LNWeather

0.1.0

Wygenerowano za pomocą Doxygen 1.13.2

Rozdział 1

Indeks przestrzeni nazw

1.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie przestrzenie nazw wraz z ich krótkimi opisami:

Rozdział 2

Indeks hierarchiczny

2.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 3

Indeks klas

3.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików wraz z ich krótkimi opisami:

Rozdział 5

Dokumentacja przestrzeni nazw

5.1 Dokumentacja przestrzeni nazw Utils

Przestrzeń nazw zawierająca funkcje narzędziowe.

Funkcje

- wxDateTime parseDateTime (const std::string &dateStr)

  Konwertuje string daty na obiekt wxDateTime.

5.1.1 Opis szczegółowy

Przestrzeń nazw zawierająca funkcje narzędziowe.

5.1.2 Dokumentacja funkcji

5.1.2.1 parseDateTime()

```
wxDateTime Utils::parseDateTime (
            const std::string & dateStr)
```

Konwertuje string daty na obiekt wxDateTime.

Parsuje datę i czas z formatu API do wxDateTime.

Parametry

dateStr	String zawierający datę w formacie "YYYY-MM-DD HH:MM:SS"

Zwraca

wxDateTime Skonwertowany obiekt daty i czasu

Parametry

dateStr	String zawierający datę w formacie "YYYY-MM-DD HH:MM:SS"

Zwraca

wxDateTime Obiekt daty i czasu

Rozdział 6

Dokumentacja klas

6.1 Dokumentacja klasy ApiService

Klasa obsługująca komunikację z zewnętrznymi API.

`#include <ApiService.h>`

Statyczne metody publiczne

- static std::string fetchDataFromAPI (const std::string &url)

  Pobiera dane z określonego URL.
- static bool isNetworkAvailable ()

  Sprawdza czy jest dostęp do internetu.

6.1.1 Opis szczegółowy

Klasa obsługująca komunikację z zewnętrznymi API.

Zapewnia metody do pobierania danych z zewnętrznych źródeł oraz sprawdzania dostępności sieci.

6.1.2 Dokumentacja funkcji składowych

6.1.2.1 fetchDataFromAPI()

```
std::string ApiService::fetchDataFromAPI (
            const std::string & url)  [static]
```

Pobiera dane z określonego URL.

Pobiera dane z podanego URL API.

Parametry

url	Adres URL do pobrania danych

Zwraca

std::string Odpowiedź z API lub pusty string w przypadku błędu

Funkcja wykorzystuje libcurl do pobrania danych z zewnętrznego API. Jeśli nie ma dostępu do internetu, zwraca pusty string.

Parametry

url	Adres URL do pobrania danych

Zwraca

std::string Odpowiedź z API lub pusty string w przypadku błędu

6.1.2.2 isNetworkAvailable()

```
bool ApiService::isNetworkAvailable ()  [static]
```

Sprawdza czy jest dostęp do internetu.

Sprawdza dostępność połączenia z internetem.

Zwraca

true Jeśli połączenie z internetem jest dostępne

false Jeśli brak połączenia z internetem

Wykonuje proste zapytanie do API, aby sprawdzić czy urządzenie ma dostęp do internetu.

Zwraca

true Jeśli połączenie jest dostępne

false Jeśli brak połączenia z internetem

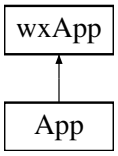
Dokumentacja dla tej klasy została wygenerowana z plików:

- src/ApiService.h
- src/ApiService.cpp

6.2 Dokumentacja klasy App

Główna klasa aplikacji wxWidgets.

Diagram dziedziczenia dla App



Metody publiczne

- virtual bool OnInit ()
  Metoda inicjalizacji aplikacji.

6.2.1 Opis szczegółowy

Główna klasa aplikacji wxWidgets.

Inicjalizuje i konfiguruje główne okno aplikacji.

6.2.2 Dokumentacja funkcji składowych

6.2.2.1 OnInit()

```
virtual bool App::OnInit ()  [inline], [virtual]
```

Metoda inicjalizacji aplikacji.

Tworzy główne okno aplikacji i ustawia jego ikonę.

Zwraca

true Jeśli inicjalizacja się powiodła

false Jeśli wystąpił błąd

Dokumentacja dla tej klasy została wygenerowana z pliku:

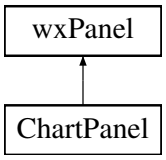
- src/main.cpp

6.3 Dokumentacja klasy ChartPanel

Panel wyświetlający wykres danych pomiarowych.

`#include <ChartPanel.h>`

Diagram dziedziczenia dla ChartPanel



Metody publiczne

- ChartPanel (wxWindow *parent)

  Konstruktor panelu wykresu.
- void SetData (const std::vector< MeasurementData > &data, const wxString &paramName)

  Ustawia dane do wyświetlenia na wykresie.
- void SetTitle (const wxString &title)

  Ustawia tytuł wykresu.
- const std::vector< MeasurementData > & GetChartData () const

  Zwraca referencję do aktualnych danych wykresu.

Metody prywatne

- void OnPaint (wxPaintEvent &event)

  Obsługa zdarzenia rysowania.
- void OnSize (wxSizeEvent &event)

  Obsługa zdarzenia zmiany rozmiaru.
- void DrawAxes (wxDC &dc, const wxRect &rect)

  Rysuje osie wykresu.
- void DrawData (wxDC &dc, const wxRect &rect)

  Rysuje dane na wykresie.
- void DrawLabels (wxDC &dc, const wxRect &rect)

  Rysuje etykiety na wykresie.

Atrybuty prywatne

- std::vector< MeasurementData > chartData
- wxString title
- wxString parameter

6.3.1 Opis szczegółowy

Panel wyświetlający wykres danych pomiarowych.

Wyświetla dane pomiarowe w postaci interaktywnego wykresu z możliwością zmiany tytułu i danych.

6.3.2 Dokumentacja konstruktora i destruktora

6.3.2.1 ChartPanel()

```
ChartPanel::ChartPanel (
            wxWindow * parent)
```

Konstruktor panelu wykresu.

Parametry

parent	Rodzic panelu

Inicjalizuje panel i konfiguruje obsluge zdarzen.

Parametry

parent	Rodzic panelu

6.3.3 Dokumentacja funkcji składowych

6.3.3.1 DrawAxes()

```
void ChartPanel::DrawAxes (
            wxDC & dc,
            const wxRect & rect)  [private]
```

Rysuje osie wykresu.

Parametry

dc	Kontekst urządzenia
rect	Prostokąt obszaru wykresu

6.3.3.2 DrawData()

```
void ChartPanel::DrawData (
            wxDC & dc,
            const wxRect & rect)  [private]
```

Rysuje dane na wykresie.

Parametry

dc	Kontekst urządzenia
rect	Prostokąt obszaru wykresu

Rysuje punkty i linie łączące dane pomiarowe.

Parametry

dc	Kontekst urządzenia
rect	Prostokąt obszaru wykresu

6.3.3.3 DrawLabels()

```
void ChartPanel::DrawLabels (
            wxDC & dc,
            const wxRect & rect)  [private]
```

Rysuje etykiety na wykresie.

Parametry

dc	Kontekst urządzenia
rect	Prostokąt obszaru wykresu

Rysuje etykiety osi X i Y z odpowiednimi wartościami.

Parametry

dc	Kontekst urządzenia
rect	Prostokąt obszaru wykresu

6.3.3.4 GetChartData()

```
const std::vector< MeasurementData > & ChartPanel::GetChartData () const  [inline]
```

Zwraca referencję do aktualnych danych wykresu.

Zwraca

const std::vector<MeasurementData>& Referencja do wektora danych

6.3.3.5 OnPaint()

```
void ChartPanel::OnPaint (
            wxPaintEvent & event)  [private]
```

Obsługa zdarzenia rysowania.

Parametry

event	Zdarzenie rysowania

Rysuje wykres z wykorzystaniem buforowania.

Parametry

event	Zdarzenie rysowania

6.3.3.6 OnSize()

```
void ChartPanel::OnSize (
            wxSizeEvent & event)  [private]
```

Obsługa zdarzenia zmiany rozmiaru.

Parametry

event	Zdarzenie zmiany rozmiaru

Odświeża wykres po zmianie rozmiaru panelu.

Parametry

event	Zdarzenie zmiany rozmiaru

6.3.3.7 SetData()

```
void ChartPanel::SetData (
            const std::vector< MeasurementData > & data,
            const wxString & paramName)
```

Ustawia dane do wyświetlenia na wykresie.

Parametry

data	Wektor danych pomiarowych
paramName	Nazwa parametru

6.3.3.8 SetTitle()

```
void ChartPanel::SetTitle (
            const wxString & newTitle)
```

Ustawia tytuł wykresu.

Parametry

title	Nowy tytuł wykresu
newTitle	Nowy tytuł wykresu

6.3.4 Dokumentacja atrybutów składowych

6.3.4.1 chartData

```
std::vector<MeasurementData> ChartPanel::chartData  [private]
```

Dane pomiarowe

6.3.4.2 parameter

```
wxString ChartPanel::parameter  [private]
```

Nazwa parametru

6.3.4.3 title

```
wxString ChartPanel::title  [private]
```

Tytuł wykresu

Dokumentacja dla tej klasy została wygenerowana z plików:

- src/ChartPanel.h
- src/ChartPanel.cpp

6.4 Dokumentacja klasy DatabaseService

Klasa obsługująca lokalną "bazę danych" (pliki JSON).

```
#include <DatabaseService.h>
```

Metody publiczne

- DatabaseService ()
  Konstruktor usługi bazy danych.
- bool initializeDatabase ()
  Inicjalizuje strukturę bazy danych.
- bool saveStationsData (const std::string &jsonData)
  Zapisuje dane stacji do bazy.
- bool saveSensorsData (int stationId, const std::string &jsonData)
  Zapisuje dane czujników dla stacji.
- bool saveMeasurementData (int sensorId, const std::string &jsonData)
  Zapisuje dane pomiarowe dla czujnika.
- bool saveAirQualityIndex (int stationId, const std::string &jsonData)
  Zapisuje indeks jakości powietrza dla stacji.
- std::string loadStationsData ()
  Wczytuje dane stacji z bazy.

- std::string loadSensorsData (int stationId)
  Wczytuje dane czujników dla stacji.
- std::string loadMeasurementData (int sensorId)
  Wczytuje dane pomiarowe dla czujnika.
- std::string loadAirQualityIndex (int stationId)
  Wczytuje indeks jakości powietrza dla stacji.
- bool hasStationsData ()
  Sprawdza czy dane stacji istnieją w bazie.
- bool hasSensorsData (int stationId)
  Sprawdza czy dane czujników dla stacji istnieją w bazie.
- bool hasMeasurementData (int sensorId)
  Sprawdza czy dane pomiarowe dla czujnika istnieją w bazie.
- bool hasAirQualityIndex (int stationId)
  Sprawdza czy indeks jakości powietrza dla stacji istnieje w bazie.
- wxDateTime getLastUpdateTime (const std::string &filePath)
  Pobiera czas ostatniej aktualizacji pliku.
- std::string getStationsFilePath ()
  Zwraca ścieżkę pliku z danymi stacji.
- std::string getSensorsFilePath (int stationId)
  Zwraca ścieżkę pliku z danymi czujników dla stacji.
- std::string getMeasurementFilePath (int sensorId)
  Zwraca ścieżkę pliku z danymi pomiarowymi dla czujnika.
- std::string getAirQualityFilePath (int stationId)
  Zwraca ścieżkę pliku z indeksem jakości powietrza dla stacji.

Metody prywatne

- bool saveToFile (const std::string &filePath, const std::string &data)
  Zapisuje dane do pliku.
- std::string loadFromFile (const std::string &filePath)
  Odczytuje dane z pliku.

Atrybuty prywatne

- std::string dbFolder

6.4.1 Opis szczegółowy

Klasa obsługująca lokalną "bazę danych" (pliki JSON).

Zapewnia funkcje do zapisywania, odczytywania i zarządzania danymi przechowywanymi lokalnie w plikach JSON.

6.4.2 Dokumentacja konstruktora i destruktora

6.4.2.1 DatabaseService()

```
DatabaseService::DatabaseService ()
```

Konstruktor usługi bazy danych.

Konstruktor klasy DatabaseService.

Inicjalizuje ścieżkę do folderu bazy danych i tworzy strukturę katalogów.

Inicjalizuje folder bazy danych i tworzy strukturę katalogów.

6.4.3 Dokumentacja funkcji składowych

6.4.3.1 getAirQualityFilePath()

```
std::string DatabaseService::getAirQualityFilePath (
            int stationId)
```

Zwraca ścieżkę pliku z indeksem jakości powietrza dla stacji.

Parametry

stationId	ID stacji

Zwraca

std::string Ścieżka do pliku

6.4.3.2 getLastUpdateTime()

```
wxDateTime DatabaseService::getLastUpdateTime (
            const std::string & filePath)
```

Pobiera czas ostatniej aktualizacji pliku.

Parametry

filePath	Ścieżka do pliku

Zwraca

wxDateTime Czas ostatniej modyfikacji

6.4.3.3 getMeasurementFilePath()

```
std::string DatabaseService::getMeasurementFilePath (
            int sensorId)
```

Zwraca ścieżkę pliku z danymi pomiarowymi dla czujnika.

Parametry

sensorId	ID czujnika

Zwraca

std::string Ścieżka do pliku

6.4.3.4 getSensorsFilePath()

```
std::string DatabaseService::getSensorsFilePath (
            int stationId)
```

Zwraca ścieżkę pliku z danymi czujników dla stacji.

Parametry

stationId	ID stacji

Zwraca

std::string Ścieżka do pliku

6.4.3.5 getStationsFilePath()

```
std::string DatabaseService::getStationsFilePath ()
```

Zwraca ścieżkę pliku z danymi stacji.

Zwraca

std::string Ścieżka do pliku

6.4.3.6 hasAirQualityIndex()

```
bool DatabaseService::hasAirQualityIndex (
            int stationId)
```

Sprawdza czy indeks jakości powietrza dla stacji istnieje w bazie.

Parametry

stationId	ID stacji

Zwraca

true Jeśli dane istnieją

false Jeśli dane nie istnieją

Parametry

stationId	ID stacji

Zwraca

true Jeśli plik z danymi istnieje

false Jeśli plik z danymi nie istnieje

6.4.3.7 hasMeasurementData()

```
bool DatabaseService::hasMeasurementData (
            int sensorId)
```

Sprawdza czy dane pomiarowe dla czujnika istnieją w bazie.

Parametry

sensor↩ Id	ID czujnika

Zwraca

true Jeśli dane istnieją

false Jeśli dane nie istnieją

Parametry

sensor↩ Id	ID czujnika

Zwraca

true Jeśli plik z danymi istnieje

false Jeśli plik z danymi nie istnieje

6.4.3.8 hasSensorsData()

```
bool DatabaseService::hasSensorsData (
            int stationId)
```

Sprawdza czy dane czujników dla stacji istnieją w bazie.

Parametry

station↩ Id	ID stacji

Zwraca

true Jeśli dane istnieją

false Jeśli dane nie istnieją

Parametry

station↩ Id	ID stacji

Zwraca

true Jeśli plik z danymi istnieje

false Jeśli plik z danymi nie istnieje

6.4.3.9 hasStationsData()

```
bool DatabaseService::hasStationsData ()
```

Sprawdza czy dane stacji istnieją w bazie.

Zwraca

true Jeśli dane istnieją
false Jeśli dane nie istnieją
true Jeśli plik z danymi istnieje
false Jeśli plik z danymi nie istnieje

6.4.3.10 initializeDatabase()

```
bool DatabaseService::initializeDatabase ()
```

Inicjalizuje strukturę bazy danych.

Tworzy folder dla bazy danych, jeśli nie istnieje.

Zwraca

true Jeśli inicjalizacja się powiodła
false Jeśli wystąpił błąd

Sprawdza czy folder bazy istnieje, jeśli nie - tworzy go.

Zwraca

true Jeśli folder istnieje lub został utworzony
false Jeśli utworzenie folderu nie powiodło się

6.4.3.11 loadAirQualityIndex()

```
std::string DatabaseService::loadAirQualityIndex (
            int stationId)
```

Wczytuje indeks jakości powietrza dla stacji.

Parametry

stationId	ID stacji

Zwraca

std::string Dane w formacie JSON

Parametry

stationId	ID stacji

Zwraca

std::string Dane w formacie JSON lub pusty string gdy brak danych

6.4.3.12 loadFromFile()

```
std::string DatabaseService::loadFromFile (
            const std::string & filePath)  [private]
```

Odczytuje dane z pliku.

Parametry

filePath	Ścieżka do pliku

Zwraca

std::string Odczytane dane

Parametry

filePath	Ścieżka do pliku

Zwraca

std::string Odczytane dane lub pusty string gdy błąd

6.4.3.13 loadMeasurementData()

```
std::string DatabaseService::loadMeasurementData (
            int sensorId)
```

Wczytuje dane pomiarowe dla czujnika.

Parametry

sensorId	ID czujnika

Zwraca

std::string Dane w formacie JSON

Parametry

sensorId	ID czujnika

Zwraca

std::string Dane w formacie JSON lub pusty string gdy brak danych

6.4.3.14 loadSensorsData()

```
std::string DatabaseService::loadSensorsData (
            int stationId)
```

Wczytuje dane czujników dla stacji.

Parametry

stationId	ID stacji

Zwraca

std::string Dane w formacie JSON

Parametry

stationId	ID stacji

Zwraca

std::string Dane w formacie JSON lub pusty string gdy brak danych

6.4.3.15 loadStationsData()

```
std::string DatabaseService::loadStationsData ()
```

Wczytuje dane stacji z bazy.

Zwraca

std::string Dane w formacie JSON

std::string Dane w formacie JSON lub pusty string gdy brak danych

6.4.3.16 saveAirQualityIndex()

```
bool DatabaseService::saveAirQualityIndex (
            int stationId,
            const std::string & jsonData)
```

Zapisuje indeks jakości powietrza dla stacji.

Parametry

stationId	ID stacji
jsonData	Dane w formacie JSON

Zwraca

true Jeśli zapis się powiódł

false Jeśli wystąpił błąd

6.4.3.17 saveMeasurementData()

```
bool DatabaseService::saveMeasurementData (
            int sensorId,
            const std::string & jsonData)
```

Zapisuje dane pomiarowe dla czujnika.

Parametry

sensorId	ID czujnika
jsonData	Dane w formacie JSON

Zwraca

true Jeśli zapis się powiódł

false Jeśli wystąpił błąd

6.4.3.18 saveSensorsData()

```
bool DatabaseService::saveSensorsData (
            int stationId,
            const std::string & jsonData)
```

Zapisuje dane czujników dla stacji.

Parametry

stationId	ID stacji
jsonData	Dane w formacie JSON

Zwraca

true Jeśli zapis się powiódł

false Jeśli wystąpił błąd

6.4.3.19 saveStationsData()

```
bool DatabaseService::saveStationsData (
            const std::string & jsonData)
```

Zapisuje dane stacji do bazy.

Zapisuje dane stacji do pliku.

Parametry

jsonData	Dane w formacie JSON

Zwraca

true Jeśli zapis się powiódł

false Jeśli wystąpił błąd

6.4.3.20 saveToFile()

```
bool DatabaseService::saveToFile (
            const std::string & filePath,
            const std::string & data)  [private]
```

Zapisuje dane do pliku.

Parametry

filePath	Ścieżka do pliku
data	Dane do zapisania

Zwraca

true Jeśli zapis się powiódł

false Jeśli wystąpił błąd

6.4.4 Dokumentacja atrybutów składowych

6.4.4.1 dbFolder

```
std::string DatabaseService::dbFolder  [private]
```

Ścieżka do folderu bazy danych

Dokumentacja dla tej klasy została wygenerowana z plików:

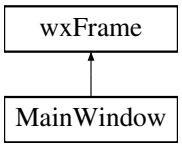
- src/DatabaseService.h
- src/DatabaseService.cpp

6.5 Dokumentacja klasy MainWindow

Klasa implementująca główne okno aplikacji.

```
#include <MainWindow.h>
```

Diagram dziedziczenia dla MainWindow



Metody publiczne

- MainWindow ()

  Konstruktor głównego okna.

Metody prywatne

- void loadStations ()

  Ładuje listę dostępnych stacji pomiarowych.
- void loadSensorsForStation (int stationId)

  Ładuje czujniki dla wybranej stacji.
- void loadAirQualityIndex (int stationId)

  Ładuje indeks jakości powietrza dla stacji.
- wxDateTime parseDateTime (const std::string &dateStr)

  Konwertuje string daty na obiekt wxDateTime.
- void OnStationSelection (wxCommandEvent &event)

  Obsługa zdarzenia wyboru stacji.
- void OnSensorSelection (wxCommandEvent &event)

  Obsługa zdarzenia wyboru czujnika.
- void OnSaveCurrentToDatabase (wxCommandEvent &event)

  Obsługa zdarzenia zapisania bieżącego pomiaru do bazy danych.
- void OnSaveAllToDatabase (wxCommandEvent &event)

  Obsługa zdarzenia zapisania wszystkich danych do bazy danych.
- void OnApplyDateRange (wxCommandEvent &event)

  Obsługa zdarzenia zastosowania zakresu dat dla wykresu.
- bool tryLoadStationsFromAPI ()

  Próbuje załadować stacje z API.
- bool tryLoadStationsFromDatabase ()

  Próbuje załadować stacje z lokalnej bazy danych.
- bool tryLoadSensorsFromAPI (int stationId)

  Próbuje załadować czujniki dla stacji z API.
- bool tryLoadSensorsFromDatabase (int stationId)

  Próbuje załadować czujniki dla stacji z lokalnej bazy danych.
- bool tryLoadMeasurementsFromAPI (int sensorId)

  Próbuje załadować pomiary z API.
- bool tryLoadMeasurementsFromDatabase (int sensorId)

  Próbuje załadować pomiary z lokalnej bazy danych.
- bool tryLoadAirQualityFromAPI (int stationId)

  Próbuje załadować indeks jakości powietrza z API.
- bool tryLoadAirQualityFromDatabase (int stationId)

  Próbuje załadować indeks jakości powietrza z lokalnej bazy danych.
- bool tryLoadMeasurementsWithDateRange (int sensorId)

  Próbuje załadować pomiary z określonego zakresu dat.
- bool tryLoadMeasurementsWithDateRangeFromDatabase (int sensorId)

  Próbuje załadować pomiary z określonego zakresu dat z lokalnej bazy danych.
- void OnShowAnalysis (wxCommandEvent &event)

  Obsługa zdarzenia pokazania analizy danych.
- void PerformDataAnalysis (const std::vector< MeasurementData > &data, const wxString &paramName)

  Wykonuje analizę danych pomiarowych.
- void updateDatabaseStatus ()

  Aktualizuje wyświetlany status bazy danych.

Atrybuty prywatne

- wxChoice * stationChoice

  Lista rozwijana do wyboru stacji pomiarowej.
- wxChoice * sensorChoice

  Lista rozwijana do wyboru czujnika.
- wxTextCtrl * measurementText

  Pole tekstowe do wyświetlania danych pomiarowych.
- wxFont font

  Czcionka używana w interfejsie.
- wxStaticText * airQualityIndex

  Etykieta wyświetlająca indeks jakości powietrza.
- wxButton * saveCurrentToDbButton

  Przycisk do zapisania bieżącego pomiaru do bazy danych.
- wxButton * saveAllToDbButton

  Przycisk do zapisania wszystkich danych do bazy danych.
- wxStaticText * dbStatusText

  Etykieta wyświetlająca status bazy danych.
- wxButton * showAnalysisButton

  Przycisk do pokazania analizy danych.
- wxTextCtrl * analysisText

  Pole tekstowe do wyświetlania analizy danych.
- wxNotebook * dataNotebook

  Kontrolka z zakładkami dla różnych widoków danych.
- ChartPanel * chartPanel

  Panel wykresu do wizualizacji danych pomiarowych.
- wxDatePickerCtrl * startDatePicker

  Kontrolka do wyboru daty początkowej zakresu.
- wxDatePickerCtrl * endDatePicker

  Kontrolka do wyboru daty końcowej zakresu.
- wxButton * applyDateRangeButton

  Przycisk do zastosowania wybranego zakresu dat.
- std::vector< std::pair< int, std::string > > stations

  Wektor przechowujący pary (ID, nazwa) dostępnych stacji.
- std::vector< std::pair< int, std::string > > sensors

  Wektor przechowujący pary (ID, nazwa) dostępnych czujników.
- DatabaseService dbService

  Usługa do obsługi lokalnej bazy danych.

6.5.1 Opis szczegółowy

Klasa implementująca główne okno aplikacji.

Zarządza interfejsem użytkownika, pobieraniem i wyświetlaniem danych z API jakości powietrza oraz lokalnej bazy danych.

6.5.2 Dokumentacja konstruktora i destruktora

6.5.2.1 MainWindow()

```
MainWindow::MainWindow ()
```

Konstruktor głównego okna.

Konstruktor klasy MainWindow.

Inicjalizuje interfejs aplikacji i ładuje dane stacji.

Inicjalizuje główne okno aplikacji, tworzy wszystkie kontrolki interfejsu użytkownika, ustawia układ elementów, wiąże zdarzenia oraz inicjalizuje bazę danych.

6.5.3 Dokumentacja funkcji składowych

6.5.3.1 loadAirQualityIndex()

```
void MainWindow::loadAirQualityIndex (
            int stationId)  [private]
```

Ładuje indeks jakości powietrza dla stacji.

Ładuje informacje o indeksie jakości powietrza dla wybranej stacji.

Parametry

stationId	ID stacji, dla której należy załadować indeks jakości powietrza
stationId	Identyfikator wybranej stacji

Pobiera indeks jakości powietrza z API lub z lokalnej bazy danych. Aktualizuje etykietę indeksu jakości powietrza z odpowiednim kolorem.

6.5.3.2 loadSensorsForStation()

```
void MainWindow::loadSensorsForStation (
            int stationId)  [private]
```

Ładuje czujniki dla wybranej stacji.

Ładuje informacje o czujnikach dla wybranej stacji.

Parametry

stationId	ID stacji, dla której należy załadować czujniki
stationId	Identyfikator wybranej stacji

Pobiera dane o czujnikach dla stacji z API lub z lokalnej bazy danych. Wypełnia kontrolkę wyboru czujników.

6.5.3.3 loadStations()

```
void MainWindow::loadStations ()  [private]
```

Ładuje listę dostępnych stacji pomiarowych.

Ładuje informacje o dostępnych stacjach pomiarowych.

Próbuje pobrać dane o stacjach z API GIOŚ, a jeśli to się nie powiedzie, próbuje załadować dane z lokalnej bazy. Wypełnia kontrolkę wyboru stacji.

6.5.3.4 OnApplyDateRange()

```
void MainWindow::OnApplyDateRange (
            wxCommandEvent & event)  [private]
```

Obsługa zdarzenia zastosowania zakresu dat dla wykresu.

Obsługuje zdarzenie zastosowania wybranego zakresu dat.

Parametry

event	Zdarzenie przycisku
event	Obiekt zdarzenia

Filtruje dane pomiarowe aktualnie wybranego czujnika według wybranego zakresu dat i aktualizuje wykres oraz tekstowy format danych.

6.5.3.5 OnSaveAllToDatabase()

```
void MainWindow::OnSaveAllToDatabase (
            wxCommandEvent & event)  [private]
```

Obsługa zdarzenia zapisania wszystkich danych do bazy danych.

Obsługuje zdarzenie zapisywania danych wszystkich stacji do lokalnej bazy.

Parametry

event	Zdarzenie przycisku
event	Obiekt zdarzenia

Pobiera dane o wszystkich stacjach, ich czujnikach, indeksach jakości powietrza oraz dane pomiarowe i zapisuje je do lokalnej bazy danych. Wyświetla pasek postępu informujący o statusie pobierania.

6.5.3.6 OnSaveCurrentToDatabase()

```
void MainWindow::OnSaveCurrentToDatabase (
            wxCommandEvent & event)  [private]
```

Obsługa zdarzenia zapisania bieżącego pomiaru do bazy danych.

Obsługuje zdarzenie zapisywania danych bieżącej stacji do lokalnej bazy.

Parametry

event	Zdarzenie przycisku
event	Obiekt zdarzenia

Pobiera dane czujników, indeksu jakości powietrza oraz dane pomiarowe dla aktualnie wybranej stacji i zapisuje je do lokalnej bazy danych. Wyświetla pasek postępu informujący o statusie pobierania.

6.5.3.7 OnSensorSelection()

```
void MainWindow::OnSensorSelection (
            wxCommandEvent & event)  [private]
```

Obsługa zdarzenia wyboru czujnika.

Obsługuje zdarzenie wyboru czujnika.

Parametry

event	Zdarzenie wyboru
event	Obiekt zdarzenia

Reaguje na wybór czujnika przez użytkownika, ładując dane pomiarowe dla wybranego czujnika i wyświetlając je na wykresie.

6.5.3.8 OnShowAnalysis()

```
void MainWindow::OnShowAnalysis (
            wxCommandEvent & event)  [private]
```

Obsługa zdarzenia pokazania analizy danych.

Obsługuje zdarzenie pokazania analizy danych.

Parametry

event	Zdarzenie przycisku
event	Obiekt zdarzenia

Inicjuje analizę danych aktualnie wybranego czujnika i przełącza na zakładkę wyświetlającą wyniki analizy.

6.5.3.9 OnStationSelection()

```
void MainWindow::OnStationSelection (
            wxCommandEvent & event)  [private]
```

Obsługa zdarzenia wyboru stacji.

Obsługuje zdarzenie wyboru stacji.

Parametry

event	Zdarzenie wyboru
event	Obiekt zdarzenia

Reaguje na wybór stacji przez użytkownika, ładując czujniki i indeks jakości powietrza dla wybranej stacji.

6.5.3.10 parseDateTime()

```
wxDateTime MainWindow::parseDateTime (
            const std::string & dateStr)  [private]
```

Konwertuje string daty na obiekt wxDateTime.

Parsuje string z datą na obiekt wxDateTime.

Parametry

dateStr	String zawierający datę w odpowiednim formacie

Zwraca

wxDateTime Skonwertowany obiekt daty i czasu

Parametry

dateStr	String zawierający datę w formacie "YYYY-MM-DD HH:MM:SS"

Zwraca

Obiekt wxDateTime reprezentujący datę z wejściowego stringa

6.5.3.11 PerformDataAnalysis()

```
void MainWindow::PerformDataAnalysis (
            const std::vector< MeasurementData > & data,
            const wxString & paramName)  [private]
```

Wykonuje analizę danych pomiarowych.

Analizuje dane, obliczając statystyki i przygotowując raport.

Parametry

data	Wektor danych pomiarowych do analizy
paramName	Nazwa parametru, dla którego wykonywana jest analiza
data	Wektor z danymi pomiarowymi do analizy
paramName	Nazwa parametru, którego dotyczą dane

Przeprowadza analizę danych pomiarowych obejmującą:

- Znalezienie wartości minimalnej i maksymalnej wraz z datami
- Obliczenie wartości średniej
- Analizę trendu zmian wartości
- Obliczenie zakresu zmian Wyniki analizy wyświetla w formie tekstowej.

6.5.3.12 tryLoadAirQualityFromAPI()

```
bool MainWindow::tryLoadAirQualityFromAPI (
            int stationId)  [private]
```

Próbuje załadować indeks jakości powietrza z API.

Próbuje załadować indeks jakości powietrza dla stacji z API GIOŚ

Parametry

stationId	ID stacji

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

stationId	Identyfikator stacji

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Pobiera indeks jakości powietrza z API, przetwarza go i aktualizuje odpowiednią etykietę. Zapisuje również pobrane dane do lokalnej bazy.

6.5.3.13 tryLoadAirQualityFromDatabase()

```
bool MainWindow::tryLoadAirQualityFromDatabase (
            int stationId)  [private]
```

Próbuje załadować indeks jakości powietrza z lokalnej bazy danych.

Próbuje załadować indeks jakości powietrza dla stacji z lokalnej bazy danych.

Parametry

stationId	ID stacji

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

stationId	Identyfikator stacji

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Odczytuje indeks jakości powietrza z lokalnej bazy, przetwarza go i aktualizuje odpowiednią etykietę.

6.5.3.14 tryLoadMeasurementsFromAPI()

```
bool MainWindow::tryLoadMeasurementsFromAPI (
            int sensorId)  [private]
```

Próbuje załadować pomiary z API.

Próbuje załadować dane pomiarowe dla czujnika z API GIOŚ

Parametry

sensorId	ID czujnika

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

sensorId	Identyfikator czujnika

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Pobiera dane pomiarowe z API, przetwarza je i wyświetla na wykresie oraz w formacie tekstowym. Zapisuje również pobrane dane do lokalnej bazy.

6.5.3.15 tryLoadMeasurementsFromDatabase()

```
bool MainWindow::tryLoadMeasurementsFromDatabase (
            int sensorId)  [private]
```

Próbuje załadować pomiary z lokalnej bazy danych.

Próbuje załadować dane pomiarowe dla czujnika z lokalnej bazy danych.

Parametry

sensorId	ID czujnika

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

sensorId	Identyfikator czujnika

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Odczytuje dane pomiarowe z lokalnej bazy, przetwarza je i wyświetla na wykresie oraz w formacie tekstowym.

6.5.3.16 tryLoadMeasurementsWithDateRange()

```
bool MainWindow::tryLoadMeasurementsWithDateRange (
            int sensorId)  [private]
```

Próbuje załadować pomiary z określonego zakresu dat.

Próbuje załadować dane pomiarowe dla wybranego zakresu dat z API GIOŚ

Parametry

sensorId	ID czujnika

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

sensorId	Identyfikator czujnika

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Pobiera dane pomiarowe z API, filtruje je według wybranego zakresu dat, a następnie aktualizuje wykres oraz tekstowy format danych.

6.5.3.17 tryLoadMeasurementsWithDateRangeFromDatabase()

```
bool MainWindow::tryLoadMeasurementsWithDateRangeFromDatabase (
            int sensorId)  [private]
```

Próbuje załadować pomiary z określonego zakresu dat z lokalnej bazy danych.

Próbuje załadować dane pomiarowe dla wybranego zakresu dat z lokalnej bazy danych.

Parametry

sensorId	ID czujnika

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

sensorId	Identyfikator czujnika

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Odczytuje dane pomiarowe z lokalnej bazy, filtruje je według wybranego zakresu dat, a następnie aktualizuje wykres oraz tekstowy format danych.

6.5.3.18 tryLoadSensorsFromAPI()

```
bool MainWindow::tryLoadSensorsFromAPI (
            int stationId)  [private]
```

Próbuje załadować czujniki dla stacji z API.

Próbuje załadować informacje o czujnikach dla stacji z API GIOŚ

Parametry

stationId	ID stacji

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

stationId	Identyfikator stacji

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Pobiera dane o czujnikach z API, przetwarza je i wypełnia kontrolkę wyboru czujników. Zapisuje również pobrane dane do lokalnej bazy.

6.5.3.19 tryLoadSensorsFromDatabase()

```
bool MainWindow::tryLoadSensorsFromDatabase (
            int stationId)  [private]
```

Próbuje załadować czujniki dla stacji z lokalnej bazy danych.

Próbuje załadować informacje o czujnikach dla stacji z lokalnej bazy danych.

Parametry

stationId	ID stacji

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

Parametry

stationId	Identyfikator stacji

Zwraca

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Odczytuje dane o czujnikach z lokalnej bazy, przetwarza je i wypełnia kontrolkę wyboru czujników.

6.5.3.20 tryLoadStationsFromAPI()

```
bool MainWindow::tryLoadStationsFromAPI ()  [private]
```

Próbuje załadować stacje z API.

Próbuje załadować informacje o stacjach z API GIOŚ

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Pobiera dane o stacjach z API, przetwarza je i wypełnia kontrolkę wyboru stacji. Zapisuje również pobrane dane do lokalnej bazy.

6.5.3.21 tryLoadStationsFromDatabase()

```
bool MainWindow::tryLoadStationsFromDatabase ()  [private]
```

Próbuje załadować stacje z lokalnej bazy danych.

Próbuje załadować informacje o stacjach z lokalnej bazy danych.

Zwraca

true Jeśli załadowanie się powiodło

false Jeśli wystąpił błąd

true jeśli operacja zakończyła się sukcesem, false w przeciwnym przypadku

Odczytuje dane o stacjach z lokalnej bazy, przetwarza je i wypełnia kontrolkę wyboru stacji.

6.5.3.22 updateDatabaseStatus()

```
void MainWindow::updateDatabaseStatus ()  [private]
```

Aktualizuje wyświetlany status bazy danych.

Aktualizuje informacje o statusie bazy danych lokalnych.

Sprawdza czy lokalna baza danych zawiera informacje o stacjach, a następnie aktualizuje tekst statusu wraz z odpowiednim kolorem.

6.5.4 Dokumentacja atrybutów składowych

6.5.4.1 airQualityIndex

```
wxStaticText* MainWindow::airQualityIndex  [private]
```

Etykieta wyświetlająca indeks jakości powietrza.

6.5.4.2 analysisText

```
wxTextCtrl* MainWindow::analysisText  [private]
```

Pole tekstowe do wyświetlania analizy danych.

6.5.4.3 applyDateRangeButton

```
wxButton* MainWindow::applyDateRangeButton  [private]
```

Przycisk do zastosowania wybranego zakresu dat.

6.5.4.4 chartPanel

```
ChartPanel* MainWindow::chartPanel  [private]
```

Panel wykresu do wizualizacji danych pomiarowych.

6.5.4.5 dataNotebook

```
wxNotebook* MainWindow::dataNotebook  [private]
```

Kontrolka z zakładkami dla różnych widoków danych.

6.5.4.6 dbService

```
DatabaseService MainWindow::dbService  [private]
```

Usługa do obsługi lokalnej bazy danych.

6.5.4.7 dbStatusText

```
wxStaticText* MainWindow::dbStatusText  [private]
```

Etykieta wyświetlająca status bazy danych.

6.5.4.8 endDatePicker

```
wxDatePickerCtrl* MainWindow::endDatePicker  [private]
```

Kontrolka do wyboru daty końcowej zakresu.

6.5.4.9 font

```
wxFont MainWindow::font  [private]
```

Czcionka używana w interfejsie.

6.5.4.10 measurementText

```
wxTextCtrl* MainWindow::measurementText  [private]
```

Pole tekstowe do wyświetlania danych pomiarowych.

6.5.4.11 saveAllToDbButton

```
wxButton* MainWindow::saveAllToDbButton  [private]
```

Przycisk do zapisania wszystkich danych do bazy danych.

6.5.4.12 saveCurrentToDbButton

```
wxButton* MainWindow::saveCurrentToDbButton  [private]
```

Przycisk do zapisania bieżącego pomiaru do bazy danych.

6.5.4.13 sensorChoice

```
wxChoice* MainWindow::sensorChoice  [private]
```

Lista rozwijana do wyboru czujnika.

6.5.4.14 sensors

```
std::vector<std::pair<int, std::string> > MainWindow::sensors  [private]
```

Wektor przechowujący pary (ID, nazwa) dostępnych czujników.

6.5.4.15 showAnalysisButton

```
wxButton* MainWindow::showAnalysisButton  [private]
```

Przycisk do pokazania analizy danych.

6.5.4.16 startDatePicker

```
wxDatePickerCtrl* MainWindow::startDatePicker  [private]
```

Kontrolka do wyboru daty początkowej zakresu.

6.5.4.17 stationChoice

```
wxChoice* MainWindow::stationChoice  [private]
```

Lista rozwijana do wyboru stacji pomiarowej.

6.5.4.18 stations

```
std::vector<std::pair<int, std::string> > MainWindow::stations  [private]
```

Wektor przechowujący pary (ID, nazwa) dostępnych stacji.

Dokumentacja dla tej klasy została wygenerowana z plików:

- src/MainWindow.h
- src/MainWindow.cpp

6.6 Dokumentacja struktury MeasurementData

Struktura do przechowywania danych pomiarowych dla wykresu.

`#include <ChartPanel.h>`

Atrybuty publiczne

- wxDateTime date
- double value
- bool hasValue

6.6.1 Opis szczegółowy

Struktura do przechowywania danych pomiarowych dla wykresu.

Pomocnicza struktura do przechowywania danych pomiarowych.

Przechowuje wartość pojedynczego pomiaru wraz z datą oraz flagą określającą czy wartość jest poprawna.

6.6.2 Dokumentacja atrybutów składowych

6.6.2.1 date

`wxDateTime MeasurementData::date`

Data i czas pomiaru

6.6.2.2 hasValue

`bool MeasurementData::hasValue`

Flaga określająca czy wartość jest dostępna

6.6.2.3 value

`double MeasurementData::value`

Wartość pomiaru

Dokumentacja dla tej struktury została wygenerowana z plików:

- src/ChartPanel.h
- src/Models.h

Rozdział 7

Dokumentacja plików

7.1 Dokumentacja pliku src/ApiService.cpp

Implementacja usługi komunikacji z API.

```
#include "ApiService.h"
#include "Utils.h"
#include <curl/curl.h>
#include <wx/wx.h>
#include <fstream>
```

7.1.1 Opis szczegółowy

Implementacja usługi komunikacji z API.

7.2 Dokumentacja pliku src/ApiService.h

Usługa do komunikacji z zewnętrznym API.

```
#include <string>
```

Komponenty

- class ApiService

  Klasa obsługująca komunikację z zewnętrznymi API.

7.2.1 Opis szczegółowy

Usługa do komunikacji z zewnętrznym API.

7.3 ApiService.h

Idź do dokumentacji tego pliku.

```
00001
00005
00006  #pragma once
00007  #include <string>
00008
00016  class ApiService {
00017  public:
00024      static std::string fetchDataFromAPI(const std::string& url);
00025
00032      static bool isNetworkAvailable();
00033  };
```

7.4 Dokumentacja pliku src/ChartPanel.cpp

Implementacja panelu wykresow danych pomiarowych.

```
#include "ChartPanel.h"
#include <wx/dcbuffer.h>
#include <algorithm>
```

7.4.1 Opis szczegółowy

Implementacja panelu wykresow danych pomiarowych.

7.5 Dokumentacja pliku src/ChartPanel.h

Panel wykresu do wizualizacji danych pomiarowych.

```
#include <wx/wx.h>
#include <wx/datetime.h>
#include <vector>
```

Komponenty

- struct MeasurementData

  Struktura do przechowywania danych pomiarowych dla wykresu.
- class ChartPanel

  Panel wyświetlający wykres danych pomiarowych.

7.5.1 Opis szczegółowy

Panel wykresu do wizualizacji danych pomiarowych.

7.6 ChartPanel.h

Idź do dokumentacji tego pliku.

```
00001
00005
00006 #pragma once
00007
00008 #include <wx/wx.h>
00009 #include <wx/datetime.h>
00010 #include <vector>
00011
00016 struct MeasurementData {
00017     wxDateTime date;
00018     double value;
00019     bool hasValue;
00020 };
00021
00029 class ChartPanel : public wxPanel {
00030 public:
00036     ChartPanel(wxWindow* parent);
00037
00044     void SetData(const std::vector<MeasurementData>& data, const wxString& paramName);
00045
00051     void SetTitle(const wxString& title);
00052
00058     const std::vector<MeasurementData>& GetChartData() const { return chartData; }
00059
00060 private:
00061     std::vector<MeasurementData> chartData;
00062
00063     wxString title;
00064     wxString parameter;
00065
00071     void OnPaint(wxPaintEvent& event);
00072
00078     void OnSize(wxSizeEvent& event);
00079
00086     void DrawAxes(wxDC& dc, const wxRect& rect);
00087
00094     void DrawData(wxDC& dc, const wxRect& rect);
00095
00102     void DrawLabels(wxDC& dc, const wxRect& rect);
00103 };
```

7.7 Dokumentacja pliku src/DatabaseService.cpp

Implementacja usługi bazy danych.

```
#include "DatabaseService.h"
#include <wx/dir.h>
#include <wx/file.h>
#include <wx/filefn.h>
#include <wx/datetime.h>
#include <wx/filename.h>
#include <fstream>
#include <iostream>
```

7.7.1 Opis szczegółowy

Implementacja usługi bazy danych.

7.8 Dokumentacja pliku src/DatabaseService.h

Usługa zarządzająca lokalną bazą danych.

```
#include <string>
#include <vector>
#include <wx/wx.h>
#include <json.hpp>
#include "Utils.h"
```

Komponenty

- class DatabaseService

  Klasa obsługująca lokalną "bazę danych" (pliki JSON).

7.8.1 Opis szczegółowy

Usługa zarządzająca lokalną bazą danych.

7.9 DatabaseService.h

Idź do dokumentacji tego pliku.

```
00001
00005
00006 #pragma once
00007
00008 #include <string>
00009 #include <vector>
00010 #include <wx/wx.h>
00011 #include <json.hpp>
00012 #include "Utils.h"
00013
00021 class DatabaseService {
00022 public:
00029     DatabaseService();
00030
00039     bool initializeDatabase();
00040
00048     bool saveStationsData(const std::string& jsonData);
00049
00058     bool saveSensorsData(int stationId, const std::string& jsonData);
00059
00068     bool saveMeasurementData(int sensorId, const std::string& jsonData);
00069
00078     bool saveAirQualityIndex(int stationId, const std::string& jsonData);
00079
00085     std::string loadStationsData();
00086
00093     std::string loadSensorsData(int stationId);
00094
00101     std::string loadMeasurementData(int sensorId);
00102
00109     std::string loadAirQualityIndex(int stationId);
00110
00117     bool hasStationsData();
00118
00126     bool hasSensorsData(int stationId);
00127
00135     bool hasMeasurementData(int sensorId);
00136
00144     bool hasAirQualityIndex(int stationId);
00145
00152     wxDateTime getLastUpdateTime(const std::string& filePath);
00153
```

```
00159     std::string getStationsFilePath();
00160
00167     std::string getSensorsFilePath(int stationId);
00168
00175     std::string getMeasurementFilePath(int sensorId);
00176
00183     std::string getAirQualityFilePath(int stationId);
00184
00185 private:
00186     std::string dbFolder;
00187
00196     bool saveToFile(const std::string& filePath, const std::string& data);
00197
00204     std::string loadFromFile(const std::string& filePath);
00205 };
```

7.10 Dokumentacja pliku src/main.cpp

Punkt wejściowy aplikacji.

```
#include <wx/wx.h>
#include "MainWindow.h"
```

Komponenty

- class App

  Główna klasa aplikacji wxWidgets.

Funkcje

- wxIMPLEMENT_APP (App)

7.10.1 Opis szczegółowy

Punkt wejściowy aplikacji.

7.10.2 Dokumentacja funkcji

7.10.2.1 wxIMPLEMENT_APP()

```
wxIMPLEMENT_APP (
            App )
```

7.11 Dokumentacja pliku src/MainWindow.cpp

Implementacja głównego okna aplikacji LNWeather.

```
#include "MainWindow.h"
#include "ApiService.h"
#include "Utils.h"
#include <json.hpp>
#include <fstream>
#include <algorithm>
```

Definicje typów

- using json = nlohmann::json

Funkcje

- bool compareStationsByName (const std::pair< int, std::string > &a, const std::pair< int, std::string > &b)

  Porównuje dwie stacje na podstawie ich nazw.

7.11.1 Opis szczegółowy

Implementacja głównego okna aplikacji LNWeather.

Plik zawiera implementację klasy MainWindow, która odpowiada za interfejs graficzny i funkcjonalność głównego okna aplikacji LNWeather. Aplikacja umożliwia pobieranie, wyświetlanie i analizę danych z API GIOŚ dotyczących jakości powietrza.

7.11.2 Dokumentacja definicji typów

7.11.2.1 json

```
using json = nlohmann::json
```

7.11.3 Dokumentacja funkcji

7.11.3.1 compareStationsByName()

```
bool compareStationsByName (
            const std::pair< int, std::string > & a,
            const std::pair< int, std::string > & b)
```

Porównuje dwie stacje na podstawie ich nazw.

Funkcja pomocnicza używana do sortowania stacji w kolejności alfabetycznej.

Parametry

a	Pierwsza para (id, nazwa) stacji do porównania
b	Druga para (id, nazwa) stacji do porównania

Zwraca

true jeśli nazwa stacji a powinna być przed nazwą stacji b, false w przeciwnym wypadku

7.12 Dokumentacja pliku src/MainWindow.h

Główne okno aplikacji do monitorowania jakości powietrza.

```
#include <wx/wx.h>
#include <wx/choice.h>
#include <wx/progdlg.h>
#include <wx/datectrl.h>
#include <wx/dateevt.h>
#include <wx/notebook.h>
#include <vector>
#include <utility>
#include "ChartPanel.h"
#include "DatabaseService.h"
```

Komponenty

- class MainWindow

  Klasa implementująca główne okno aplikacji.

7.12.1 Opis szczegółowy

Główne okno aplikacji do monitorowania jakości powietrza.

7.13 MainWindow.h

Idź do dokumentacji tego pliku.

```
00001
00005
00006 #pragma once
00007 #include <wx/wx.h>
00008 #include <wx/choice.h>
00009 #include <wx/progdlg.h>
00010 #include <wx/datectrl.h>
00011 #include <wx/dateevt.h>
00012 #include <wx/notebook.h>
00013 #include <vector>
00014 #include <utility>
00015 #include "ChartPanel.h"
00016 #include "DatabaseService.h"
00017
00025 class MainWindow : public wxFrame {
00026 public:
00032     MainWindow();
00033
00034 private:
00035
00039     wxChoice* stationChoice;
00040
00044     wxChoice* sensorChoice;
00045
00049     wxTextCtrl* measurementText;
00050
00054     wxFont font;
00055
00059     wxStaticText* airQualityIndex;
00060
00064     wxButton* saveCurrentToDbButton;
00065
00069     wxButton* saveAllToDbButton;
00070
00074     wxStaticText* dbStatusText;
```

```
00075
00076     //Analiza danych
00080     wxButton* showAnalysisButton;
00081
00085     wxTextCtrl* analysisText;
00086
00090     wxNotebook* dataNotebook;
00091
00092     //Chart Stuff
00096     ChartPanel* chartPanel;
00097
00101     wxDatePickerCtrl* startDatePicker;
00102
00106     wxDatePickerCtrl* endDatePicker;
00107
00111     wxButton* applyDateRangeButton;
00112
00113     // Data Storage
00117     std::vector<std::pair<int, std::string>> stations;
00118
00122     std::vector<std::pair<int, std::string>> sensors;
00123
00127     DatabaseService dbService;
00128
00129
00133     void loadStations();
00134
00140     void loadSensorsForStation(int stationId);
00141
00147     void loadAirQualityIndex(int stationId);
00148
00155     wxDateTime parseDateTime(const std::string& dateStr);
00156
00157
00163     void OnStationSelection(wxCommandEvent& event);
00164
00170     void OnSensorSelection(wxCommandEvent& event);
00171
00177     void OnSaveCurrentToDatabase(wxCommandEvent& event);
00178
00184     void OnSaveAllToDatabase(wxCommandEvent& event);
00185
00191     void OnApplyDateRange(wxCommandEvent& event);
00192
00193     // Metody
00200     bool tryLoadStationsFromAPI();
00201
00208     bool tryLoadStationsFromDatabase();
00209
00217     bool tryLoadSensorsFromAPI(int stationId);
00218
00226     bool tryLoadSensorsFromDatabase(int stationId);
00227
00235     bool tryLoadMeasurementsFromAPI(int sensorId);
00236
00244     bool tryLoadMeasurementsFromDatabase(int sensorId);
00245
00253     bool tryLoadAirQualityFromAPI(int stationId);
00254
00262     bool tryLoadAirQualityFromDatabase(int stationId);
00263
00271     bool tryLoadMeasurementsWithDateRange(int sensorId);
00272
00280     bool tryLoadMeasurementsWithDateRangeFromDatabase(int sensorId);
00281
00282
00288     void OnShowAnalysis(wxCommandEvent& event);
00289
00298     void PerformDataAnalysis(const std::vector<MeasurementData>& data, const wxString& paramName);
00299
00300
00304     void updateDatabaseStatus();
00305 };
```

7.14 Dokumentacja pliku src/Models.h

Definicje struktur danych używanych w aplikacji.

```
#include <wx/datetime.h>
```

Komponenty

- struct MeasurementData

  Struktura do przechowywania danych pomiarowych dla wykresu.

7.14.1 Opis szczegółowy

Definicje struktur danych używanych w aplikacji.

7.15 Models.h

Idź do dokumentacji tego pliku.

```
00001
00005
00006 #pragma once
00007 #include <wx/datetime.h>
00008
00016 struct MeasurementData {
00017     wxDateTime date;
00018     double value;
00019     bool hasValue;
00020 };
```

7.16 Dokumentacja pliku src/resource.h

Definicje

- #define IDI_ICON1 102

7.16.1 Dokumentacja definicji

7.16.1.1 IDI_ICON1

```
#define IDI_ICON1 102
```

7.17 resource.h

Idź do dokumentacji tego pliku.

```
00001 //{{NO_DEPENDENCIES}}
00002 // Microsoft Visual C++ generated include file.
00003 // Used by resource.rc
00004 //
00005 #define IDI_ICON1                       102
00006
00007 // Next default values for new objects
00008 //
00009 #ifdef APSTUDIO_INVOKED
00010 #ifndef APSTUDIO_READONLY_SYMBOLS
00011 #define _APS_NEXT_RESOURCE_VALUE        103
00012 #define _APS_NEXT_COMMAND_VALUE         40001
00013 #define _APS_NEXT_CONTROL_VALUE         1001
00014 #define _APS_NEXT_SYMED_VALUE           101
00015 #endif
00016 #endif
```

7.18 Dokumentacja pliku src/Utils.cpp

Implementacja narzędzi pomocniczych.

```
#include "Utils.h"
#include <wx/wx.h>
```

Przestrzenie nazw

- namespace Utils

  Przestrzeń nazw zawierająca funkcje narzędziowe.

Funkcje

- size_t WriteCallback (void ∗contents, size_t size, size_t nmemb, std::string ∗output)

  Funkcja callback do zapisywania odpowiedzi HTTP.
- wxDateTime Utils::parseDateTime (const std::string &dateStr)

  Konwertuje string daty na obiekt wxDateTime.

7.18.1 Opis szczegółowy

Implementacja narzędzi pomocniczych.

7.18.2 Dokumentacja funkcji

7.18.2.1 WriteCallback()

```
size_t WriteCallback (
            void * contents,
            size_t size,
            size_t nmemb,
            std::string * output)
```

Funkcja callback do zapisywania odpowiedzi HTTP.

Funkcja callback do odbierania danych HTTP poprzez libcurl.

Używana jako callback w libcurl do odbierania danych HTTP.

Parametry

contents	Wskaźnik do odebranych danych
size	Rozmiar pojedynczego elementu
nmemb	Liczba elementów
output	Wskaźnik do bufora wyjściowego (std::string)

Zwraca

Całkowity rozmiar odebranych danych w bajtach

7.19 Dokumentacja pliku src/Utils.h

Narzędzia pomocnicze używane w aplikacji.

```
#include <string>
#include <wx/wx.h>
#include <wx/datetime.h>
```

Przestrzenie nazw

- namespace Utils

  Przestrzeń nazw zawierająca funkcje narzędziowe.

Funkcje

- size_t WriteCallback (void ∗contents, size_t size, size_t nmemb, std::string ∗output)

  Funkcja callback do odbierania danych HTTP poprzez libcurl.
- wxDateTime Utils::parseDateTime (const std::string &dateStr)

  Konwertuje string daty na obiekt wxDateTime.

7.19.1 Opis szczegółowy

Narzędzia pomocnicze używane w aplikacji.

7.19.2 Dokumentacja funkcji

7.19.2.1 WriteCallback()

```
size_t WriteCallback (
            void * contents,
            size_t size,
            size_t nmemb,
            std::string * output)
```

Funkcja callback do odbierania danych HTTP poprzez libcurl.

Parametry

contents	Wskaźnik do odebranych danych
size	Rozmiar pojedynczego elementu
nmemb	Liczba elementów
output	Wskaźnik do bufora wyjściowego

Zwraca

Całkowity rozmiar odebranych danych

Funkcja callback do odbierania danych HTTP poprzez libcurl.

Używana jako callback w libcurl do odbierania danych HTTP.

Parametry

contents	Wskaźnik do odebranych danych
size	Rozmiar pojedynczego elementu
nmemb	Liczba elementów
output	Wskaźnik do bufora wyjściowego (std::string)

Zwraca

Całkowity rozmiar odebranych danych w bajtach

7.20 Utils.h

Idź do dokumentacji tego pliku.

```
00001
00005
00006 #pragma once
00007
00008 #include <string>
00009 #include <wx/wx.h>
00010 #include <wx/datetime.h>
00011
00021 size_t WriteCallback(void* contents, size_t size, size_t nmemb, std::string* output);
00022
00027 namespace Utils {
00034     wxDateTime parseDateTime(const std::string& dateStr);
00035 }
```

Skorowidz