LNWeather 0.1.0

Wygenerowano za pomocą Doxygen 1.13.2

1 Indeks przestrzeni nazw	1
1.1 Lista przestrzeni nazw	. 1
2 Indeks hierarchiczny	3
2.1 Hierarchia klas	. 3
3 Indeks klas	5
3.1 Lista klas	. 5
4 Indeks plików	7
4.1 Lista plików	. 7
5 Dokumentacja przestrzeni nazw	9
5.1 Dokumentacja przestrzeni nazw Utils	. 9
5.1.1 Opis szczegółowy	. 9
5.1.2 Dokumentacja funkcji	. 9
5.1.2.1 parseDateTime()	. 9
6 Dokumentacja klas	11
6.1 Dokumentacja klasy ApiService	. 11
6.1.1 Opis szczegółowy	. 11
6.1.2 Dokumentacja funkcji składowych	. 11
6.1.2.1 fetchDataFromAPI()	. 11
6.1.2.2 isNetworkAvailable()	. 12
6.2 Dokumentacja klasy App	13
6.2.1 Opis szczegółowy	13
6.2.2 Dokumentacja funkcji składowych	13
6.2.2.1 Onlnit()	13
6.3 Dokumentacja klasy ChartPanel	14
6.3.1 Opis szczegółowy	14
6.3.2 Dokumentacja konstruktora i destruktora	. 15
6.3.2.1 ChartPanel()	. 15
6.3.3 Dokumentacja funkcji składowych	. 15
6.3.3.1 DrawAxes()	. 15
6.3.3.2 DrawData()	. 15
6.3.3.3 DrawLabels()	. 16
6.3.3.4 GetChartData()	. 16
6.3.3.5 OnPaint()	. 16
6.3.3.6 OnSize()	. 17
6.3.3.7 SetData()	. 17
6.3.3.8 SetTitle()	. 17
6.3.4 Dokumentacja atrybutów składowych	. 18
6.3.4.1 chartData	. 18
6.3.4.2 parameter	. 18

6.3.4.3 title	18
6.4 Dokumentacja klasy DatabaseService	18
6.4.1 Opis szczegółowy	19
6.4.2 Dokumentacja konstruktora i destruktora	19
6.4.2.1 DatabaseService()	19
6.4.3 Dokumentacja funkcji składowych	20
6.4.3.1 getAirQualityFilePath()	20
6.4.3.2 getLastUpdateTime()	20
6.4.3.3 getMeasurementFilePath()	20
6.4.3.4 getSensorsFilePath()	20
6.4.3.5 getStationsFilePath()	21
6.4.3.6 hasAirQualityIndex()	21
6.4.3.7 hasMeasurementData()	21
6.4.3.8 hasSensorsData()	22
6.4.3.9 hasStationsData()	23
6.4.3.10 initializeDatabase()	23
6.4.3.11 loadAirQualityIndex()	23
6.4.3.12 loadFromFile()	24
6.4.3.13 loadMeasurementData()	24
6.4.3.14 loadSensorsData()	25
6.4.3.15 loadStationsData()	25
6.4.3.16 saveAirQualityIndex()	25
6.4.3.17 saveMeasurementData()	26
6.4.3.18 saveSensorsData()	26
6.4.3.19 saveStationsData()	26
6.4.3.20 saveToFile()	27
6.4.4 Dokumentacja atrybutów składowych	27
6.4.4.1 dbFolder	27
6.5 Dokumentacja klasy MainWindow	27
6.5.1 Opis szczegółowy	29
6.5.2 Dokumentacja konstruktora i destruktora	30
6.5.2.1 MainWindow()	30
6.5.3 Dokumentacja funkcji składowych	30
6.5.3.1 loadAirQualityIndex()	30
6.5.3.2 loadSensorsForStation()	30
6.5.3.3 loadStations()	31
6.5.3.4 OnApplyDateRange()	31
6.5.3.5 OnSaveAllToDatabase()	31
6.5.3.6 OnSaveCurrentToDatabase()	31
6.5.3.7 OnSensorSelection()	32
6.5.3.8 OnShowAnalysis()	32
6.5.3.9 OnStationSelection()	32

6.5.3.10 parseDateTime()	33
6.5.3.11 PerformDataAnalysis()	33
6.5.3.12 tryLoadAirQualityFromAPI()	34
6.5.3.13 tryLoadAirQualityFromDatabase()	34
6.5.3.14 tryLoadMeasurementsFromAPI()	35
6.5.3.15 tryLoadMeasurementsFromDatabase()	35
6.5.3.16 tryLoadMeasurementsWithDateRange()	36
6.5.3.17 tryLoadMeasurementsWithDateRangeFromDatabase()	36
6.5.3.18 tryLoadSensorsFromAPI()	37
6.5.3.19 tryLoadSensorsFromDatabase()	38
6.5.3.20 tryLoadStationsFromAPI()	38
6.5.3.21 tryLoadStationsFromDatabase()	39
6.5.3.22 updateDatabaseStatus()	39
6.5.4 Dokumentacja atrybutów składowych	39
6.5.4.1 airQualityIndex	39
6.5.4.2 analysisText	39
6.5.4.3 applyDateRangeButton	39
6.5.4.4 chartPanel	40
6.5.4.5 dataNotebook	40
6.5.4.6 dbService	40
6.5.4.7 dbStatusText	40
6.5.4.8 endDatePicker	40
6.5.4.9 font	40
6.5.4.10 measurementText	40
6.5.4.11 saveAllToDbButton	40
6.5.4.12 saveCurrentToDbButton	41
6.5.4.13 sensorChoice	41
6.5.4.14 sensors	41
6.5.4.15 showAnalysisButton	41
6.5.4.16 startDatePicker	41
6.5.4.17 stationChoice	41
6.5.4.18 stations	41
6.6 Dokumentacja struktury MeasurementData	42
6.6.1 Opis szczegółowy	42
6.6.2 Dokumentacja atrybutów składowych	42
6.6.2.1 date	42
6.6.2.2 has Value	42
6.6.2.3 value	42
7 Dokumentacja plików	43
7.1 Dokumentacja pliku src/ApiService.cpp	43
7.1.1 Opis szczegółowy	43

7.2 Dokumentacja pliku src/ApiService.h	43
7.2.1 Opis szczegółowy	43
7.3 ApiService.h	44
7.4 Dokumentacja pliku src/ChartPanel.cpp	44
7.4.1 Opis szczegółowy	44
7.5 Dokumentacja pliku src/ChartPanel.h	44
7.5.1 Opis szczegółowy	44
7.6 ChartPanel.h	45
7.7 Dokumentacja pliku src/DatabaseService.cpp	45
7.7.1 Opis szczegółowy	45
7.8 Dokumentacja pliku src/DatabaseService.h	46
7.8.1 Opis szczegółowy	46
7.9 DatabaseService.h	46
7.10 Dokumentacja pliku src/main.cpp	47
7.10.1 Opis szczegółowy	47
7.10.2 Dokumentacja funkcji	47
7.10.2.1 wxIMPLEMENT_APP()	47
	47
7.11.1 Opis szczegółowy	48
7.11.2 Dokumentacja definicji typów	48
7.11.2.1 json	48
7.11.3 Dokumentacja funkcji	48
7.11.3.1 compareStationsByName()	48
7.12 Dokumentacja pliku src/MainWindow.h	49
7.12.1 Opis szczegółowy	49
7.13 MainWindow.h	49
7.14 Dokumentacja pliku src/Models.h	50
7.14.1 Opis szczegółowy	51
7.15 Models.h	51
7.16 Dokumentacja pliku src/resource.h	51
7.16.1 Dokumentacja definicji	51
7.16.1.1 IDI_ICON1	51
7.17 resource.h	51
7.18 Dokumentacja pliku src/Utils.cpp	52
7.18.1 Opis szczegółowy	52
7.18.2 Dokumentacja funkcji	52
7.18.2.1 WriteCallback()	52
7.19 Dokumentacja pliku src/Utils.h	53
7.19.1 Opis szczegółowy	53
7.19.2 Dokumentacja funkcji	53
7.19.2.1 WriteCallback()	53
7.20 Utils.h	54

Skorowidz 55

Indeks przestrzeni nazw

1.1 Lista przestrzeni nazw

lutai	znaidilla	SIE W	/C7VCtkIA	przestrzenie nazv	v wraz z ic	n krotkimi	unicami.
iulaj	Znajaują	SIÇ W	132 y 3 ti 110	pizostizonio nazv	v vviaz z ic		opisaiii.

Utils Prze

estrzeń nazw zawierająca funkcje narzędziowe	•	a
estizeti nazw zawierająca iurikcje narzędziowe	,	9

2	Indeks przestrzeni nazw

Indeks hierarchiczny

2.1 Hierarchia klas

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

ApiService										 				 		 			 		11
DatabaseServic	е									 				 		 			 		18
MeasurementDa	ata									 				 		 			 		42
wxApp																					
App																 					13
wxFrame																					
MainWindow	<i>.</i>															 					27
wxPanel																					
ChartPanel							 									 			 		14

Indeks klas

3.1 Lista klas

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

ApiService	
Klasa obsługująca komunikację z zewnętrznymi API	11
App	
Główna klasa aplikacji wxWidgets	13
ChartPanel	
Panel wyświetlający wykres danych pomiarowych	14
DatabaseService	
Klasa obsługująca lokalną "bazę danych" (pliki JSON)	18
MainWindow	
Klasa implementująca główne okno aplikacji	27
MeasurementData	
Struktura do przechowywania danych pomiarowych dla wykresu	42

6 Indeks klas

Indeks plików

4.1 Lista plików

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

Src/ApiService.cpp	
Implementacja usługi komunikacji z API	43
src/ApiService.h	
Usługa do komunikacji z zewnętrznym API	43
src/ChartPanel.cpp	
Implementacja panelu wykresow danych pomiarowych	44
src/ChartPanel.h	
Panel wykresu do wizualizacji danych pomiarowych	44
src/DatabaseService.cpp	
Implementacja usługi bazy danych	45
src/DatabaseService.h	
Usługa zarządzająca lokalną bazą danych	46
src/main.cpp	
Punkt wejściowy aplikacji	47
src/MainWindow.cpp	
Implementacja głównego okna aplikacji LNWeather	47
src/MainWindow.h	
Główne okno aplikacji do monitorowania jakości powietrza	49
src/Models.h	
Definicje struktur danych używanych w aplikacji	
src/resource.h	51
src/Utils.cpp	
Implementacja narzędzi pomocniczych	52
src/Utils.h	
Narzędzia pomocnicze używane w aplikacji	53

8 Indeks plików

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()

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

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()

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 Onlnit ()
 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 ¶mName)

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()

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()

Rysuje osie wykresu.

Parametry

dc	Kontekst urządzenia
rect	Prostokąt obszaru wykresu

6.3.3.2 DrawData()

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()

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()

Obsługa zdarzenia rysowania.

Parametry

event	Zdarzenie rysowania
event	∠darzenie rysowania

Rysuje wykres z wykorzystaniem buforowania.

Parametry

event	Zdarzenie rysowania
-------	---------------------

6.3.3.6 OnSize()

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()

Ustawia dane do wyświetlenia na wykresie.

Parametry

data	Wektor danych pomiarowych
paramName	Nazwa parametru

6.3.3.8 SetTitle()

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 sensorld)

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()

```
\begin{tabular}{ll} {\tt std::string DatabaseService::getAirQualityFilePath (} \\ & {\tt int } stationId) \end{tabular}
```

Zwraca ścieżkę pliku z indeksem jakości powietrza dla stacji.

Parametry

station←	ID stacji
ld	

Zwraca

std::string Ścieżka do pliku

6.4.3.2 getLastUpdateTime()

Pobiera czas ostatniej aktualizacji pliku.

Parametry

filePath	Ścieżka do pliku
----------	------------------

Zwraca

wxDateTime Czas ostatniej modyfikacji

6.4.3.3 getMeasurementFilePath()

Zwraca ścieżkę pliku z danymi pomiarowymi dla czujnika.

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

std::string Ścieżka do pliku

6.4.3.4 getSensorsFilePath()

```
\begin{tabular}{ll} {\tt std::string DatabaseService::getSensorsFilePath (} \\ & int \ stationId) \end{tabular}
```

Zwraca ścieżkę pliku z danymi czujników dla stacji.

Parametry

station⊷	ID stacji
ld	

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()

Sprawdza czy indeks jakości powietrza dla stacji istnieje w bazie.

Parametry

station⊷	ID stacji
ld	

Zwraca

true Jeśli dane istnieją false Jeśli dane nie istnieją

Parametry

station←	ID stacji
ld	

Zwraca

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

6.4.3.7 hasMeasurementData()

```
\label{local_problem} \begin{tabular}{ll} \b
```

Sprawdza czy dane pomiarowe dla czujnika istnieją w bazie.

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

true Jeśli dane istnieją false Jeśli dane nie istnieją

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

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

6.4.3.8 hasSensorsData()

Sprawdza czy dane czujników dla stacji istnieją w bazie.

Parametry

station←	ID stacji
ld	

Zwraca

true Jeśli dane istnieją false Jeśli dane nie istnieją

Parametry

station⊷	ID stacji
ld	

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()

```
\begin{tabular}{ll} {\tt std::string DatabaseService::loadAirQualityIndex (} \\ & & {\tt int } stationId) \end{tabular}
```

Wczytuje indeks jakości powietrza dla stacji.

Parametry

station←	ID stacji
ld	

Zwraca

std::string Dane w formacie JSON

Parametry

station⇔	ID stacji
ld	

Zwraca

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

6.4.3.12 loadFromFile()

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()

Wczytuje dane pomiarowe dla czujnika.

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

std::string Dane w formacie JSON

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

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

6.4.3.14 loadSensorsData()

```
\begin{tabular}{ll} {\tt std::string DatabaseService::loadSensorsData (} \\ & & {\tt int } \begin{tabular}{ll} {\tt stationId}) \end{tabular}
```

Wczytuje dane czujników dla stacji.

Parametry

station←	ID stacji
ld	

Zwraca

std::string Dane w formacie JSON

Parametry

station←	ID stacji
ld	

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()

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()

Zapisuje dane pomiarowe dla czujnika.

Parametry

sensor⊷	ID czujnika
ld	
jsonData	Dane w formacie JSON

Zwraca

true Jeśli zapis się powiódł false Jeśli wystąpił błąd

6.4.3.18 saveSensorsData()

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()

Zapisuje dane stacji do bazy.

Zapisuje dane stacji do pliku.

Parametry

jsonData Dane w formacie JSO	V
------------------------------	---

Zwraca

true Jeśli zapis się powiódł false Jeśli wystąpił błąd

6.4.3.20 saveToFile()

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:

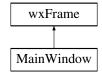
- src/DatabaseService.h
- src/DatabaseService.cpp

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 ¶mName)

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()

Ładuje indeks jakości powietrza dla stacji.

Ładuje informacje o indeksie jakości powietrza dla wybranej stacji.

Parametry

station← Id	ID stacji, dla której należy załadować indeks jakości powietrza
station⊷ Id	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()

Ładuje czujniki dla wybranej stacji.

Ładuje informacje o czujnikach dla wybranej stacji.

Parametry

station⊷ Id	ID stacji, dla której należy załadować czujniki
station⊷ Id	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()

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()

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()

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()

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()

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()

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()

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()

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()

Próbuje załadować indeks jakości powietrza z API.

Próbuje załadować indeks jakości powietrza dla stacji z API GIOŚ

Parametry

station←	ID stacji
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

station←	Identyfikator stacji
ld	

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()

Próbuje załadować indeks jakości powietrza z lokalnej bazy danych.

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

Parametry

station←	ID stacji
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

station←	Identyfikator stacji
ld	

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()

Próbuje załadować pomiary z API.

Próbuje załadować dane pomiarowe dla czujnika z API GIOŚ

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

sensor⊷	Identyfikator czujnika
ld	

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()

Próbuje załadować pomiary z lokalnej bazy danych.

Próbuje załadować dane pomiarowe dla czujnika z lokalnej bazy danych.

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

sensor⊷	Identyfikator czujnika
ld	

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()

Próbuje załadować pomiary z określonego zakresu dat.

Próbuje załadować dane pomiarowe dla wybranego zakresu dat z API GIOŚ

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

sensor⊷	Identyfikator czujnika
ld	

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()

```
\begin{tabular}{ll} bool $MainWindow::tryLoadMeasurementsWithDateRangeFromDatabase ( \\ int $sensorId$) [private] \end{tabular}
```

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

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

Parametry

sensor⊷	ID czujnika
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

sensor⊷	Identyfikator czujnika
ld	

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()

Próbuje załadować czujniki dla stacji z API.

Próbuje załadować informacje o czujnikach dla stacji z API GIOŚ

Parametry

station←	ID stacji
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

station⊷	Identyfikator stacji
ld	

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()

```
\begin{tabular}{ll} bool $\tt MainWindow::tryLoadSensorsFromDatabase ( \\ & int $\it stationId)$ [private] \\ \end{tabular}
```

Próbuje załadować czujniki dla stacji z lokalnej bazy danych.

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

Parametry

station←	ID stacji
ld	

Zwraca

true Jeśli załadowanie się powiodło false Jeśli wystąpił błąd

Parametry

station←	Identyfikator stacji
ld	

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

 $\verb|std::vector<| std::pair<| int, std::string> > MainWindow::stations [private]|$

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 has Value

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 has Value

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

ldź do dokumentacji tego pliku.

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 45

7.6 ChartPanel.h

ldź do dokumentacji tego pliku.

```
00005
00006 #pragma once
00007
00008 #include <wx/wx.h>
00009 #include <wx/datetime.h>
00010 #include <vector>
00011
00016 struct MeasurementData {
        wxDateTime date;
00017
00018
         double value;
00019
         bool hasValue;
00020 };
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
          const std::vector<MeasurementData>& GetChartData() const { return chartData; }
00058
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
          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

ldź 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
          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
```

```
std::string getStationsFilePath();
00160
00167
          std::string getSensorsFilePath(int stationId);
00168
          std::string getMeasurementFilePath(int sensorId);
00175
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()

Porównuje dwie stacje na podstawie ich nazw.

Funkcja pomocnicza używana do sortowania stacji w kolejności alfabetycznej.

Parametry

а	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

ldź 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
           wxStaticText* dbStatusText;
```

```
00075
00076
          //Analiza danych
08000
          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;</pre>
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
00200
          bool tryLoadStationsFromAPI();
00201
00208
          bool tryLoadStationsFromDatabase();
00209
00217
          bool trvLoadSensorsFromAPI(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>
```

7.15 Models.h 51

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

ldź do dokumentacji tego pliku.

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

ldź 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
00012 #define _APS_NEXT_COMMAND_VALUE
00013 #define _APS_NEXT_CONTROL_VALUE
00014 #define _APS_NEXT_SYMED_VALUE
                                                       40001
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()

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()

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

ldź do dokumentacji tego pliku.

Skorowidz

airQualityIndex	saveAirQualityIndex, 25
MainWindow, 39	saveMeasurementData, 26
analysisText	saveSensorsData, 26
MainWindow, 39	saveStationsData, 26
ApiService, 11	saveToFile, 27
fetchDataFromAPI, 11	dataNotebook
isNetworkAvailable, 12	MainWindow, 40
App, 13	date
Onlnit, 13	MeasurementData, 42
applyDateRangeButton	dbFolder
MainWindow, 39	DatabaseService, 27
,	dbService
chartData	MainWindow, 40
ChartPanel, 18	dbStatusText
ChartPanel, 14	MainWindow, 40
chartData, 18	DrawAxes
ChartPanel, 15	ChartPanel, 15
DrawAxes, 15	DrawData
DrawData, 15	ChartPanel, 15
DrawLabels, 16	DrawLabels
GetChartData, 16	ChartPanel, 16
OnPaint, 16	, , ,
OnSize, 17	endDatePicker
parameter, 18	MainWindow, 40
SetData, 17	
SetTitle, 17	fetchDataFromAPI
title, 18	ApiService, 11
chartPanel	font
MainWindow, 39	MainWindow, 40
compareStationsByName	
MainWindow.cpp, 48	getAirQualityFilePath
	DatabaseService, 20
DatabaseService, 18	GetChartData
DatabaseService, 19	ChartPanel, 16
dbFolder, 27	getLastUpdateTime
getAirQualityFilePath, 20	DatabaseService, 20
getLastUpdateTime, 20	getMeasurementFilePath
getMeasurementFilePath, 20	DatabaseService, 20
getSensorsFilePath, 20	getSensorsFilePath
getStationsFilePath, 21	DatabaseService, 20
hasAirQualityIndex, 21	getStationsFilePath
hasMeasurementData, 21	DatabaseService, 21
hasSensorsData, 22	le a a Alia Occasi i territoral acces
hasStationsData, 22	hasAirQualityIndex
initializeDatabase, 23	DatabaseService, 21
loadAirQualityIndex, 23	hasMeasurementData
loadFromFile, 24	DatabaseService, 21
loadMeasurementData, 24	hasSensorsData
loadSensorsData, 25	DatabaseService, 22
loadStationsData, 25	hasStationsData

56 SKOROWIDZ

DatabaseService, 22	showAnalysisButton, 41
hasValue	startDatePicker, 41
MeasurementData, 42	stationChoice, 41
	stations, 41
IDI_ICON1	tryLoadAirQualityFromAPI, 33
resource.h, 51	tryLoadAirQualityFromDatabase, 34
initializeDatabase	tryLoadMeasurementsFromAPI, 35
DatabaseService, 23	tryLoadMeasurementsFromDatabase, 35
isNetworkAvailable	tryLoadMeasurementsWithDateRange, 36
ApiService, 12	tryLoadMeasurementsWithDateRangeFromData-
	base, 36
json	tryLoadSensorsFromAPI, 37
MainWindow.cpp, 48	tryLoadSensorsFromDatabase, 37
land Air Ovalitula day	tryLoadStationsFromAPI, 38
loadAirQualityIndex	tryLoadStationsFromDatabase, 38
DatabaseService, 23	updateDatabaseStatus, 39
MainWindow, 30	MainWindow.cpp
loadFromFile	compareStationsByName, 48
DatabaseService, 24	json, 48
loadMeasurementData	MeasurementData, 42
DatabaseService, 24 IoadSensorsData	date, 42
	hasValue, 42
DatabaseService, 25 IoadSensorsForStation	value, 42
	measurementText
MainWindow, 30	MainWindow, 40
loadStations MainWindow 30	
MainWindow, 30	OnApplyDateRange
loadStationsData	MainWindow, 31
DatabaseService, 25	OnInit
main.cpp	App, 13
wxIMPLEMENT_APP, 47	OnPaint
MainWindow, 27	ChartPanel, 16
airQualityIndex, 39	OnSaveAllToDatabase
analysisText, 39	MainWindow, 31
applyDateRangeButton, 39	OnSaveCurrentToDatabase
chartPanel, 39	MainWindow, 31
dataNotebook, 40	OnSensorSelection
dbService, 40	MainWindow, 32
dbStatusText, 40	OnShowAnalysis
endDatePicker, 40	MainWindow, 32
font, 40	OnSize
loadAirQualityIndex, 30	ChartPanel, 17
loadSensorsForStation, 30	OnStationSelection
loadStations, 30	MainWindow, 32
MainWindow, 30	parameter
measurementText, 40	ChartPanel, 18
OnApplyDateRange, 31	parseDateTime
OnSaveAllToDatabase, 31	MainWindow, 33
OnSaveCurrentToDatabase, 31	Utils, 9
OnSensorSelection, 32	PerformDataAnalysis
OnShowAnalysis, 32	MainWindow, 33
OnStationSelection, 32	waniyyinaow, 33
parseDateTime, 33	resource.h
PerformDataAnalysis, 33	IDI_ICON1, 51
saveAllToDbButton, 40	,
saveCurrentToDbButton, 40	saveAirQualityIndex
sensorChoice, 41	DatabaseService, 25
sensors, 41	saveAllToDbButton

SKOROWIDZ 57

MainWindow, 40	tryLoadStationsFromAPI
saveCurrentToDbButton	MainWindow, 38
MainWindow, 40	tryLoadStationsFromDatabase
saveMeasurementData	MainWindow, 38
DatabaseService, 26	
saveSensorsData	updateDatabaseStatus
DatabaseService, 26	MainWindow, 39
saveStationsData	Utils, 9
DatabaseService, 26	parseDateTime, 9
saveToFile	Utils.cpp
	WriteCallback, 52
DatabaseService, 27 sensorChoice	Utils.h
	WriteCallback, 53
MainWindow, 41	vviite dailback, 30
sensors	value
MainWindow, 41	MeasurementData, 42
SetData	Wicasarcinicitibata, 42
ChartPanel, 17	WriteCallback
SetTitle	Utils.cpp, 52
ChartPanel, 17	
showAnalysisButton	Utils.h, 53
MainWindow, 41	wxIMPLEMENT_APP
src/ApiService.cpp, 43	main.cpp, 47
src/ApiService.h, 43, 44	
src/ChartPanel.cpp, 44	
src/ChartPanel.h, 44, 45	
src/DatabaseService.cpp, 45	
src/DatabaseService.h, 46	
src/main.cpp, 47	
src/MainWindow.cpp, 47	
src/MainWindow.h, 49	
src/Models.h, 50, 51	
src/resource.h, 51	
src/Utils.cpp, 52	
src/Utils.h, 53, 54	
startDatePicker	
MainWindow, 41	
stationChoice	
MainWindow, 41	
stations	
MainWindow, 41	
title	
ChartPanel, 18	
tryLoadAirQualityFromAPI	
MainWindow, 33	
tryLoadAirQualityFromDatabase	
MainWindow, 34	
tryLoadMeasurementsFromAPI	
MainWindow, 35	
tryLoadMeasurementsFromDatabase	
•	
MainWindow, 35	
tryLoadMeasurementsWithDateRange	
MainWindow, 36	
tryLoadMeasurementsWithDateRangeFromDatabase	
MainWindow, 36	
tryLoadSensorsFromAPI	
MainWindow, 37	
tryLoadSensorsFromDatabase	
MainWindow, 37	