System w urzędzie pocztowym

Paulina Czapla

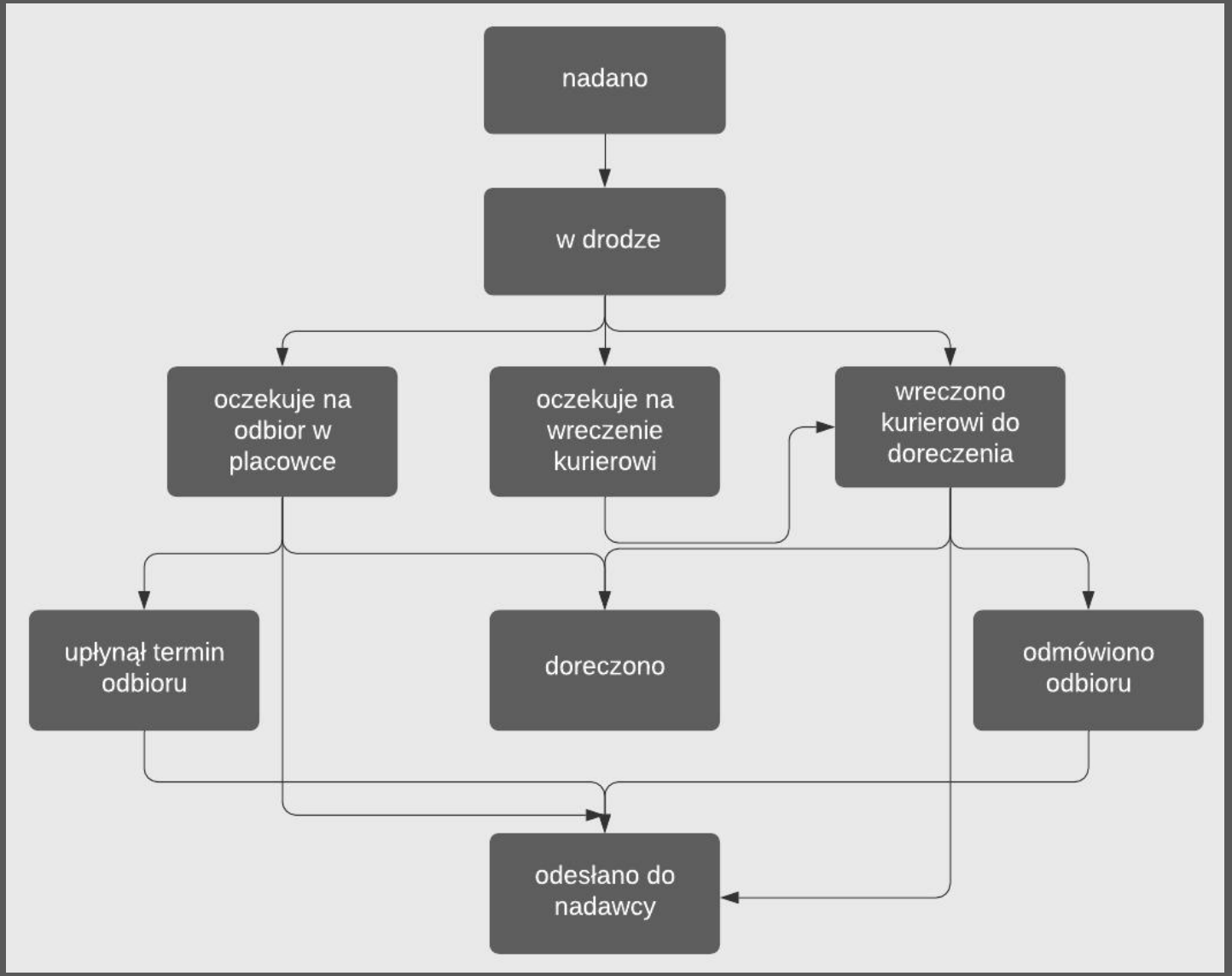
Ogólny opis programu

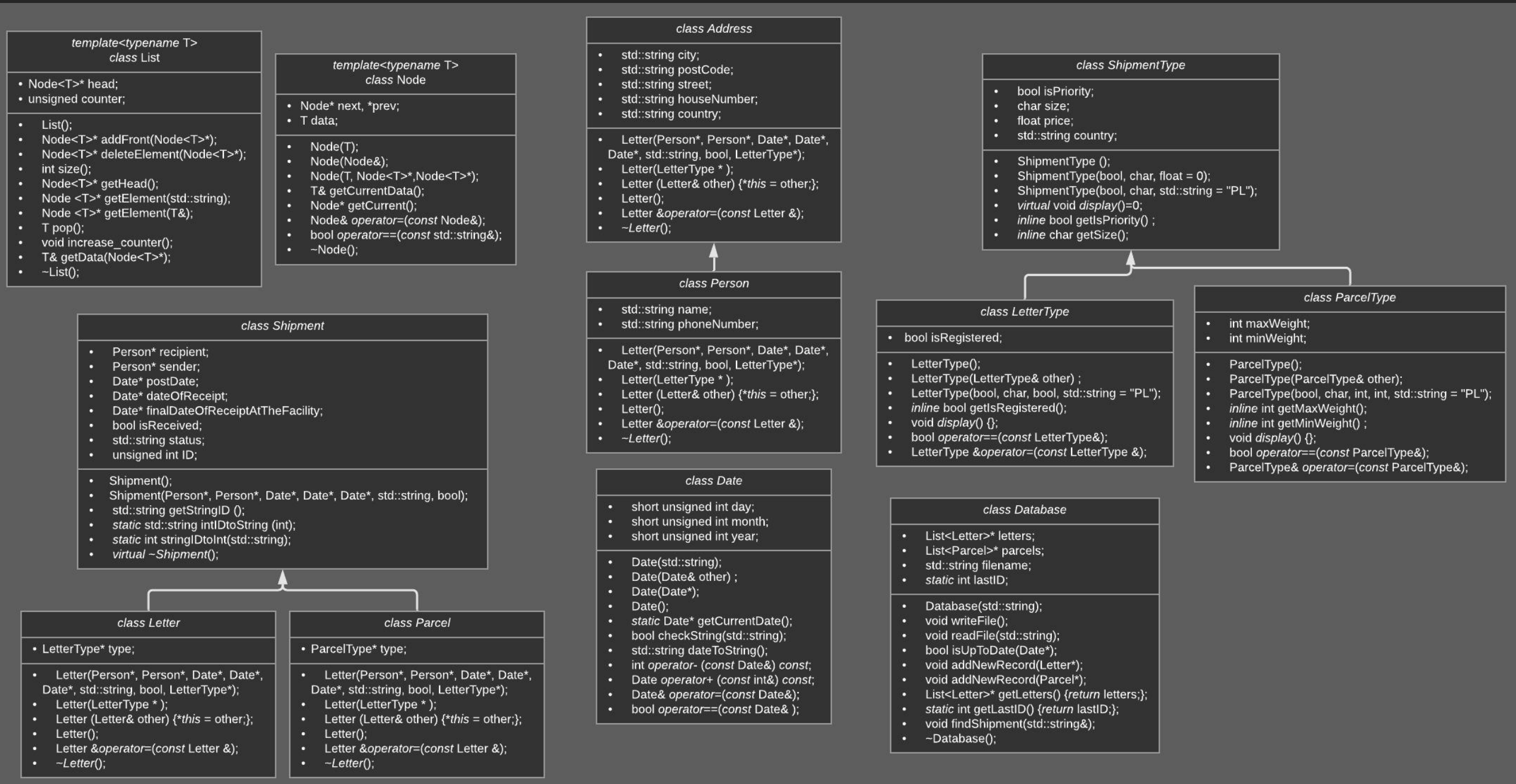
- System w urzędzie pocztowym, przeznaczony do użytku przez pracownika filii poczty;
- Program okienkowy zaprojektowany w technologii Qt;
- Funkcje programu:
  - nadanie przesyłki;
  - wyznaczenie ceny przesyłki w zależności od rodzaju;
  - przeglądanie baz danych: baza lokalna i główna;
  - szukanie rekordów w bazach po numerze telefonu odbiorcy/nadawcy i numerze przesyłki;
  - zmiana statusów przesyłek.

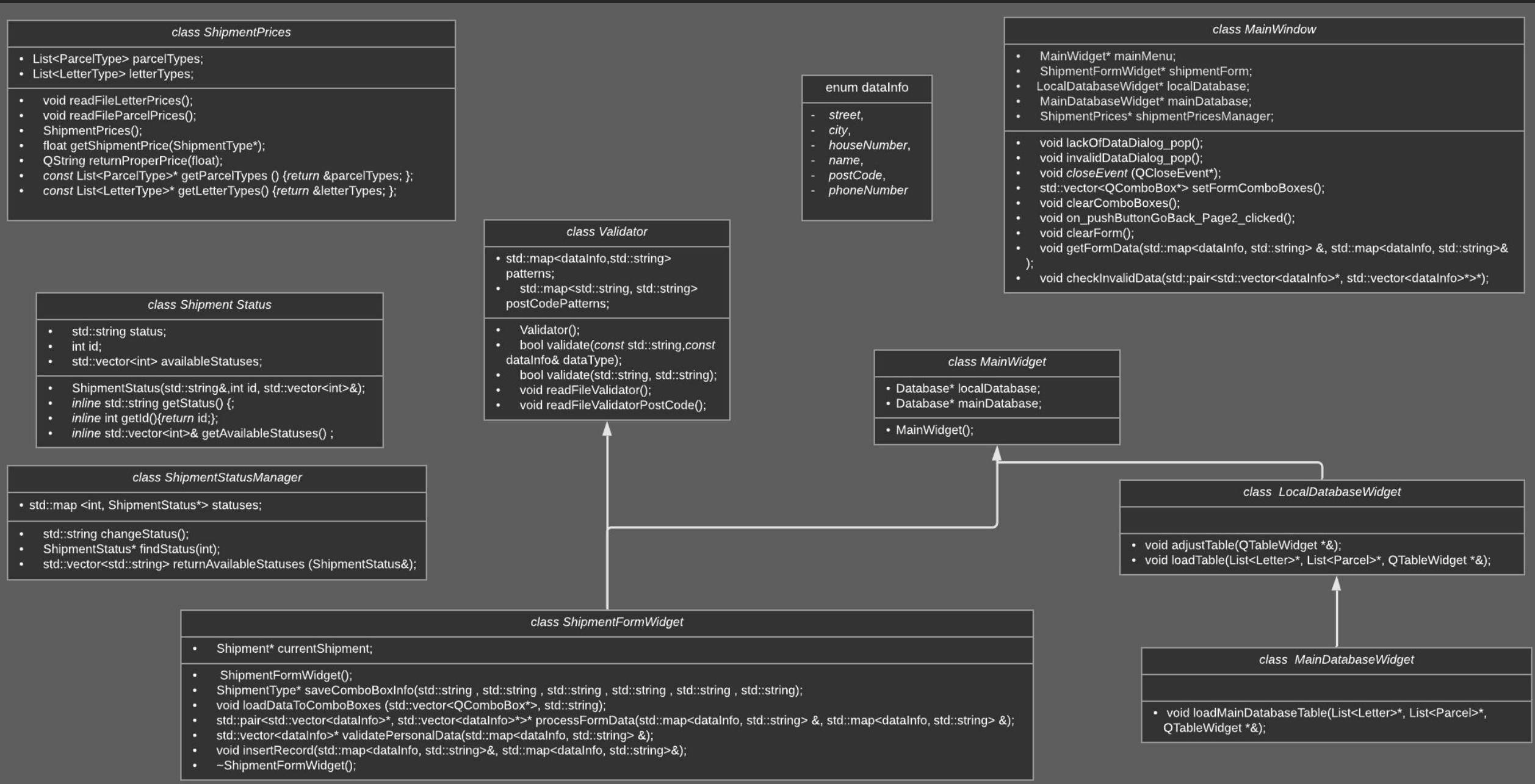
Założenia

- Przesyłka ze statusem „odebrano” zostaje usunięta na stałe z bazy danych po 7 dniach od terminu odbioru;
- Przesyłce ze statusem „oczekuje na odbiór w placówce” po przekroczeniu terminu na odebranie przesyłki, status zostaje zmieniony na „upłynął termin odbioru”;
- Użytkownik nie może dowolnie zmieniać statusów;
- Cena przesyłki ustalana jest tylko przy nadawaniu, nie jest zapisywana do bazy;
- Program obsługuje pliki .txt i .csv.

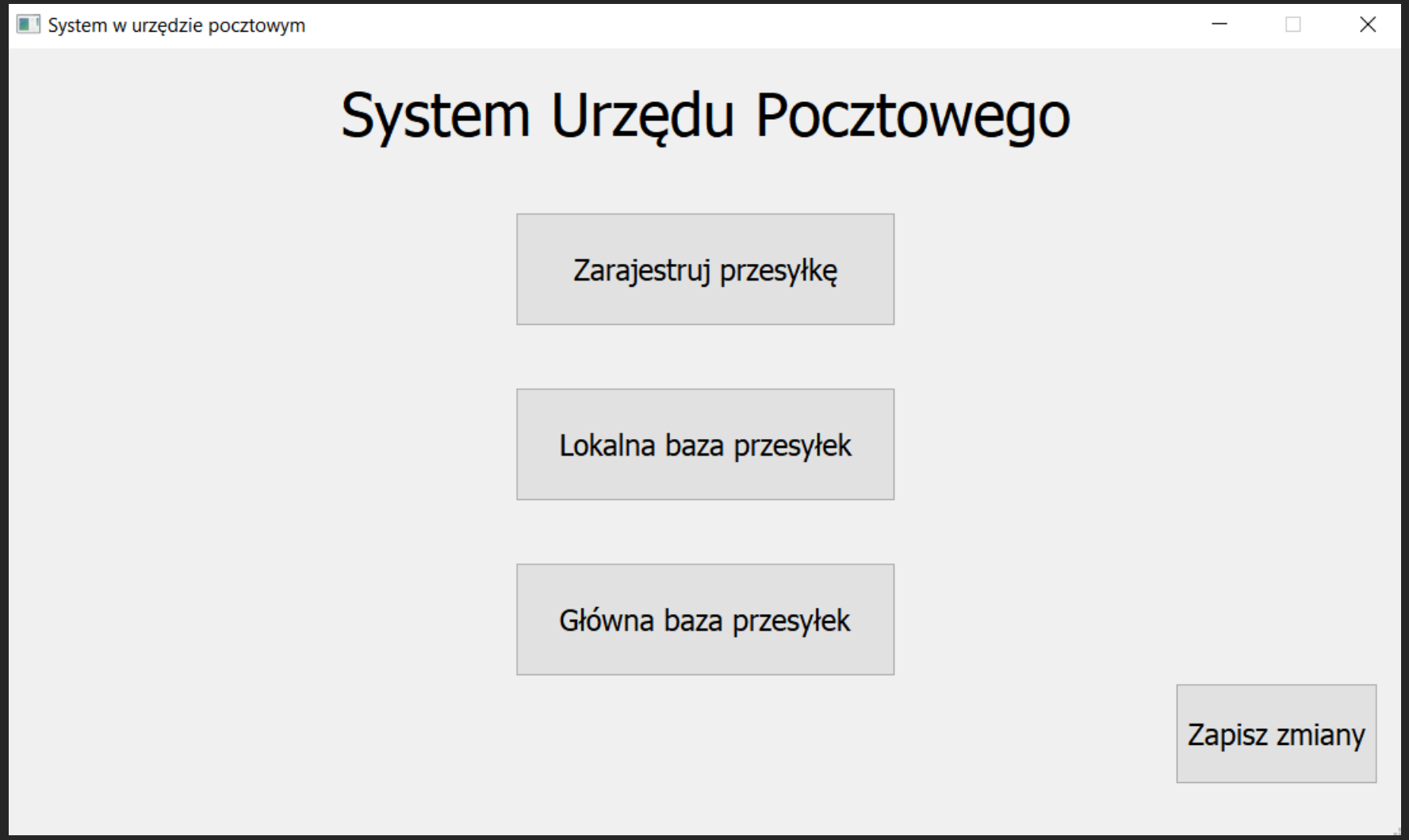
Statusy przesyłek

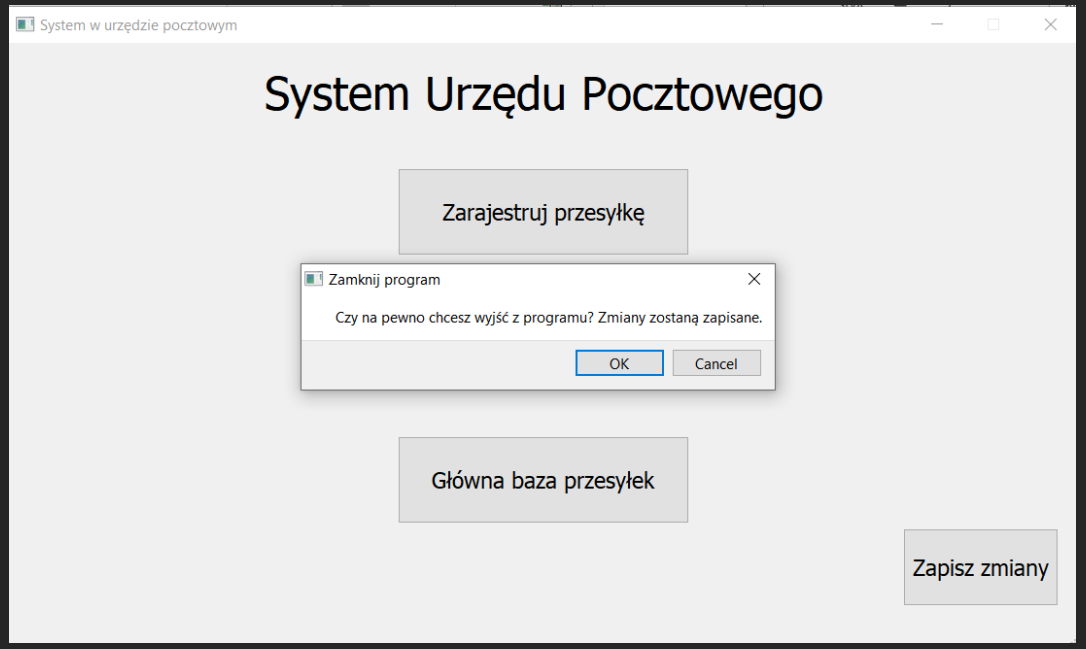



template<typename T>
class List
Node<T>* head;
unsigned counter;
List();
Node<T>* addFront(Node<T>*);
Node<T>* deleteElement(Node<T>*);
int size();
Node<T>* getHead();
Node <T>* getElement(std::string);
Node <T>* getElement(T&);
T pop();
void increase_counter();
T& getData(Node<T>*);
~List();
template<typename T>
class Node
Node* next, *prev;
T data;
Node(T);
Node(Node&);
Node(T, Node<T>*,Node<T>*);
T& getCurrentData();
Node* getCurrent();
Node& operator=(const Node&);
bool operator==(const std::string&);
~Node();
class Address
std::string city;
std::string postCode;
std::string street;
std::string houseNumber;
std::string country;
Letter(Person*, Person*, Date*, Date*, Date*, std::string, bool, LetterType*);
Letter(LetterType *);
Letter (Letter& other) {*this = other;};
Letter();
Letter &operator=(const Letter &);
~Letter();
class Person
std::string name;
std::string phoneNumber;
Letter(Person*, Person*, Date*, Date*, Date*, std::string, bool, LetterType*);
Letter(LetterType *);
Letter (Letter& other) {*this = other;};
Letter();
Letter &operator=(const Letter &);
~Letter();
class ShipmentType
bool isPriority;
char size;
float price;
std::string country;
ShipmentType ();
ShipmentType(bool, char, float = 0);
ShipmentType(bool, char, std::string = "PL");
virtual void display()=0;
inline bool getIsPriority() ;
inline char getSize();
class LetterType
bool isRegistered;
LetterType();
LetterType(LetterType& other) ;
LetterType(bool, char, bool, std::string = "PL");
inline bool getIsRegistered();
void display() {};
bool operator==(const LetterType&);
LetterType &operator=(const LetterType &);
class ParcelType
int maxWeight;
int minWeight;
ParcelType();
ParcelType(ParcelType& other);
ParcelType(bool, char, int, int, std::string = "PL");
inline int getMaxWeight();
inline int getMinWeight() ;
void display() {};
bool operator==(const ParcelType&);
ParcelType& operator=(const ParcelType&);
class Shipment
Person* recipient;
Person* sender;
Date* postDate;
Date* dateOfReceipt;
Date* finalDateOfReceiptAtTheFacility;
bool isReceived;
std::string status;
unsigned int ID;
Shipment();
Shipment(Person*, Person*, Date*, Date*, Date*, std::string, bool);
std::string getStringID ();
static std::string intIDtoString (int);
static int stringIDtoInt(std::string);
virtual ~Shipment();
class Date
short unsigned int day;
short unsigned int month;
short unsigned int year;
Date(std::string);
Date(Date& other) ;
Date(Date*);
Date();
static Date* getCurrentDate();
bool checkString(std::string);
std::string dateToString();
int operator- (const Date&) const;
Date operator+ (const int&) const;
Date& operator=(const Date&);
bool operator==(const Date&);
class Database
List<Letter>* letters;
List<Parcel>* parcels;
std::string filename;
static int lastID;
Database(std::string);
void writeFile();
void readFile(std::string);
bool isUpToDate(Date*);
void addNewRecord(Letter*);
void addNewRecord(Parcel*);
List<Letter>* getLetters() {return letters;};
static int getLastID() {return lastID;};
void findShipment(std::string&);
~Database();
class Letter
LetterType* type;
Letter(Person*, Person*, Date*, Date*, Date*, std::string, bool, LetterType*);
Letter(LetterType *);
Letter (Letter& other) {*this = other;};
Letter();
Letter &operator=(const Letter &);
~Letter();
class Parcel
ParcelType* type;
Letter(Person*, Person*, Date*, Date*, Date*, std::string, bool, LetterType*);
Letter(LetterType *);
Letter (Letter& other) {*this = other;};
Letter();
Letter &operator=(const Letter &);
~Letter();

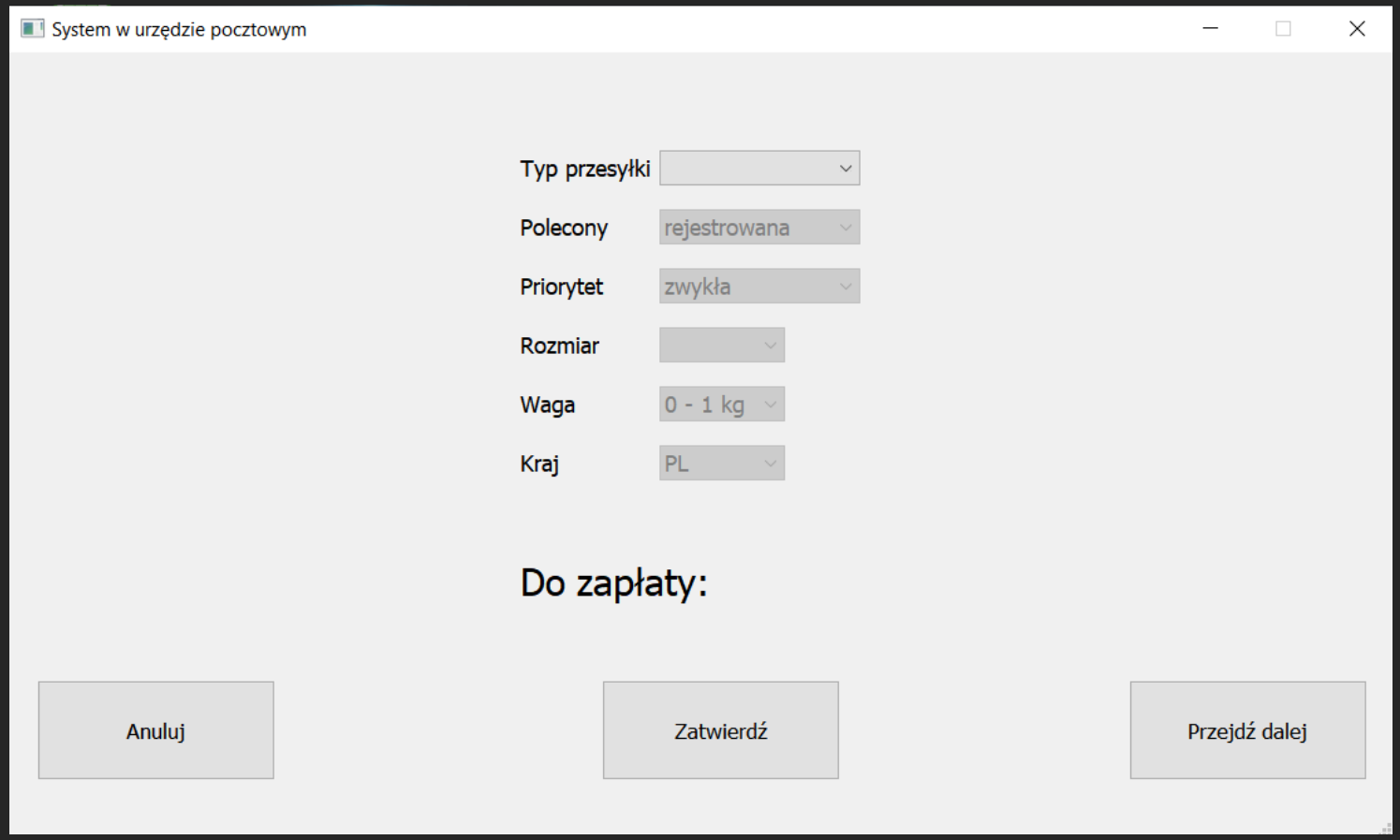

class ShipmentPrices
• List<ParcelType> parcelTypes;
• List<LetterType> letterTypes;
• void readFileLetterPrices();
• void readFileParcelPrices();
• ShipmentPrices();
• float getShipmentPrice(ShipmentType*);
• QString returnProperPrice(float);
• const List<ParcelType>* getParcelTypes () {return &parcelTypes; };
• const List<LetterType>* getLetterTypes() {return &letterTypes; };
enum dataInfo
- street,
- city,
- houseNumber,
- name,
- postCode,
- phoneNumber
class MainWindow
• MainWidget* mainMenu;
• ShipmentFormWidget* shipmentForm;
• LocalDatabaseWidget* localDatabase;
• MainDatabaseWidget* mainDatabase;
• ShipmentPrices* shipmentPricesManager;
• void lackOfDataDialog_pop();
• void invalidDataDialog_pop();
• void closeEvent (QCloseEvent*);
• std::vector<QComboBox*> setFormComboBoxes();
• void clearComboBoxes();
• void on_pushButtonGoBack_Page2_clicked();
• void clearForm();
• void getFormData(std::map<dataInfo, std::string> &, std::map<dataInfo, std::string>&);
• void checkInvalidData(std::pair<std::vector<dataInfo>*, std::vector<dataInfo>*>*);
class Validator
• std::map<dataInfo,std::string> patterns;
• std::map<std::string, std::string> postCodePatterns;
• Validator();
• bool validate(const std::string,const dataInfo& dataType);
• bool validate(std::string, std::string);
• void readFileValidator();
• void readFileValidatorPostCode();
class Shipment Status
• std::string status;
• int id;
• std::vector<int> availableStatuses;
• ShipmentStatus(std::string&,int id, std::vector<int>&);
• inline std::string getStatus() {;
• inline int getId(){return id;};
• inline std::vector<int>& getAvailableStatuses() ;
class MainWidget
• Database* localDatabase;
• Database* mainDatabase;
• MainWidget();
class ShipmentStatusManager
• std::map <int, ShipmentStatus*> statuses;
• std::string changeStatus();
• ShipmentStatus* findStatus(int);
• std::vector<std::string> returnAvailableStatuses (ShipmentStatus&);
class LocalDatabaseWidget
• void adjustTable(QTableWidget *&);
• void loadTable(List<Letter>*, List<Parcel>*, QTableWidget *&);
class ShipmentFormWidget
• Shipment* currentShipment;
• ShipmentFormWidget();
• ShipmentType* saveComboBoxInfo(std::string , std::string , std::string , std::string , std::string , std::string);
• void loadDataToComboBoxes (std::vector<QComboBox*>, std::string);
• std::pair<std::vector<dataInfo>*, std::vector<dataInfo>*>* processFormData(std::map<dataInfo, std::string> &, std::map<dataInfo, std::string> &);
• std::vector<dataInfo>* validatePersonalData(std::map<dataInfo, std::string> &);
• void insertRecord(std::map<dataInfo, std::string>&, std::map<dataInfo, std::string>&);
• ~ShipmentFormWidget();
class MainDatabaseWidget
• void loadMainDatabaseTable(List<Letter>*, List<Parcel>*, QTableWidget *&);

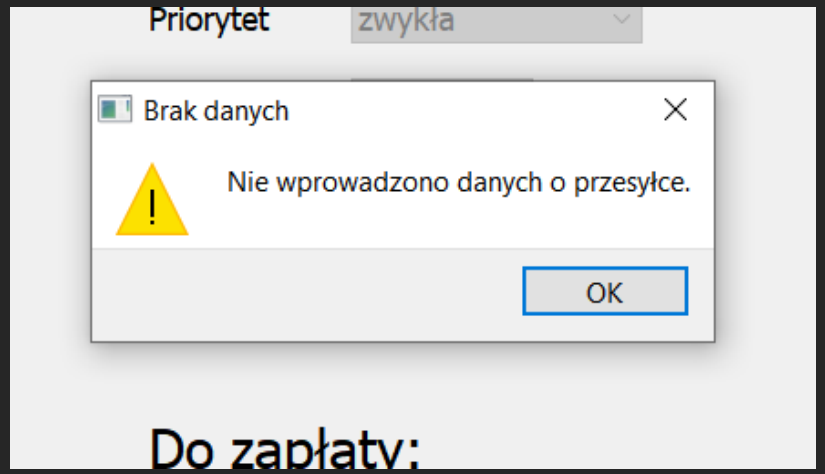
Aktualny stan projektu

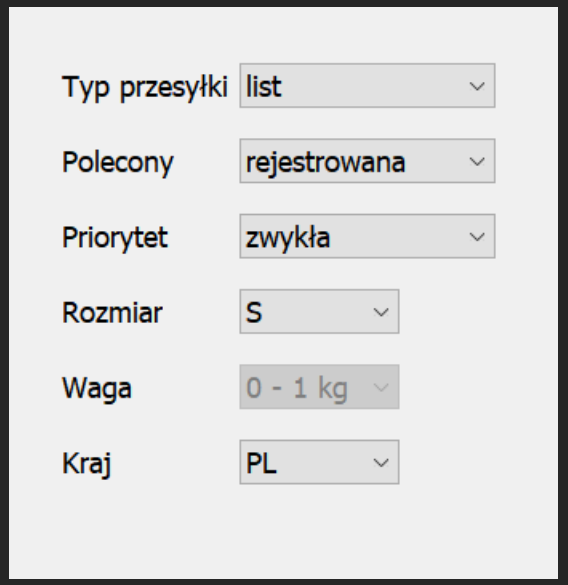

System w urzędzie pocztowym
System Urzędu Pocztowego
Zarajestruj przesyłkę
Lokalna baza przesyłek
Główna baza przesyłek
Zapisz zmiany

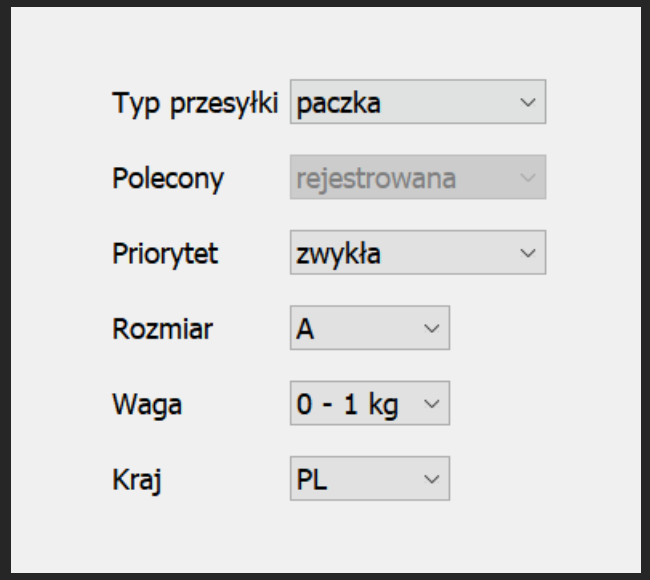


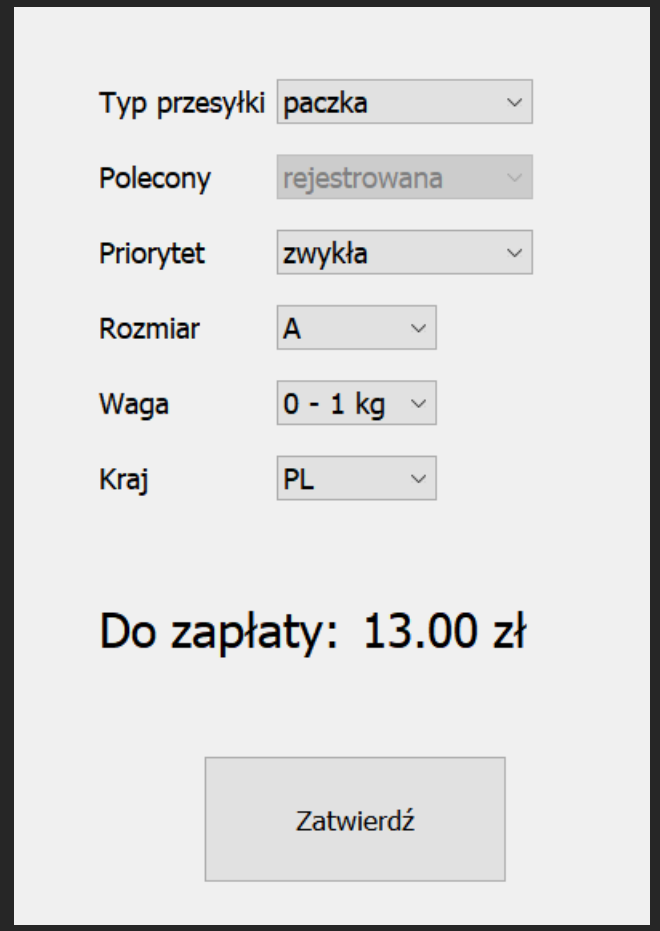

```
void MainWindow::closeEvent (QCloseEvent *event)
{
    QMessageBox msgBox;
    msgBox.setWindowTitle("Zamknij program");
    msgBox.setText("Czy na pewno chcesz wyjść z programu? Zmiany zostaną zapisane.");
    msgBox.setStandardButtons(QMessageBox::Ok| QMessageBox::Cancel);
    int ret = msgBox.exec();

    if(ret==QMessageBox::Ok)
    {
        saveDatabase();
         event->accept();
        exit(1);
    }
    if(ret==QMessageBox::Cancel)
    {
        event->ignore();
    }
}
```


System w urzędzie pocztowym
Typ przesyłki
Polecony
rejestrowana
Priorytet
zwykła
Rozmiar
Waga
0 - 1 kg
Kraj
PL
Do zapłaty:
Anuluj
Zatwierdź
Przejdź dalej


Priorytet
zwykła
Brak danych
Nie wprowadzono danych o przesyłce.
OK
Do zapłaty:


Typ przesyłki
list
Polecony
rejestrowana
Priorytet
zwykła
Rozmiar
S
Waga
0 - 1 kg
Kraj
PL


Typ przesyłki
paczka
Polecony
rejestrowana
Priorytet
zwykła
Rozmiar
A
Waga
0 - 1 kg
Kraj
PL


Typ przesyłki
paczka
Polecony
rejestrowana
Priorytet
zwykła
Rozmiar
A
Waga
0 - 1 kg
Kraj
PL
Do zapłaty: 13.00 zł
Zatwierdź

```
class ShipmentPrices
{
    List<ParcelType> parcelTypes;
    List<LetterType> letterTypes;

    void readFileLetterPrices();
    void readFileParcelPrices();
public:
    ShipmentPrices();
    float getShipmentPrice(ShipmentType*);
    QString returnProperPrice(float);
    const List<ParcelType>* getParcelTypes () {return &parcelTypes; };
    const List<LetterType>* getLetterTypes() {return &letterTypes; };

};
```

```
float ShipmentPrices::getShipmentPrice(ShipmentType* type)
{
    if(typeid(*type).name()==typeid (ParcelType).name())
    {
       auto parcelType= dynamic_cast<ParcelType*>(type);
       auto found = parcelTypes.getElement(*parcelType);
       return found->getCurrentData().getPrice();

    } else if (typeid(*type).name()==typeid (LetterType).name())
    {
        auto letterType= dynamic_cast<LetterType*>(type);
        auto found = letterTypes.getElement(*letterType);
        return found->getCurrentData().getPrice();
    }
    return 0;
}
```

System w urzędzie pocztowym

Numer przesyłki: 000000008

Dane nadawcy

Imię i nazwisko	Katarzyna Kowalska	
Nr telefonu	111444777	
Kod pocztowy	42-506	
Ulica i numer mieszkania	akacjowa	12/4a
Miasto	Warszawa	

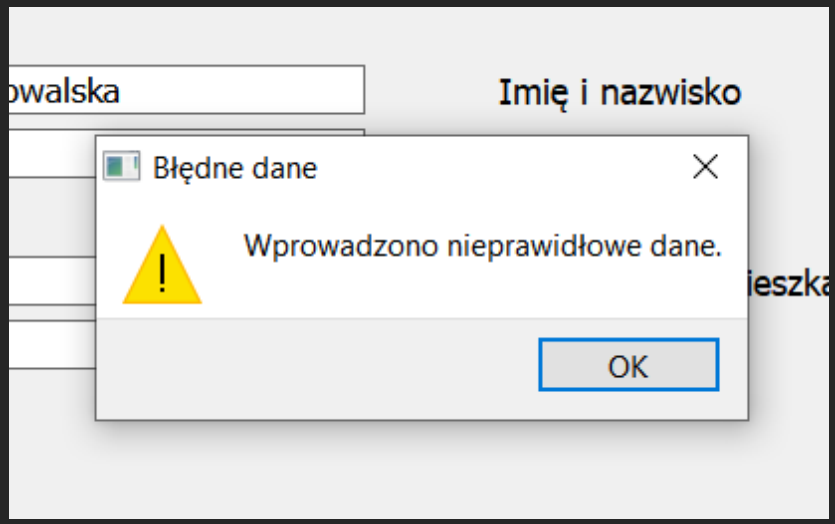
Dane Odbiorcy

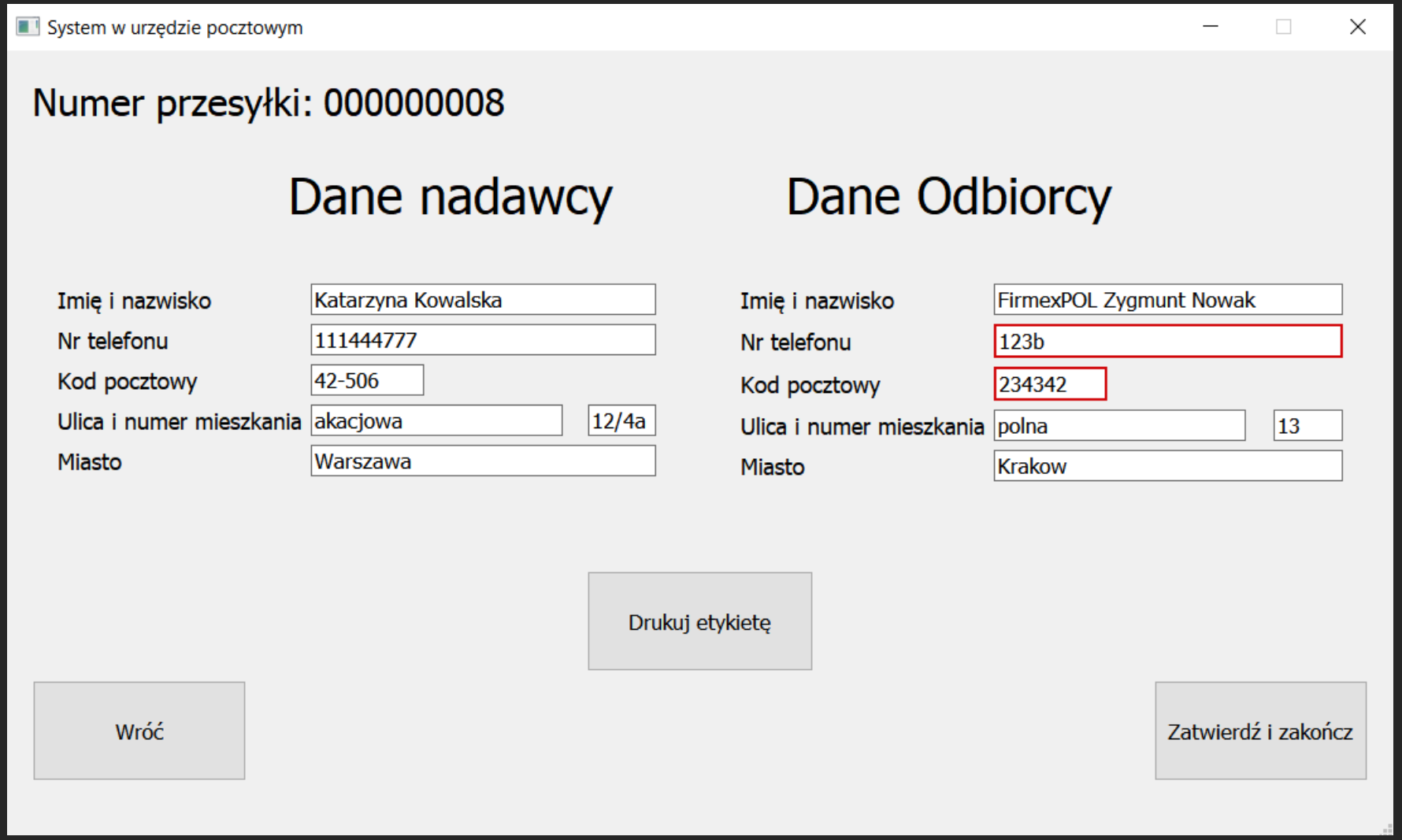
Imię i nazwisko	FirmexPOL Zygmunt Nowak	
Nr telefonu	123b	
Kod pocztowy	234342	
Ulica i numer mieszkania	polna	13
Miasto	Krakow	

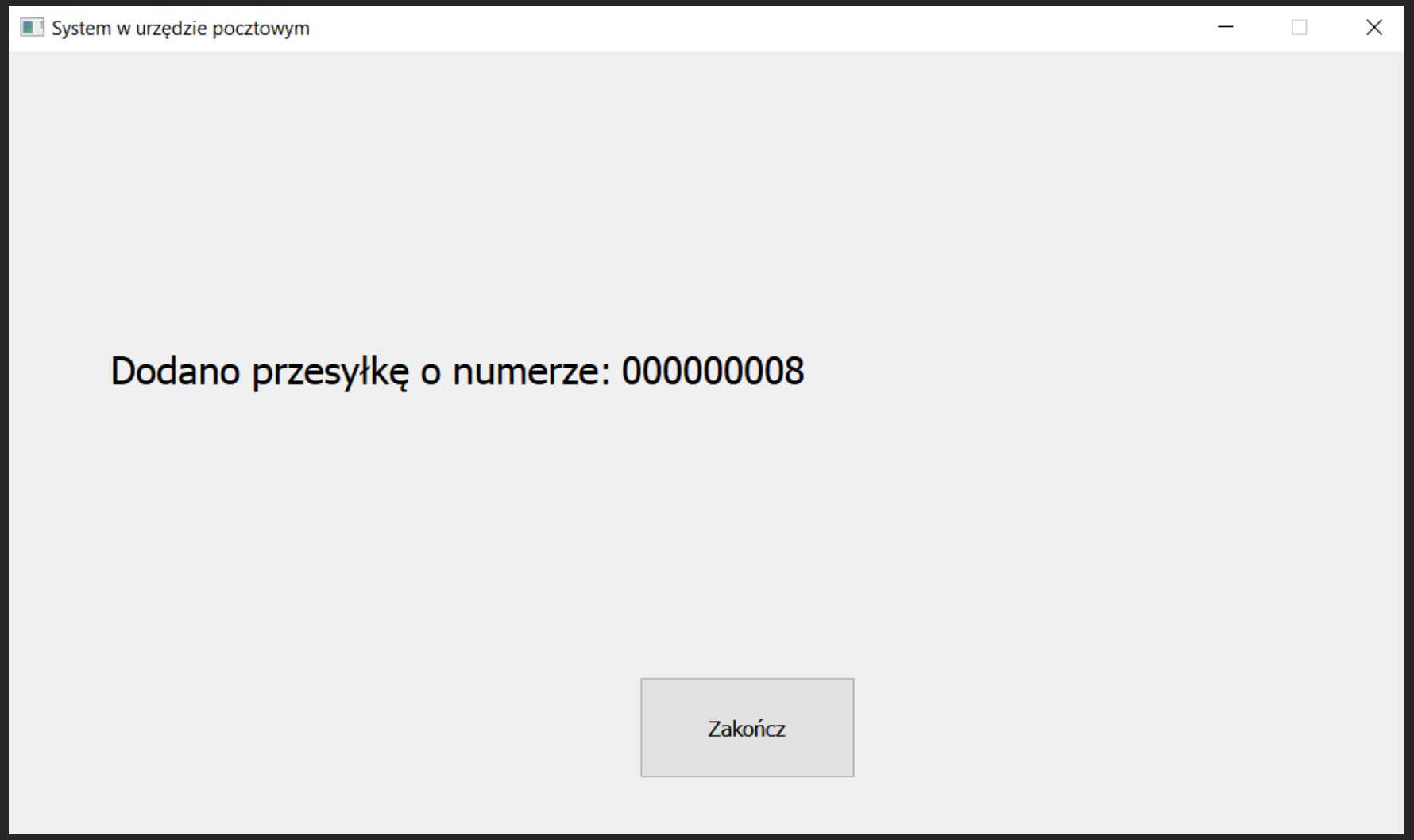
Drukuj etykietę

Wróć

Zatwierdź i zakończ

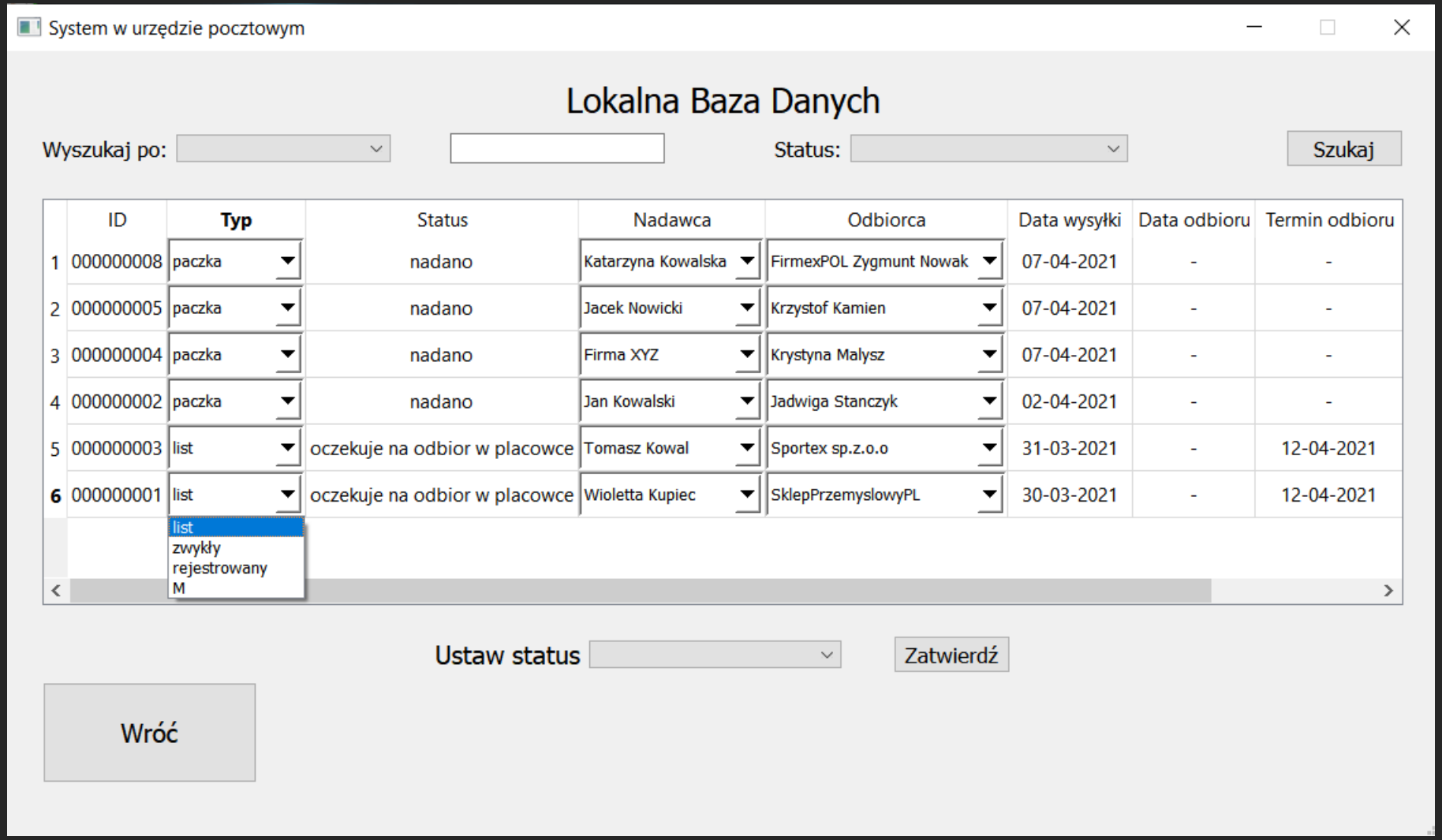



System w urzędzie pocztowym
Numer przesyłki: 000000008
Dane nadawcy
Imię i nazwisko
Katarzyna Kowalska
Nr telefonu
111444777
Kod pocztowy
42-506
Ulica i numer mieszkania
akacjowa
12/4a
Miasto
Warszawa
Dane Odbiorcy
Imię i nazwisko
FirmexPOL Zygmunt Nowak
Nr telefonu
123b
Kod pocztowy
234342
Ulica i numer mieszkania
polna
13
Miasto
Krakow
Drukuj etykietę
Wróć
Zatwierdź i zakończ


System w urzędzie pocztowym
Dodano przesyłkę o numerze: 000000008
Zakończ

```
std::pair<std::vector<dataInfo>*, std::vector<dataInfo>*>* ShipmentFormWidget::processFormData(std::map<dataInfo, std::string> & sender,
                                                                                               std::map<dataInfo, std::string> & recipient)
{
    auto senderInvalid = validatePersonalData(sender);
    auto recipientInvalid = validatePersonalData(recipient);

    std::pair <std::vector<dataInfo>*, std::vector<dataInfo>*> * result = new std::pair <std::vector<dataInfo>*, std::vector<dataInfo>*>;
    *result = std::make_pair(senderInvalid, recipientInvalid);
    return result;
}
```

```
std::vector<dataInfo>* ShipmentFormWidget::validatePersonalData(std::map<dataInfo, std::string> & person)
{
    //wykorzystanie iteratorów i kontenerów
    std::vector<dataInfo>* invalidData = new std::vector<dataInfo>;
    std::back_insert_iterator<std::vector<dataInfo>> insert(*invalidData);

    for (std::map<dataInfo, std::string>::iterator it=person.begin() ; it != person.end(); ++it)
    {
        if(!validate(it->second,it->first))
            insert =it->first;
    }
    return invalidData;
}
```


System w urzędzie pocztowym
Lokalna Baza Danych
Wyszukaj po:
Status:
Szukaj
ID
Typ
Status
Nadawca
Odbiorca
Data wysyłki
Data odbioru
Termin odbioru
1 000000008 paczka nadano Katarzyna Kowalska FirmexPOL Zygmunt Nowak 07-04-2021 - -
2 000000005 paczka nadano Jacek Nowicki Krzystof Kamien 07-04-2021 - -
3 000000004 paczka nadano Firma XYZ Krystyna Malysz 07-04-2021 - -
4 000000002 paczka nadano Jan Kowalski Jadwiga Stanczyk 02-04-2021 - -
5 000000003 list oczekuje na odbior w placowce Tomasz Kowal Sportex sp.z.o.o 31-03-2021 - 12-04-2021
6 000000001 list oczekuje na odbior w placowce Wioletta Kupiec SklepPrzemyslowyPL 30-03-2021 - 12-04-2021
list
zwykły
rejestrowany
M
Ustaw status
Zatwierdź
Wróć

System w urzędzie pocztowym

Główna Baza Danych

Wyszukaj po: Status: Szukaj

	ID	Typ	Status	Nadawca	Odbiorca	Data wysyłki	Data odbioru	Te
1	000000009	paczka	nadano	Katarzyna Kowalska	FirmexPOL Zygmunt Nowak	07-04-2021	-	
2	000000005	paczka	nadano	Jacek Nowicki	Krzystof Kamien	07-04-2021	-	
3	000000004	paczka	nadano	Firma XYZ	Krystyna Malysz	07-04-2021	-	
4	000000002	paczka	nadano	Jan Kowalski	Jadwiga Stanczyk	02-04-2021	-	
5	000000007	list	odebrano	Artykuly plastyczne PLASTEX	Karol Malarz	30-03-2021	07-04-2021	
6	000000006	list	w drodze	Filip Piotr	SuperSklep Adam Kowal	30-03-2021	-	
7	000000003	list	oczekuje na odbior w placowce	Tomasz Kowal	Sportex sp.z.o.o	31-03-2021	-	

Ustaw status Zatwierdź

Wróć

Czego się nauczyłam

- Środowisko Qt, tworzenie interfejsu graficznego;
- Wykorzystanie w praktyce zagadnień z laboratorium – w szczególności kontenery, iteratory;
- Poznanie lepszego sposobu na walidację danych;
- Pliki .csv.

Wykorzystane zagadnienia z laboratorium

1. RTTI

```
if(typeid (*currentShipment).name() == typeid(Letter).name())
{
    localDatabase->addNewRecord(dynamic_cast<Letter*>(currentShipment));
    localDatabase->decrementLastID();
    mainDatabase->addNewRecord(dynamic_cast<Letter*>(currentShipment));
}
else if(typeid (*currentShipment).name() == typeid(Parcel).name())
{
    localDatabase->addNewRecord(dynamic_cast<Parcel*>(currentShipment));
    localDatabase->decrementLastID();
    mainDatabase->addNewRecord(dynamic_cast<Parcel*>(currentShipment));
}
```

2. i 3. Kontenery, Iteratory i Algorytmy STL

```
class ShipmentStatusManager
{
    std::map <int, ShipmentStatus*> statuses;
```

```
ShipmentStatus* ShipmentStatusManager::findStatus(int id)
{
    auto found = statuses.find(id);
    return found->second;
}

std::vector<std::string> ShipmentStatusManager::returnAvailableStatuses(ShipmentStatus& _status)
{
    auto vec = _status.getAvailableStatuses();
    std::vector<int>::iterator itVec = vec.begin();
    std::vector<std::string> result;
    do
    {
        result.push_back(findStatus(*itVec)->getStatus());
        itVec++;

    }  while(itVec!=vec.end());

    return result;
}
```

4. Szablony

```
template<typename T>
class Node
{
    Node* next, *prev;
    T data;

public:
    Node(T);
    Node(Node&);
    Node(T, Node<T>*, Node<T>*);
    void display();
    void setNext(Node*);
    Node * getNext();
    void setPrev(Node*);
    Node* getPrev();
    T& getCurrentData();
    Node* getCurrent();
    Node& operator=(const Node&);
    bool operator==(const std::string&);
    ~Node();
};
```

```
template<typename T>
class List
{
    Node<T>* head;
    unsigned counter;

public:
    List();
    Node<T>* addFront(Node<T>*);
    Node<T>* deleteElement(Node<T>*);
    void displayList();
    int size();
    Node<T>* getHead();
    Node <T>* getElement(std::string);
    Node <T>* getElement(T&);
    T pop();
    void setCounter(int);
    void increase_counter();
    T& getData(Node<T>*);
    ~List();
};
```

5. Wyrażenia regularne

```
class Validator
{
    std::map<dataInfo,std::string> patterns;
    std::map<std::string, std::string> postCodePatterns;
public:
    Validator();
    bool validate(const std::string,const dataInfo& dataType);
    bool validate(std::string, std::string);
    void readFileValidator();
    void readFileValidatorPostCode();
};
```

```
bool Validator::validate(const std::string _data, const dataInfo& dataType)
{
    if(!patterns.empty())
    {
    auto pattern = patterns.find(dataType);
    std::regex reg(pattern->second);
    std::smatch result;

    return std::regex_match(_data, result, reg);
    } else
        return true;
}
```

Dalsze prace

- Dodanie przesyłek zagranicznych;
- Dodanie wyszukiwania danych po numerach telefonu, numerze przesyłki, statusie;
- Dokończenie implementacji zmiany statusu;
- Drukowanie etykiety;
- Refaktoryzacja kodu.

Dziękuję za uwagę!