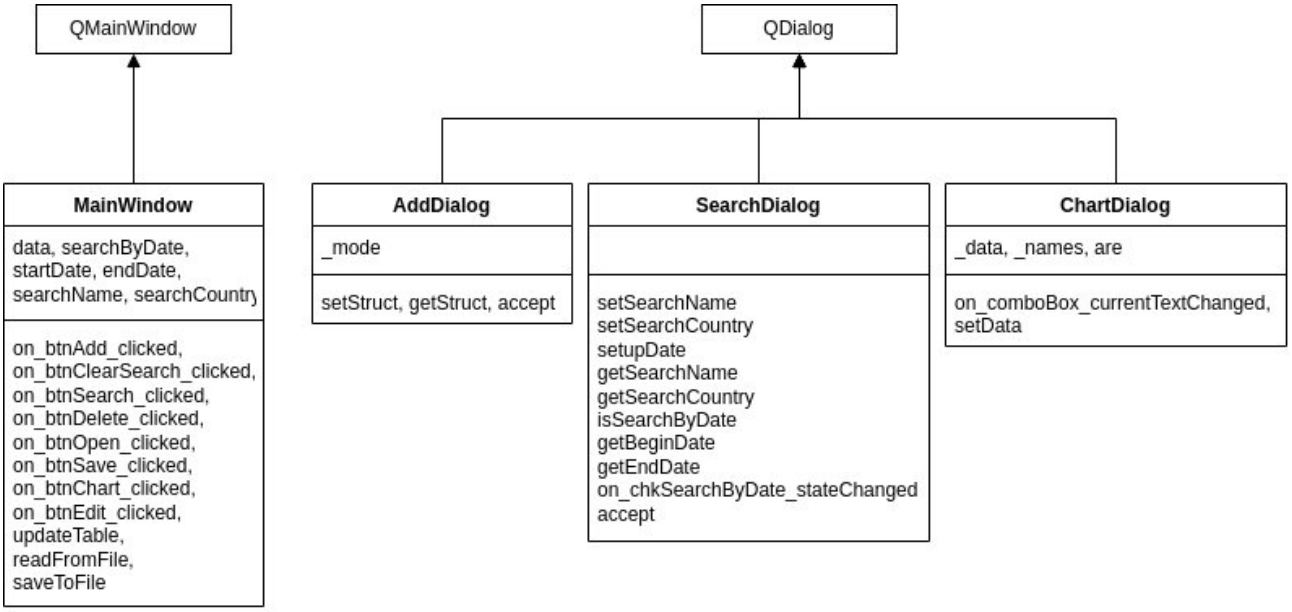

Рисунок 3 — Диаграма классов интерфейсной области

```
struct entrance {
  QString name;
  QDate date;
  int amount;
  QString country;
};
```

Рисунок 4 — Структура, отвечающая за данные

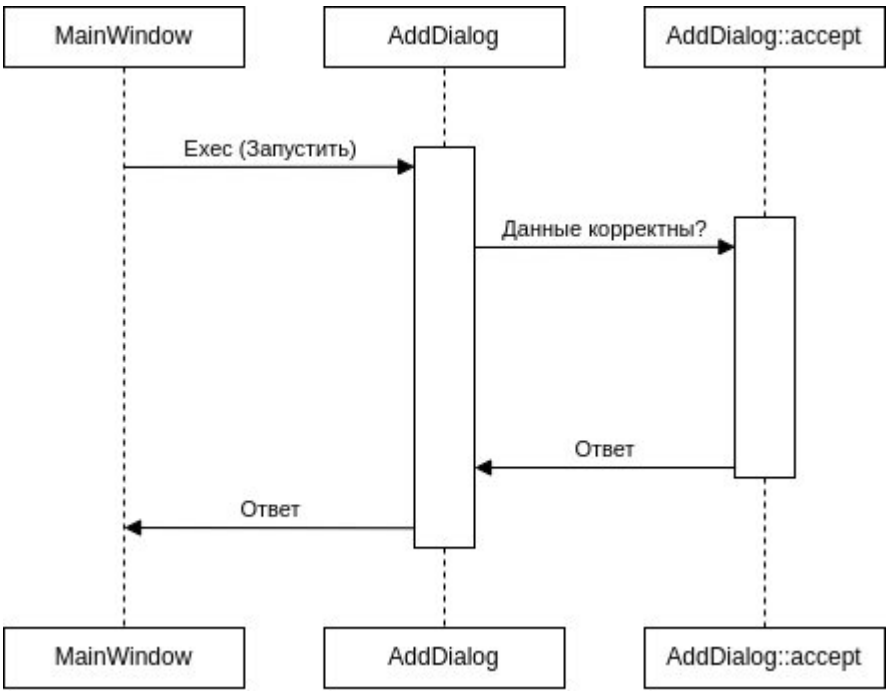

Рисунок 5 — Диаграмма последовательности действий

```
class MainWindow : public QMainWindow {
  Q_OBJECT;
  vector<entrance> data;

  // search parameters
  bool searchByDate = false;
  QDate startDate;
  QDate endDate;
  QString searchName;
  QString searchCountry;

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

private slots:
  void on_btnAdd_clicked();
  void on_btnClearSearch_clicked();
  void on_btnSearch_clicked();
  void on_btnDelete_clicked();
  void on_btnOpen_clicked();
  void on_btnSave_clicked();
  void on_btnChart_clicked();
  void on_btnEdit_clicked();

private:
  Ui::MainWindow *ui;

  void updateTable();
  void updateTable(QString additional_data);
  void readFromFile(const QString &filename);
  void saveToFile(const QString &filename);
};
```

Рисунок 6 — Заголовочный файл MainWindow

```cpp
#define MESSAGE_TIME 5000

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow) {
  ui->setupUi(this);
  ui->tableWidget->setColumnCount(4);
  ui->tableWidget->setHorizontalHeaderLabels({"Наименование",
                                              "Дата поступления", "Количество",
                                              "Страна-производитель"});
  ui->tableWidget->horizontalHeader()->setSectionResizeMode(
      QHeaderView::ResizeToContents);
  readFromFile("data.dat");
  updateTable();
}

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

void MainWindow::updateTable() { updateTable(""); }

void MainWindow::updateTable(QString additional_data) {
  ui->tableWidget->setRowCount(0);
  int i = 0;
  for (auto &e : data) {
    if (e.name.contains(searchName, Qt::CaseInsensitive) &&
        e.country.contains(searchCountry, Qt::CaseInsensitive) &&
        (!searchByDate || (startDate <= e.date && e.date <= endDate))) {
      ui->tableWidget->setRowCount(i + 1);
      ui->tableWidget->setItem(i, 0, new QTableWidgetItem(e.name));
      ui->tableWidget->setItem(i, 1, new QTableWidgetItem(e.date.toString()));
      ui->tableWidget->setItem(i, 2,
                               new QTableWidgetItem(QString::number(e.amount)));
      ui->tableWidget->setItem(i, 3, new QTableWidgetItem(e.country));
      i++;
    }
  }
  statusBar()->showMessage(additional_data +
                           "Отображено элементов: " + QString::number(i),
                           MESSAGE_TIME);
}
```

Рисунок 7 — Релизация отрисовки таблицы и конструктора

```cpp
void MainWindow::readFromFile(const QString &filename) {
  data.clear();
  QFile file(filename);
  if (file.open(QIODevice::ReadOnly)) {
    QDataStream in(&file);
    entrance e;
    while (!in.atEnd()) {
      in >> e.name >> e.date >> e.amount >> e.country;
      data.push_back(e);
    }
    statusBar()->showMessage("Успешно прочитано!", MESSAGE_TIME);
  } else {
    statusBar()->showMessage("Ошибка при чтении!", MESSAGE_TIME);
  }
}

void MainWindow::saveToFile(const QString &filename) {
  QFile file(filename);
  if (file.open(QIODevice::WriteOnly)) {
    QDataStream out(&file);
    for (auto &i : data) {
      out << i.name << i.date << i.amount << i.country;
    }
    file.close();
    statusBar()->showMessage("Успешно записано!", MESSAGE_TIME);
  } else {
    statusBar()->showMessage("Ошибка при записи!", MESSAGE_TIME);
  }
}
```

Рисунок 8 — Реализация чтения/записи файлов

```
void MainWindow::on_btnAdd_clicked() {
  AddDialog addDialog(this, ADD);
  if (addDialog.exec() == QDialog::Accepted) {
    data.push_back(addDialog.getStruct());
    updateTable("Добавлена одна запись. ");
  }
}

void MainWindow::on_btnClearSearch_clicked() {
  searchByDate = false;
  searchName.clear();
  searchCountry.clear();
  updateTable();
}

void MainWindow::on_btnSearch_clicked() {
  SearchDialog searchDialog(this);
  searchDialog.setSearchName(searchName);
  searchDialog.setSearchCountry(searchCountry);
  if (searchByDate)
    searchDialog.setupDate(startDate, endDate);

  if (searchDialog.exec() == QDialog::Accepted) {
    searchName = searchDialog.getSearchName();
    searchCountry = searchDialog.getSearchCountry();
    searchByDate = searchDialog.isSearchByDate();
    if (searchByDate) {
      startDate = searchDialog.getBeginDate();
      endDate = searchDialog.getEndDate();
    }
    updateTable();
  }
}

void MainWindow::on_btnDelete_clicked() {
  if (ui->tableWidget->selectedItems().isEmpty())
    statusBar()->showMessage("Не выбрана ни одна строка!", MESSAGE_TIME);
  else {
    auto row = ui->tableWidget->selectedItems().first()->row();
    data.erase(data.begin() + row);
    updateTable("Удалена 1 строка. ");
  }
}

void MainWindow::on_btnOpen_clicked() {
  readFromFile("data.dat");
  updateTable();
}

void MainWindow::on_btnSave_clicked() { saveToFile("data.dat"); }

void MainWindow::on_btnChart_clicked() {
  ChartDialog chartDialog(this);
  chartDialog.setData(data);
  chartDialog.exec();
}

void MainWindow::on_btnEdit_clicked() {
  if (ui->tableWidget->selectedItems().isEmpty())
    statusBar()->showMessage("Не выбрана ни одна строка!", MESSAGE_TIME);
  else {
    auto row = ui->tableWidget->selectedItems().first()->row();
    AddDialog editDialog(this, EDIT);
    editDialog.setStruct(data[row]);
    if (editDialog.exec() == QDialog::Accepted) {
      data[row] = editDialog.getStruct();
    }
    updateTable("Обновлена одна запись! ");
  }
}
```

Рисунок 9 — Реализация сигналов в MainWindow

```
class AddDialog : public QDialog {
  Q_OBJECT;
  int _mode;

public:
  explicit AddDialog(QWidget *parent = nullptr, int mode = ADD);
  ~AddDialog();

  void setStruct(const entrance &e);
  entrance getStruct();

private:
  Ui::AddDialog *ui;
  void accept() override;
};
```

Рисунок 10 — Заголовочный файл AddDialog

```
AddDialog::AddDialog(QWidget *parent, int mode)
    : QDialog(parent), ui(new Ui::AddDialog) {
  ui->setupUi(this);
  ui->dateEdit->setDate(QDate::currentDate());
  _mode = mode;
  if (_mode == EDIT) {
    setWindowTitle("Изменить запись");
  }
}

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

void AddDialog::setStruct(const entrance &e) {
  ui->edtName->setText(e.name);
  ui->dateEdit->setDate(e.date);
  ui->spnAmount->setValue(e.amount);
  ui->edtCountry->setText(e.country);
}

entrance AddDialog::getStruct() {
  return {ui->edtName->text(), ui->dateEdit->date(), ui->spnAmount->value(),
          ui->edtCountry->text()};
}

void AddDialog::accept() {
  if (ui->edtName->text().isEmpty()) {
    ui->edtName->setFocus();
    return;
  }
  if (ui->edtCountry->text().isEmpty()) {
    ui->edtCountry->setFocus();
    return;
  }

  QDialog::accept();
}
```

Рисунок 11 — Реализация AddDialog

```cpp
SearchDialog::SearchDialog(QWidget *parent)
    : QDialog(parent), ui(new Ui::SearchDialog) {
  ui->setupUi(this);
}

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

void SearchDialog::setSearchName(QString &s) { ui->edtSearchName->setText(s); }

void SearchDialog::setSearchCountry(QString &s) {
  ui->edtCountryName->setText(s);
}

void SearchDialog::setupDate(QDate &begin, QDate &end) {
  ui->chkSearchByDate->setCheckState(Qt::Checked);
  ui->datBegin->setEnabled(true);
  ui->datEnd->setEnabled(true);
  ui->datBegin->setDate(begin);
  ui->datEnd->setDate(end);
}

QString SearchDialog::getSearchName() { return ui->edtSearchName->text(); }
QString SearchDialog::getSearchCountry() { return ui->edtCountryName->text(); }
bool SearchDialog::isSearchByDate() {
  return ui->chkSearchByDate->checkState() == Qt::Checked;
}
QDate SearchDialog::getBeginDate() { return ui->datBegin->date(); }
QDate SearchDialog::getEndDate() { return ui->datEnd->date(); }

void SearchDialog::accept() {
  if (ui->datBegin->date() > ui->datEnd->date()) {
    QMessageBox msg;
    msg.setText("Начальная дата поиска не может быть больше конечной!");
    msg.setWindowTitle("Ошибка!");
    msg.resize(400, 300);
    msg.exec();
    return;
  }
  QDialog::accept();
}

void SearchDialog::on_chkSearchByDate_stateChanged(int arg1) {
  ui->datBegin->setEnabled(arg1 == Qt::Checked);
  ui->datEnd->setEnabled(arg1 == Qt::Checked);
}
```

Рисунок 12 — Реализация SearchDialog

```
class ChartDialog : public QDialog {
  Q_OBJECT;
  vector<entrance> _data;
  set<QString> _names;
  Area *area;

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

  void setData(vector<entrance> &data);

private slots:
  void on_comboBox_currentTextChanged(const QString &arg1);

private:
  Ui::ChartDialog *ui;
};
```

Рисунок 13 — Заголовочный файл chartdialog.h

```
ChartDialog::ChartDialog(QWidget *parent)
    : QDialog(parent), ui(new Ui::ChartDialog) {
  ui->setupUi(this);
  area = new Area(this);
  ui->verticalLayout->insertWidget(1, area);
}

ChartDialog::~ChartDialog() {
  delete ui;
  delete area;
}

void ChartDialog::setData(vector<entrance> &data) {
  _data = data;
  _names.clear();
  for (auto &it : _data) {
    _names.insert(it.name);
  }
  ui->comboBox->clear();
  for (auto &it : _names) {
    ui->comboBox->addItem(it);
  }
  area->setData(_data, ui->comboBox->currentText());
}

void ChartDialog::on_comboBox_currentTextChanged(const QString &arg1) {
  area->setData(_data, arg1);
  area->repaint();
}
```

Рисунок 14 — Реализация ChartDialog

```
void Area::paintEvent(QPaintEvent *) {
  QPainter painter(this);
  auto size = this->size();
  // painter.drawLine(0, 0, size.width(), size.height());
  // painter.drawText(100, 100, _name);

  int minAmount = INT_MAX;
  int maxAmount = INT_MIN;
  vector<entrance> toShow;
  for (const auto &it : _data) {
    if (it.name == _name) {
      toShow.push_back(it);
      minAmount = min(minAmount, it.amount);
      maxAmount = max(maxAmount, it.amount);
    }
  }

  QDate startDate = toShow.begin()->date;
  QDate endDate = toShow.begin()->date;
  for (const auto &it : toShow) {
    startDate = min(startDate, it.date);
    endDate = max(endDate, it.date);
  }

  sort(
      toShow.begin(), toShow.end(),
      [](const entrance &e1, const entrance &e2) { return e1.date < e2.date; });

  const int PADDING = 20;
  int chart_width = size.width() - 2 * PADDING;
  int chart_height = size.height() - 2 * PADDING;
  int xPerEn = chart_width / (toShow.size() + 1);

  // Оси системы координат
  painter.drawLine(PADDING * 2, size.height() - PADDING,
                   size.width() - PADDING / 3, size.height() - PADDING);
  painter.drawLine(PADDING * 2, size.height() - PADDING, PADDING * 2,
                   PADDING / 3);

  if (startDate == endDate) {
    painter.drawText(PADDING + chart_width / 2 - PADDING, size.height() - 7,
                     startDate.toString());
  } else {
    painter.drawText(PADDING * 2, size.height() - 7, startDate.toString());
    painter.drawText(PADDING + chart_width - 70, size.height() - 7,
                     endDate.toString());
  }

  int i = 0;
  int last_x, last_y;
  painter.setBrush(Qt::red);
  for (auto &it : toShow) {
    int x = PADDING + xPerEn * (++i);
    int y = PADDING + chart_height -
            chart_height * (static_cast<float>(it.amount) / maxAmount);
    painter.setPen(Qt::white);
    painter.drawText(10, y, QString::number(it.amount));

    if (i > 1) {
      painter.drawLine(last_x + 3, last_y + 3, x + 3, y + 3);
    }

    painter.setPen(Qt::red);
    if (i > 1)
      painter.drawEllipse(last_x, last_y, 6, 6);
    painter.drawEllipse(x, y, 6, 6);

    last_x = x;
    last_y = y;
  }
}
```

Рисунок 15 — Функция отрисовки графика

Тестирование программы:

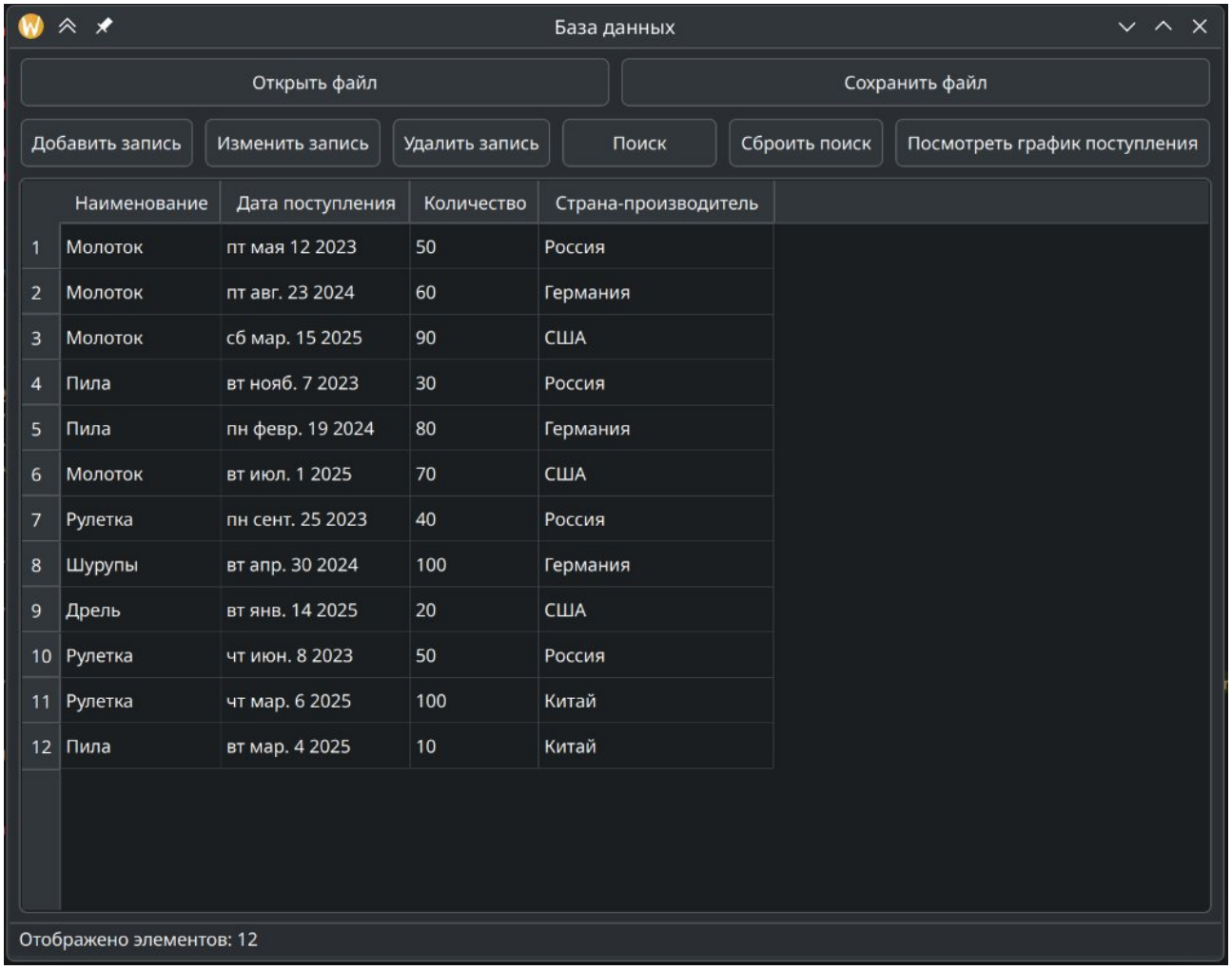

Рисунок 16 — Основное окно программы

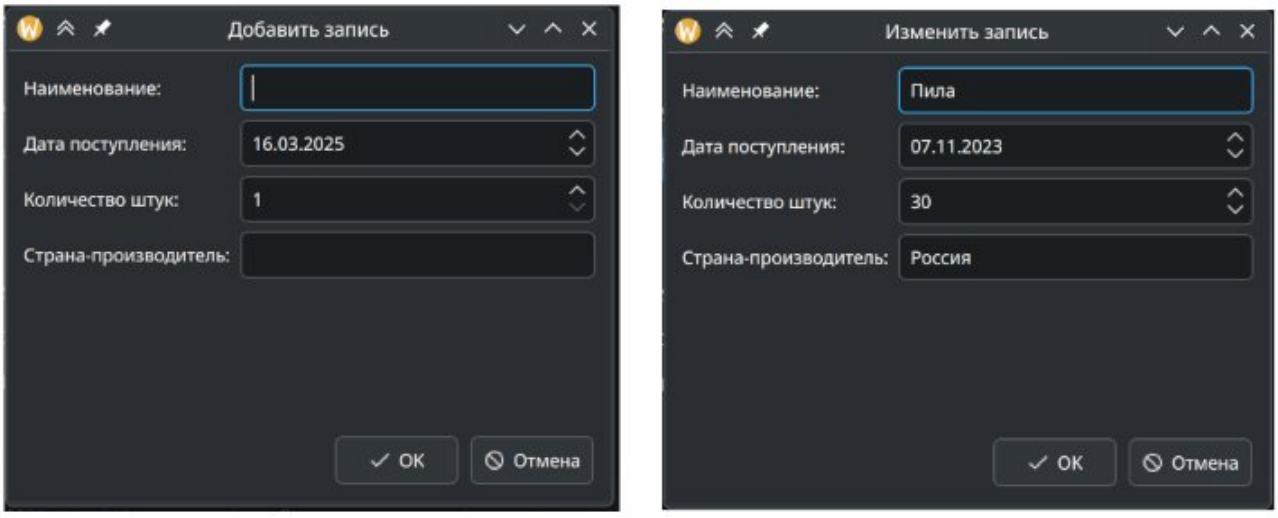

Рисунок 17 — Диалоги создания и редактирования записи

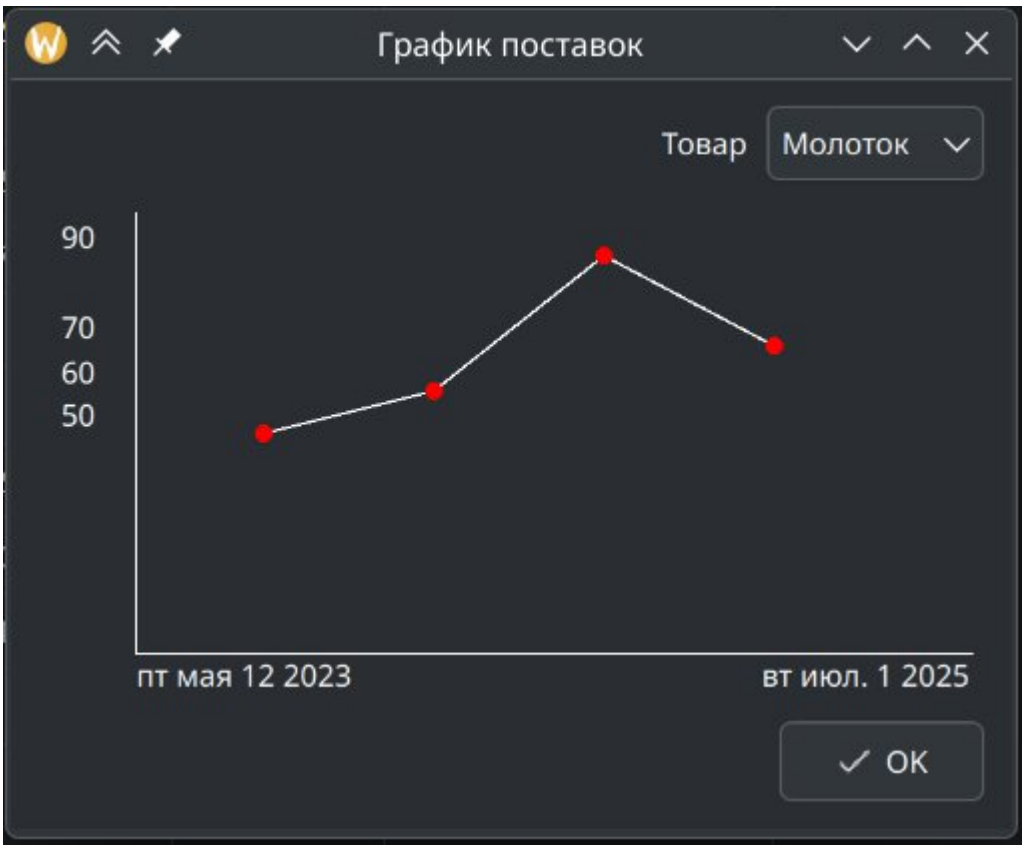

Рисунок 18 — График поставок

Задание 3 Изучение средств создания графических интерфейсов на C#

Задание: Разработать приложение «Крестики-нолики».

Интерфейс приложения будет состоять:

игрового поля: 9 клеток, в которых пользователями будут выставляться крестики – символ «X» и нолики – символ «O»;

поля вывода результатов ходов, в которое будет выводиться номер клетки, в которую установил свой символ пользователь, и результаты игры;

кнопки «Начать заново», перезапускающей игру.

Для упрощения программа будет разработана для двух игроков-людей. Она будет проверять, победил ли один из игроков после каждого хода, а в конце игры, когда все клетки будут заполнены, будет проверять, выиграл ли один из игроков или игра закончилась ничьей.

После завершения игры либо победой, либо ничьей, программа не будет реагировать на любые действия пользователей до нажатия ими кнопки «Начать заново». По нажатию той же кнопки игра может быть завершена досрочно.

Клетки игрового поля будут реализованы с помощью кнопок, по нажатию которых в названиях кнопок будут появляться символы «X» или «O».

Выполнение:

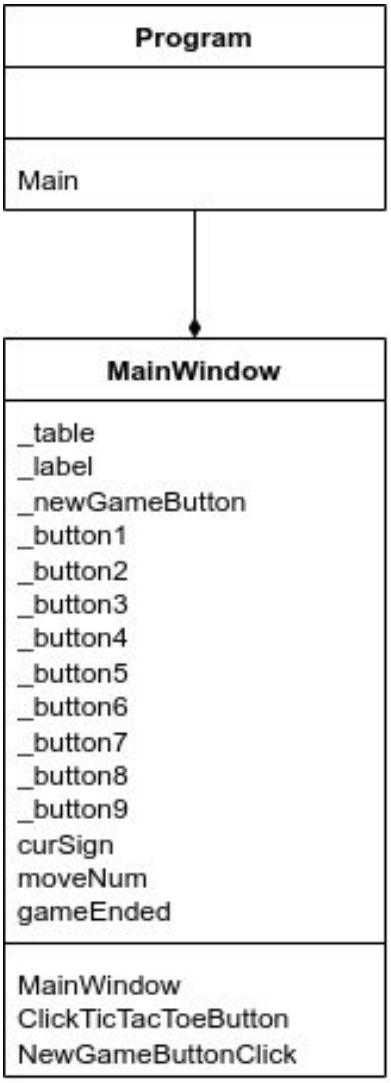

Рисунок 19 — Диаграмма классов

```
using System;
using Gtk;

namespace _03;

class Program
{
    [STAThread]
    public static void Main(string[] args)
    {
        Application.Init();

        var app = new Application("org._03._03", GLib.ApplicationFlags.None);
        app.Register(GLib.Cancellable.Current);

        var win = new MainWindow();
        app.AddWindow(win);

        win.Show();
        Application.Run();
    }
}
```

Рисунок 20 — Класс, отвечающий за запуск приложения

```
class MainWindow : Window
{
    [UI] private readonly Label _label = null;

    [UI] private readonly Button _button1 = null;
    [UI] private readonly Button _button2 = null;
    [UI] private readonly Button _button3 = null;
    [UI] private readonly Button _button4 = null;
    [UI] private readonly Button _button5 = null;
    [UI] private readonly Button _button6 = null;
    [UI] private readonly Button _button7 = null;
    [UI] private readonly Button _button8 = null;
    [UI] private readonly Button _button9 = null;

    private string _curSign;
    private int _moveNum;
    private bool _gameEnded;

    1 usage
    public MainWindow() : this(new Builder( resource_name: "MainWindow.glade"))
    {
        _curSign = "X";
        _moveNum = 1;
        _gameEnded = false;
    }

    1 usage
    private MainWindow(Builder builder) : base(builder.GetRawOwnedObject( name: "MainWindow"))
    {
        builder.Autoconnect( handler: this);

        DeleteEvent += Window_DeleteEvent;
    }

    1 usage
    private void Window_DeleteEvent(object sender, DeleteEventArgs a)
    {
        Application.Quit();
    }
```

Рисунок 21 — Класс MainWindow и часть с GTK#

```
private void NewGameButtonClick(object sender, EventArgs a)
{
    _curSign = "X";
    _moveNum = 1;
    _gameEnded = false;
    _button1.Label = "";
    _button2.Label = "";
    _button3.Label = "";
    _button4.Label = "";
    _button5.Label = "";
    _button6.Label = "";
    _button7.Label = "";
    _button8.Label = "";
    _button9.Label = "";
    _label.Text = "Игра началась!";
}
```

Рисунок 22 — Обработчик нажатия на кнопку «Новая игра»

```
private void ClickTicTacToeButton(object sender, EventArgs a)
{
    Button button = (Button)sender;
    if (string.IsNullOrEmpty(button.Label) & !_gameEnded)
    {
        button.Label = _curSign;
        if (!string.IsNullOrEmpty(_button1.Label)
            & _button1.Label == _button2.Label
            & _button2.Label == _button3.Label
            |
            !string.IsNullOrEmpty(_button4.Label)
            & _button4.Label == _button5.Label
            & _button5.Label == _button6.Label
            |
            !string.IsNullOrEmpty(_button7.Label)
            & _button7.Label == _button8.Label
            & _button8.Label == _button9.Label
            |
            !string.IsNullOrEmpty(_button1.Label)
            & _button1.Label == _button4.Label
            & _button4.Label == _button7.Label
            |
            !string.IsNullOrEmpty(_button2.Label)
            & _button2.Label == _button5.Label
            & _button5.Label == _button8.Label
            |
            !string.IsNullOrEmpty(_button3.Label)
            & _button3.Label == _button6.Label
            & _button6.Label == _button9.Label
            |
            !string.IsNullOrEmpty(_button1.Label)
            & _button1.Label == _button5.Label
            & _button5.Label == _button9.Label
            |
            !string.IsNullOrEmpty(_button3.Label)
            & _button3.Label == _button5.Label
            & _button5.Label == _button7.Label
           )
        {
            _gameEnded = true;
            _label.Text = "Ход " + _moveNum + ": Победил " + _curSign + "!";
        }
        else if (!string.IsNullOrEmpty(_button1.Label) &
                 !string.IsNullOrEmpty(_button2.Label) &
                 !string.IsNullOrEmpty(_button3.Label) &
                 !string.IsNullOrEmpty(_button4.Label) &
                 !string.IsNullOrEmpty(_button5.Label) &
                 !string.IsNullOrEmpty(_button6.Label) &
                 !string.IsNullOrEmpty(_button7.Label) &
                 !string.IsNullOrEmpty(_button8.Label) &
                 !string.IsNullOrEmpty(_button9.Label)
                )
        {
            _gameEnded = true;
            _label.Text = "Ход " + _moveNum + ": Ничья!";
        }
        else
            _label.Text = "Ход " + _moveNum + ": " + _curSign + " сходил на " + button.Name + " клетку";

        if (_curSign == "X")
            _curSign = "O";
        else
            _curSign = "X";
        _moveNum += 1;
```

Рисунок 23 — Обработчик нажатия на кнопку игрового поля

Тестирование программы:

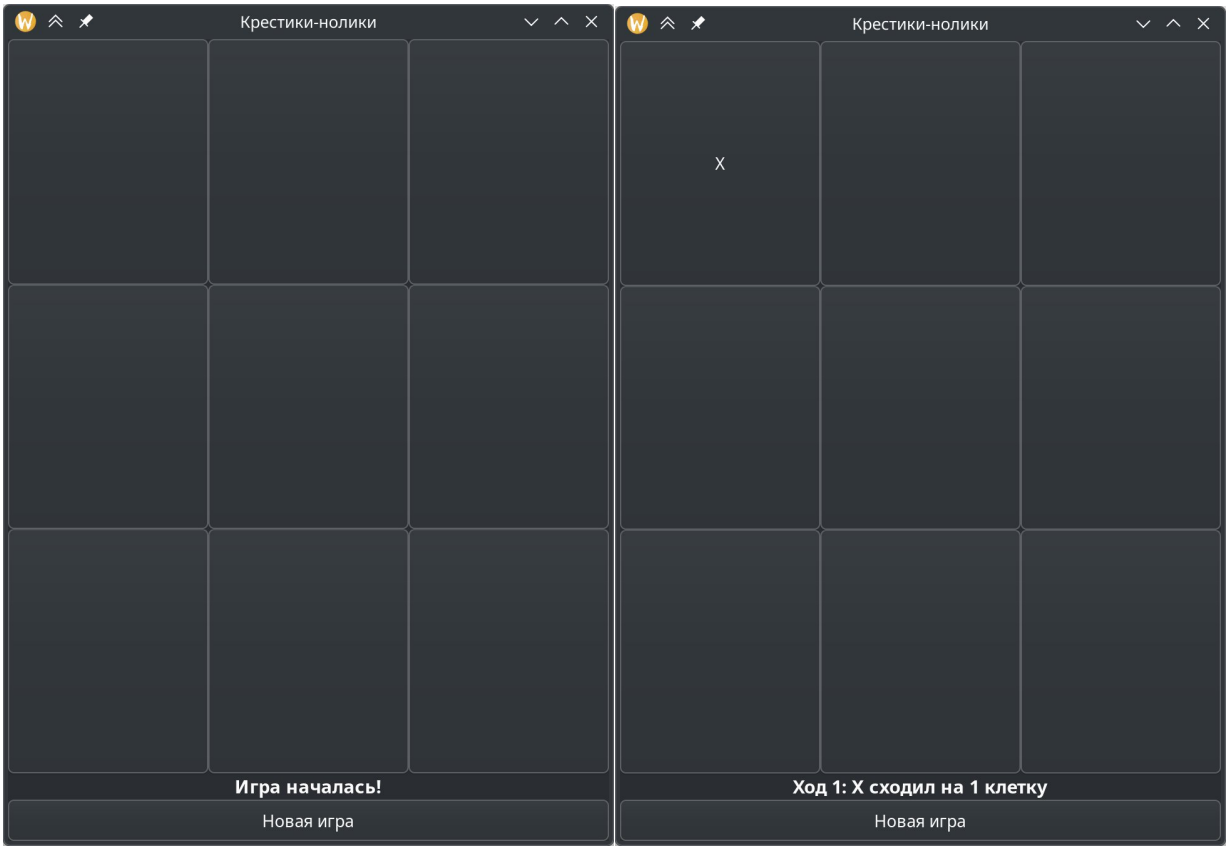

Рисунок 24 — Окно программы после запуска (слева) и после хода (справа)

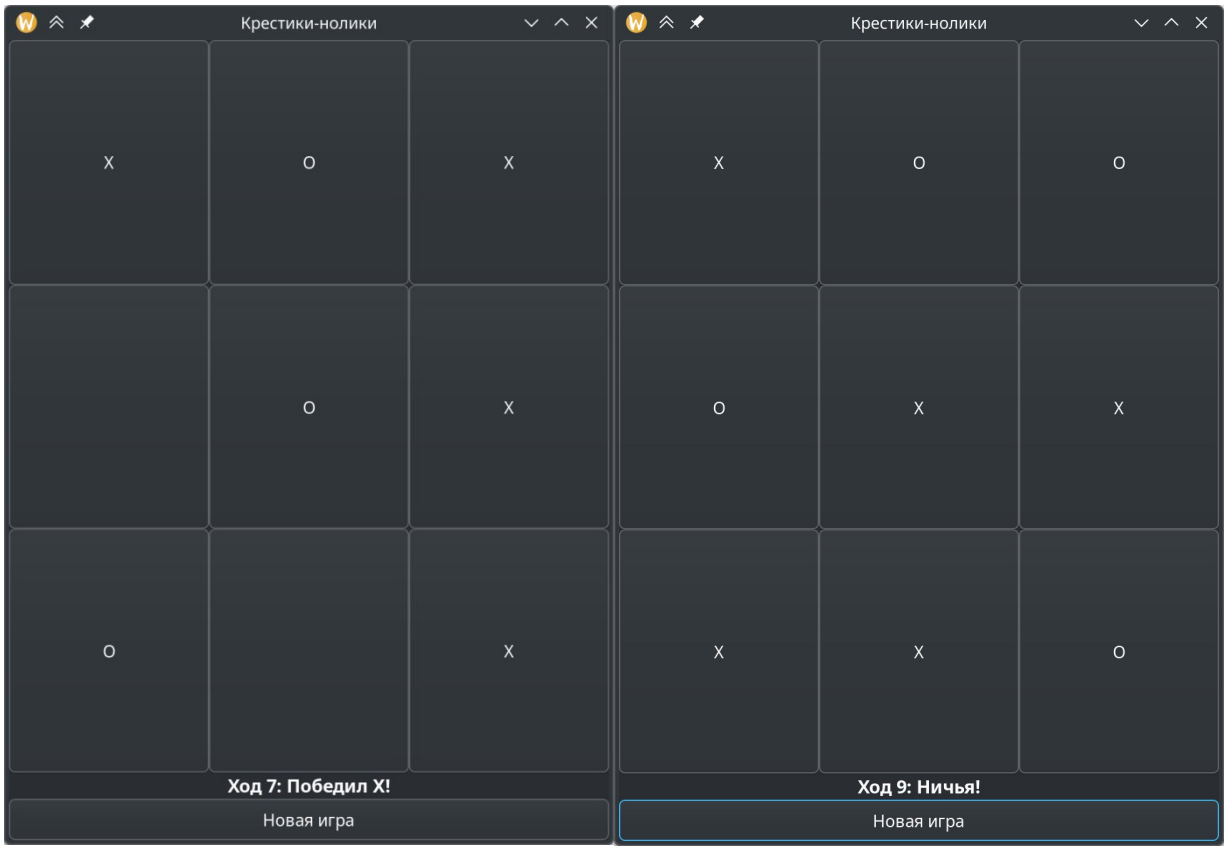

Рисунок 25 — Окно программы при победе (слева) и при ничье (справа)

Задание 4 Создание программной системы с графическим интерфейсом на C#

Задание: Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу.

База данных (файл) магазина хозяйственных товаров содержит сведения о поступлении товаров: наименование, дата поступления, количество штук, страна-производитель. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Определить, есть ли в магазине указанный товар производства данной страны.

2. Показать количество товара каждого наименования в магазине.

3. Определить товары (наименование, страна), поступившие в указанный период.

Выполнение:

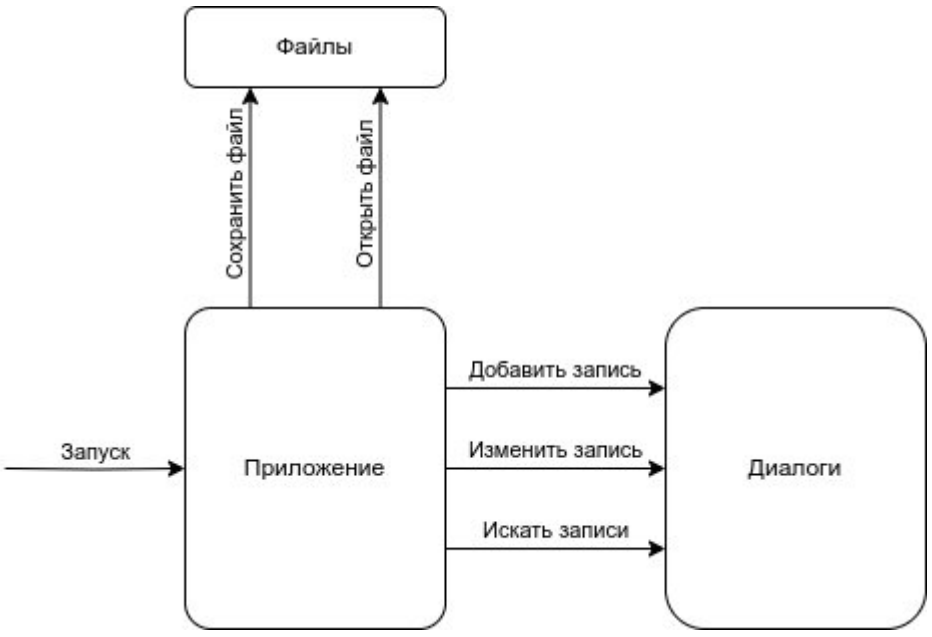

Рисунок 26 — Объектная декомпозиция

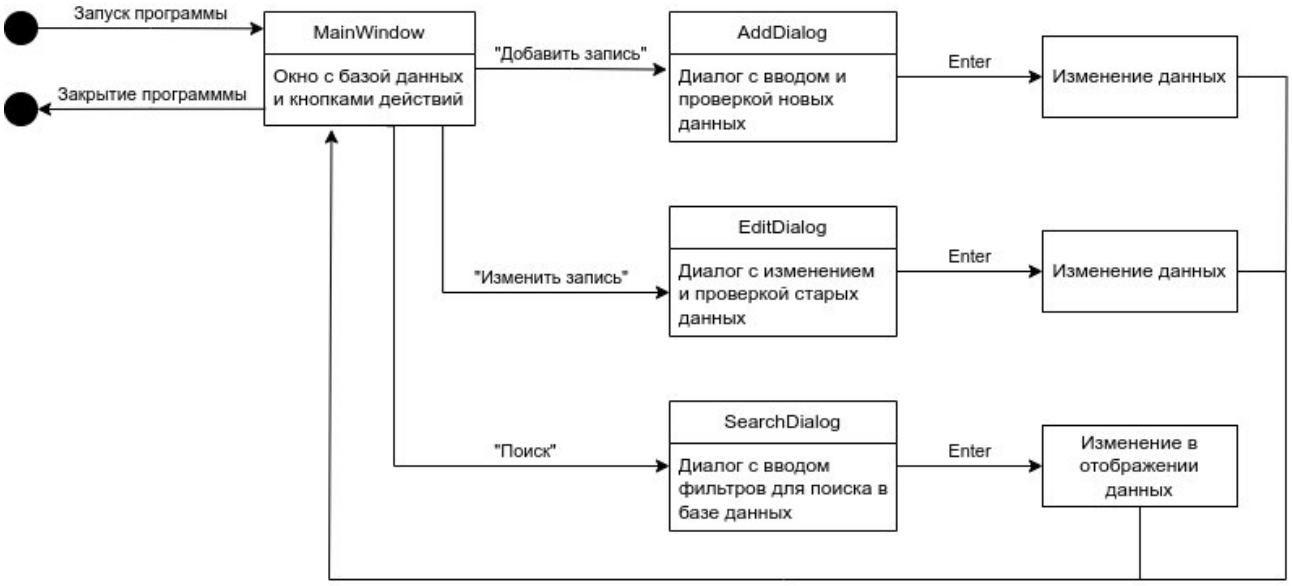

Рисунок 27 — Диаграмма классов интерфейса

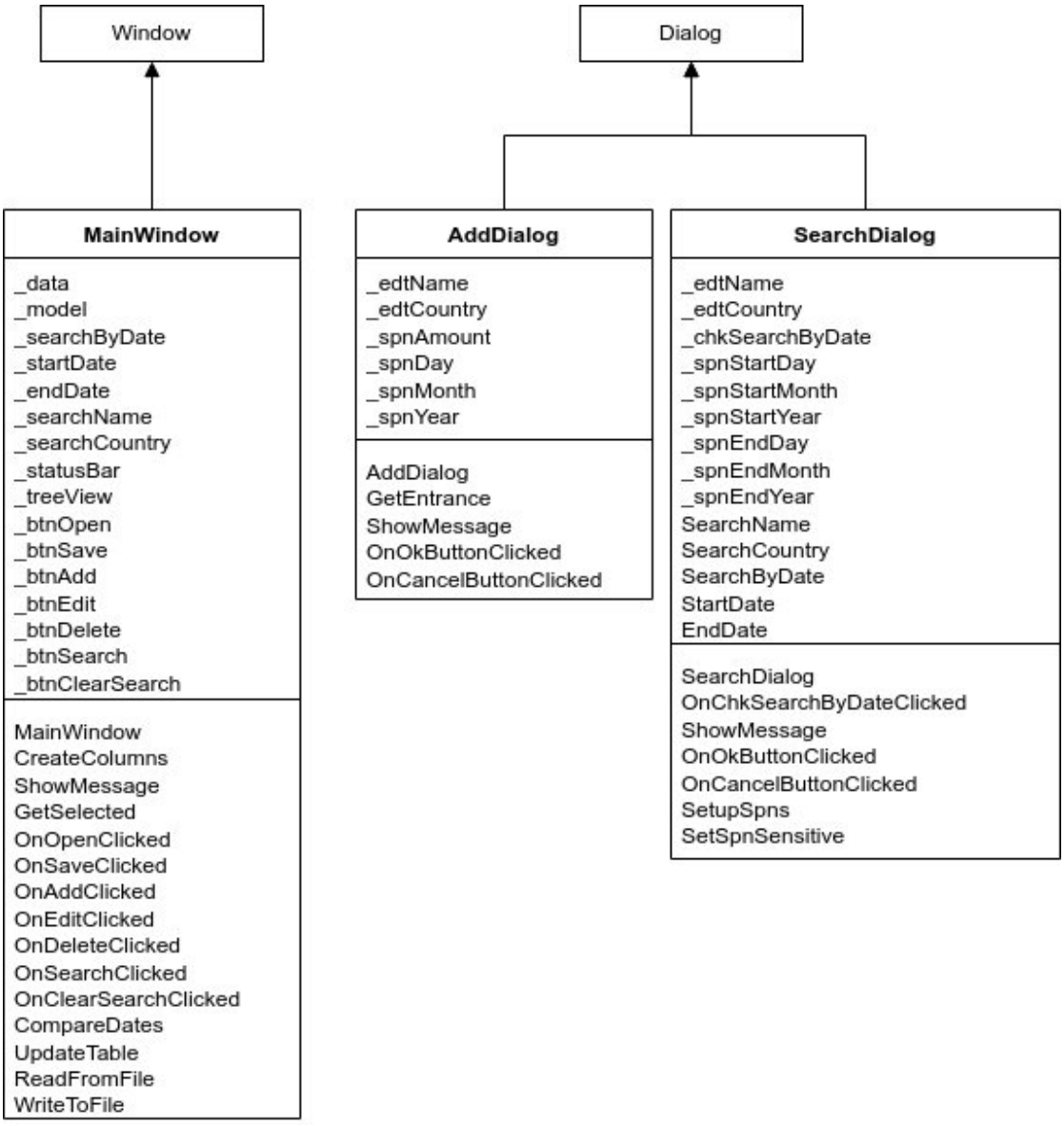

Рисунок 28 — Диаграмма классов интерфейсной области

```
public struct Entrance
{
    public string Name;
    public string Date;
    public int Amount;
    public string Country;
}
```

Рисунок 29 — Структура, отвечающая за данные

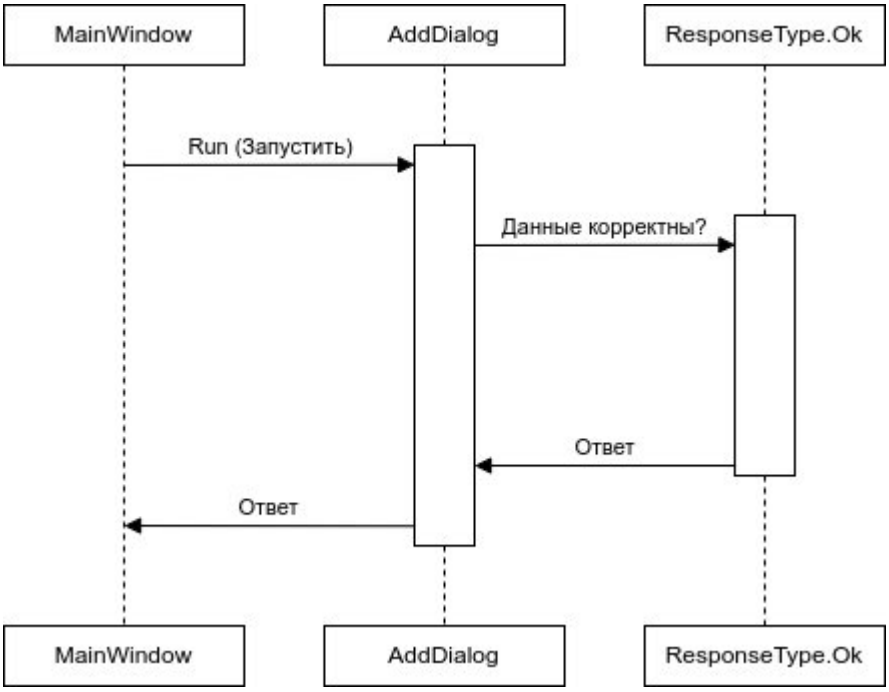

Рисунок 30 — Диаграмма последовательности действий

```
class Program
{
    [STAThread]
    public static void Main(string[] args)
    {
        Application.Init();

        var app = new Application("org._04._04", GLib.ApplicationFlags.None);
        app.Register(GLib.Cancellable.Current);

        var win = new MainWindow();
        app.AddWindow(win);

        win.Show();
        Application.Run();
    }
}
```

Рисунок 31 — Класс, отвечающий за запуск приложения

```
class MainWindow : Window
{
    private readonly List<Entrance> _data = new List<Entrance>();

    private readonly ListStore _model = new(
        params types: typeof(string),
        typeof(string),
        typeof(int),
        typeof(string)
    );

    private bool _searchByDate;
    private string _startDate = "";
    private string _endDate = "";
    private string _searchName = "";
    private string _searchCountry = "";

    [UI] private Statusbar _statusBar;
    [UI] private TreeView _treeView;

    [UI] private Button _btnOpen;
    [UI] private Button _btnSave;
    [UI] private Button _btnAdd;
    [UI] private Button _btnEdit;
    [UI] private Button _btnDelete;
    [UI] private Button _btnSearch;
    [UI] private Button _btnClearSearch;
```

Рисунок 32 — Инициализация класса MainWindow

```
private void OnOpenClicked(object sender, EventArgs e)
{
    _data.Clear();
    ReadFromFile();
}

private void OnSaveClicked(object sender, EventArgs e)
{
    WriteToFile();
}

protected void OnAddClicked(object sender, EventArgs e)
{
    var dialog = new AddDialog( parent: this);
    if (dialog.Run() == (int)ResponseType.Ok)
    {
        var newEntrance = dialog.GetEntrance();
        _data.Add(newEntrance);
        _model.AppendValues(newEntrance.Name, newEntrance.Date, newEntrance.Amount, newEntrance.Country);
    }

    dialog.Destroy();
}
```

Рисунок 33 — Обработка нажатий на кнопки

```
2 usages
private void ReadFromFile()
{
    try
    {
        var streamReader = new StreamReader( path: "data.txt");
        var lines :string[] = streamReader.ReadToEnd().Split('\n');
        for (var i = 0; i < lines.Length / 4; i++)
        {
            var newEntrance = new Entrance
            {
                Name = lines[i * 4],
                Date = lines[i * 4 + 1],
                Amount = int.Parse(lines[i * 4 + 2]),
                Country = lines[i * 4 + 3]
            };
            _data.Add(newEntrance);
        }

        streamReader.Close();
        ShowMessage($"Прочитано строк: {lines.Length / 4}");
        UpdateTable();
    }
    catch (Exception)
    {
        ShowMessage("Ошибка при чтении файла!");
    }
}
```

Рисунок 34 — Метод чтения данных из файла

```
private void UpdateTable()
{
    _model.Clear();
    foreach (var entrance in _data)
    {
        if (entrance.Name.Contains(_searchName) && entrance.Country.Contains(_searchCountry))
        {
            if (!_searchByDate ||
                (_searchByDate && CompareDates(_startDate, entrance.Date) &&
                 CompareDates(entrance.Date, _endDate)))
                _model.AppendValues(entrance.Name, entrance.Date, entrance.Amount, entrance.Country);
        }
    }
}
```

Рисунок 35 — Метод обновления таблицы

```
public class AddDialog : Dialog
{
    [UI] private Entry _edtName;
    [UI] private Entry _edtCountry;

    [UI] private SpinButton _spnAmount;
    [UI] private SpinButton _spnDay;
    [UI] private SpinButton _spnMonth;
    [UI] private SpinButton _spnYear;
```

```
public class SearchDialog : Dialog
{
    [UI] private Entry _edtName;
    [UI] private Entry _edtCountry;

    [UI] private CheckButton _chkSearchByDate;

    [UI] private SpinButton _spnStartDay;
    [UI] private SpinButton _spnStartMonth;
    [UI] private SpinButton _spnStartYear;
    [UI] private SpinButton _spnEndDay;
    [UI] private SpinButton _spnEndMonth;
    [UI] private SpinButton _spnEndYear;
```

Рисунок 36 — Инициализация диалогов

```
public Entrance GetEntrance()
{
    return new Entrance
    {
        Name = _edtName.Text,
        Date = $"{_spnDay.Value}/{_spnMonth.Value}/{_spnYear.Value}",
        Amount = (int)_spnAmount.Value,
        Country = _edtCountry.Text
    };
}

2 usages
private void ShowMessage(string message)
{
    var md = new MessageDialog(
        parent_window: this,
        DialogFlags.Modal,
        MessageType.Error,
        bt: ButtonsType.Ok,
        message
    )
    {
        Title = "Ошибка!"
    };

    md.Run();
    md.Destroy();
}
```

Рисунок 37 — Методы класса AddDialog

Тестирование программы:

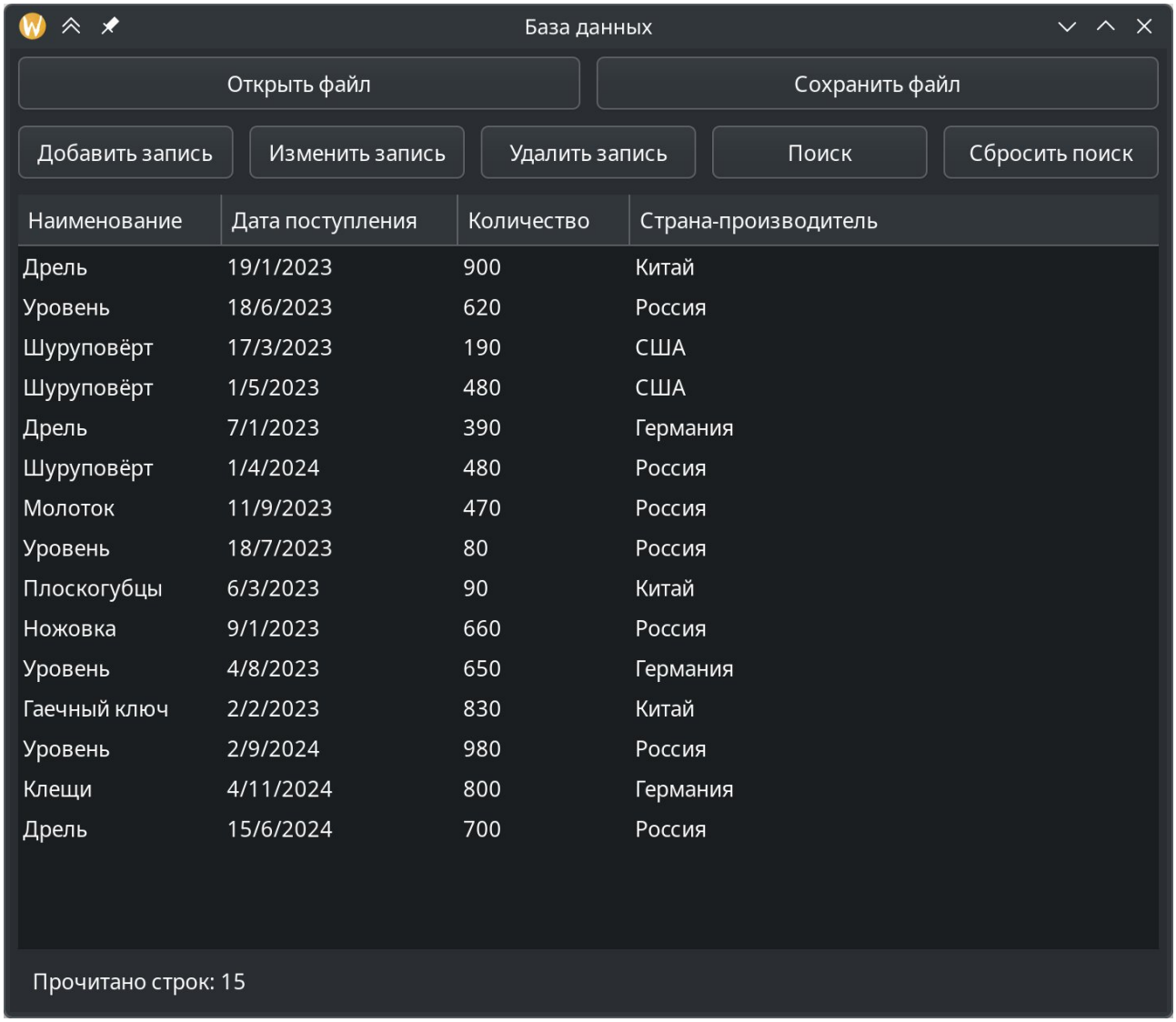

Рисунок 38 — Главное окно приложения

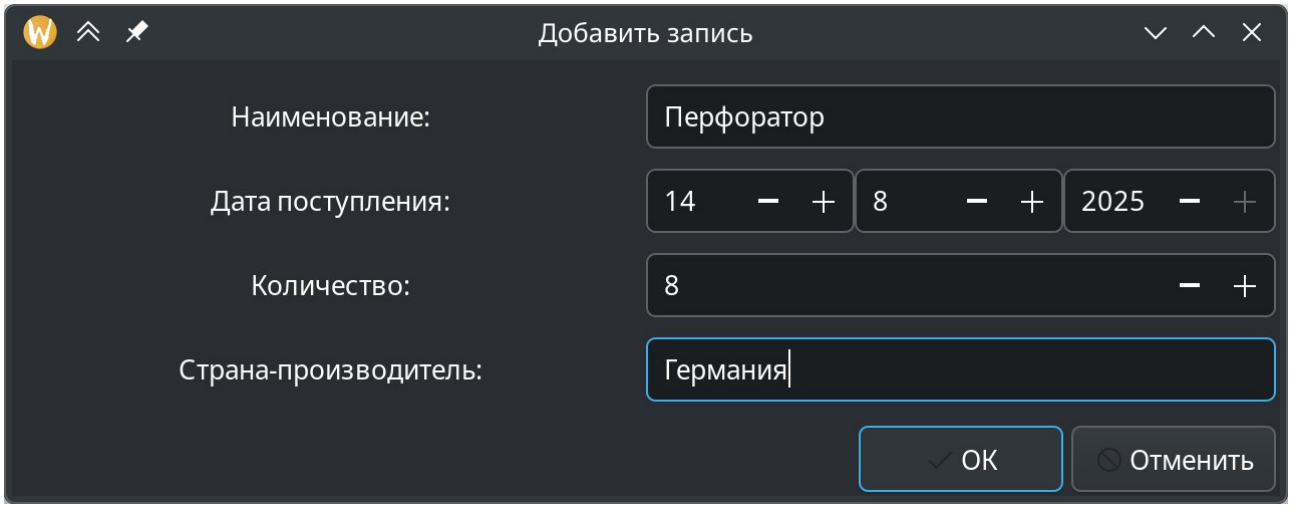

Рисунок 39 — Диалог добавления новой записи

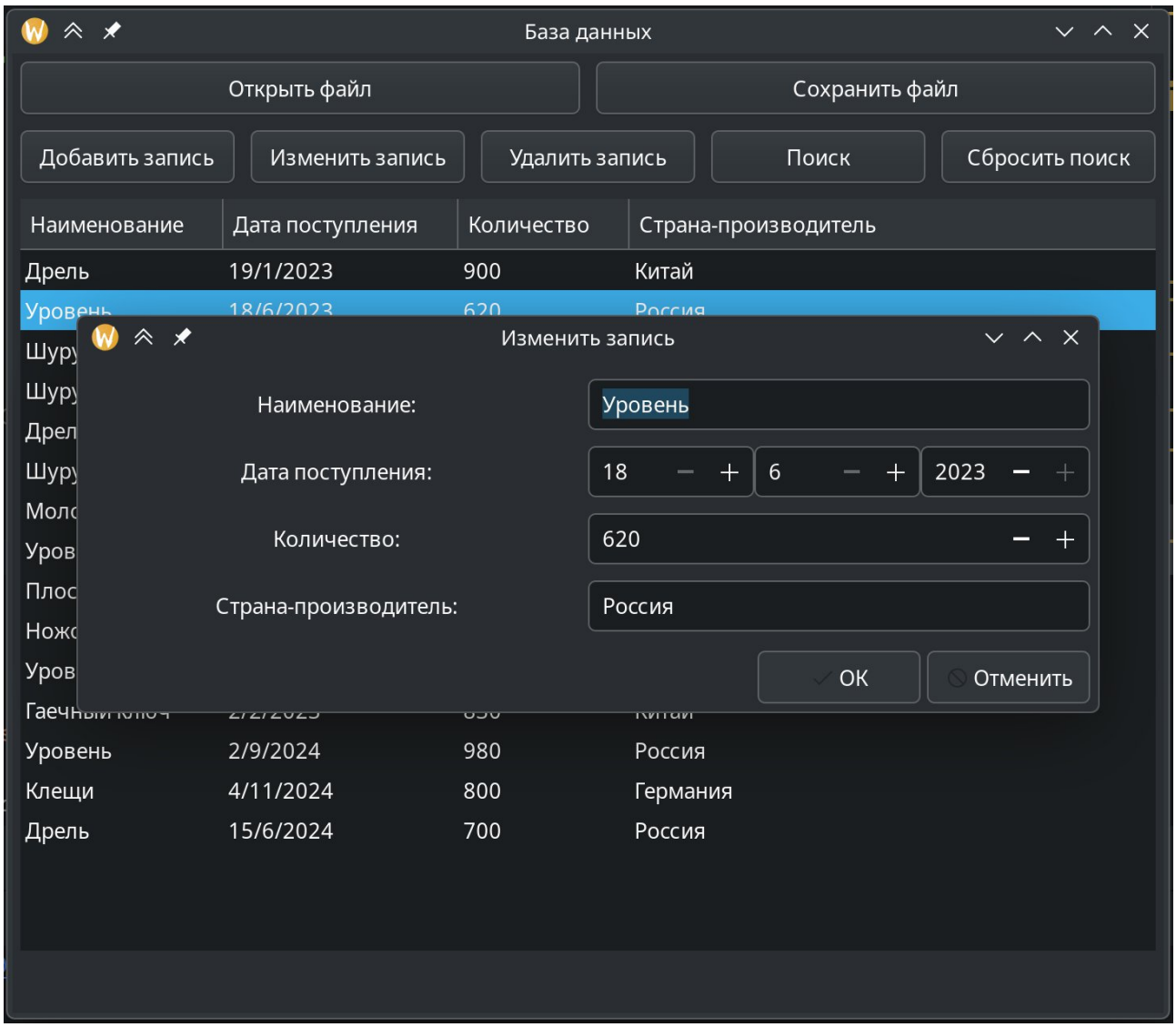

Рисунок 40 — Диалог изменения существующей записи

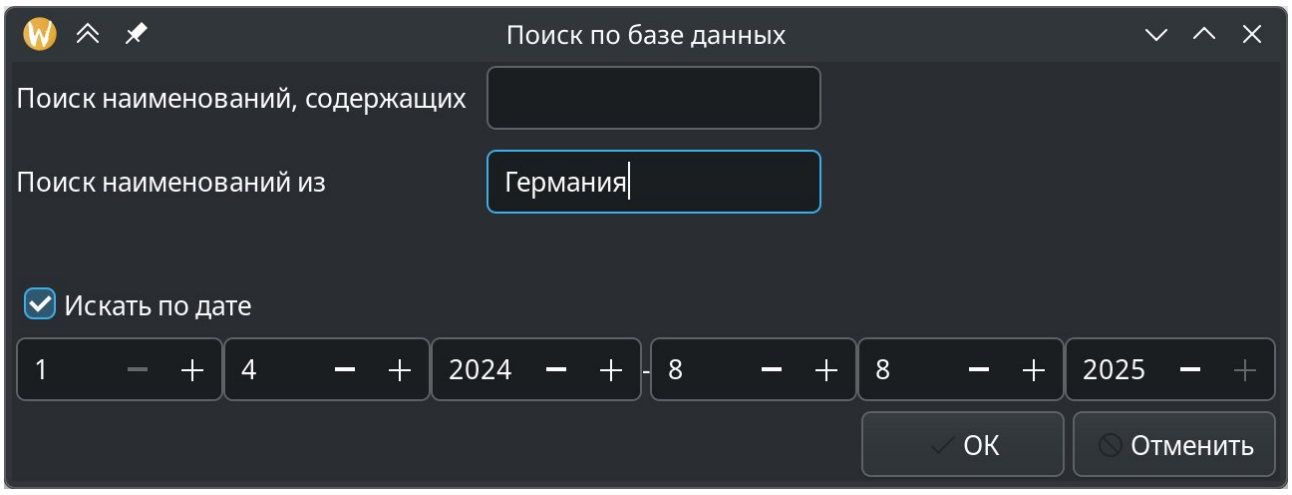

Рисунок 41 — Класс, отвечающий за запуск приложения

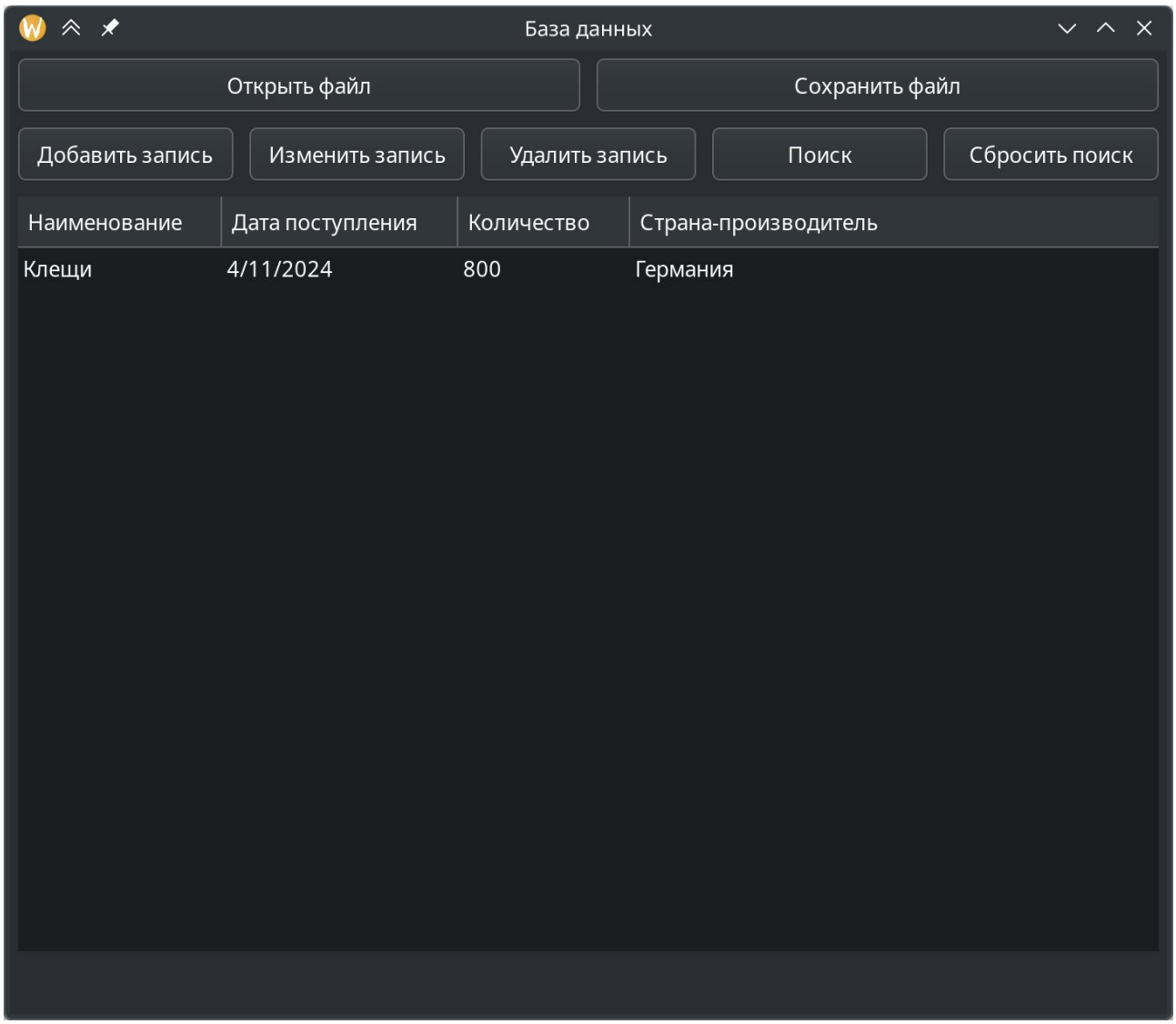

Рисунок 42 — Результат поиска

Вывод:

В ходе практики разработаны две программы на C++ (с Qt) и C# (GTK#) для управления БД магазина, а также игра крестики-нолики. В БД реализованы:

1. Графический интерфейс с формами добавления/удаления данных, запросами (наличие товара, фильтрация по дате, статистика).

2. UML-диаграммы (классов, состояний, последовательностей) для визуализации архитектуры.

3. Работа с файлами, построение графиков.

4. Тестирование подтвердило корректность операций и обработку ошибок.