Progetto di Tecnologie e applicazioni web

a.a. 2014-15

Meteo Grappa app e server

Alessandro Bicciato

Indice

1 Introduzione

Come progetto di tecnologie e applicazioni web doveva venire preso in analisi un sito web che fornisse dati di qualsiasi natura e che venissero elaborati. I compiti si possono dividere in tre categorie:

- Il parser dei dati dal sito origine
- Il sito che mostra i dati e una loro elaborazione
- L'app android che anch'essa mostra i dati e una loro elaborazione

Il sito scelto per questo progetto è quello della stazione meteo del rifugio sul monte Grappa visitabile a http://www.cimagrappa.it/meteo/ che fornisce quattro webcam e dati esaurienti delle condizioni meteo in aggiornamento continuo (il tempo di aggiornamento oscilla fra i 10 e i 30 secondi). I sorgenti del server e dell'app sono disponibili su github.

1.1 Strumenti

Per la realizzazione del progetto è stata usato l'ide Netbeans 8.0 per il server java, in cui sono integrati il server web Glassfish 4.0 e il servizio di database Apache Derby 10.11, Android Studio 1.1.0 per l'app, Adobe Dreamveawer CC 2014 per l'interfaccia web, Adobe Photoshop CC 2014 per la modifica e la realizzazione delle immagini, più a livello sono state usate le librerie esterne jsoup, per scaricare le pagine web, e gson, per generare stringhe con la sintassi JSON.

2 Server

Il server ha due funzioni: deve scaricare i dati e deve fornire il sito web. Come linguaggio server side è stato scelto java, come suggerito. Per scaricare i dati viene lanciata una servlet ogni 5 minuti che scarica i dati e per il sito vengono usate delle pagine jsp con codice html, java e javascript.

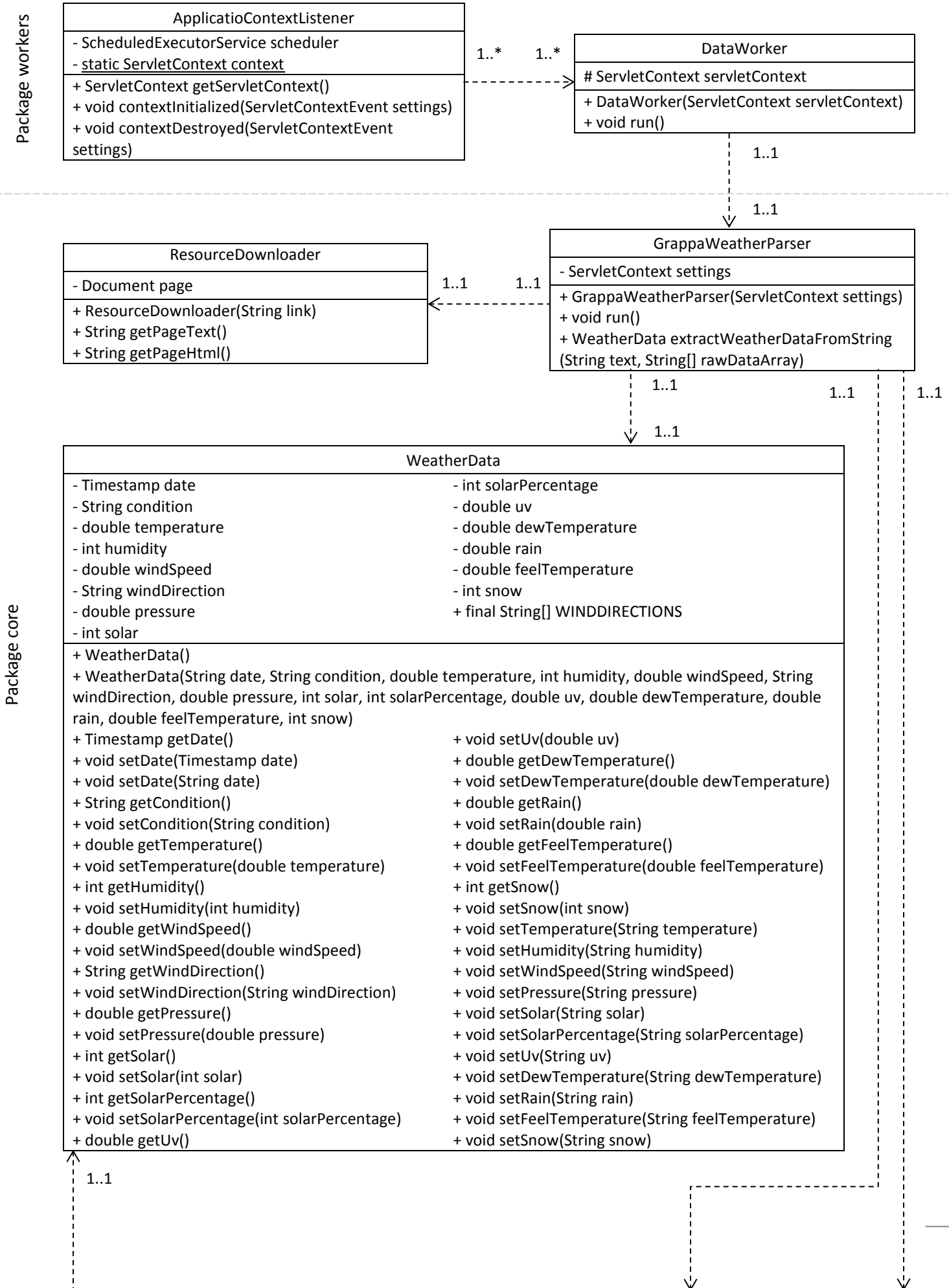
2.1 Servlet per lo scaricamento dei dati

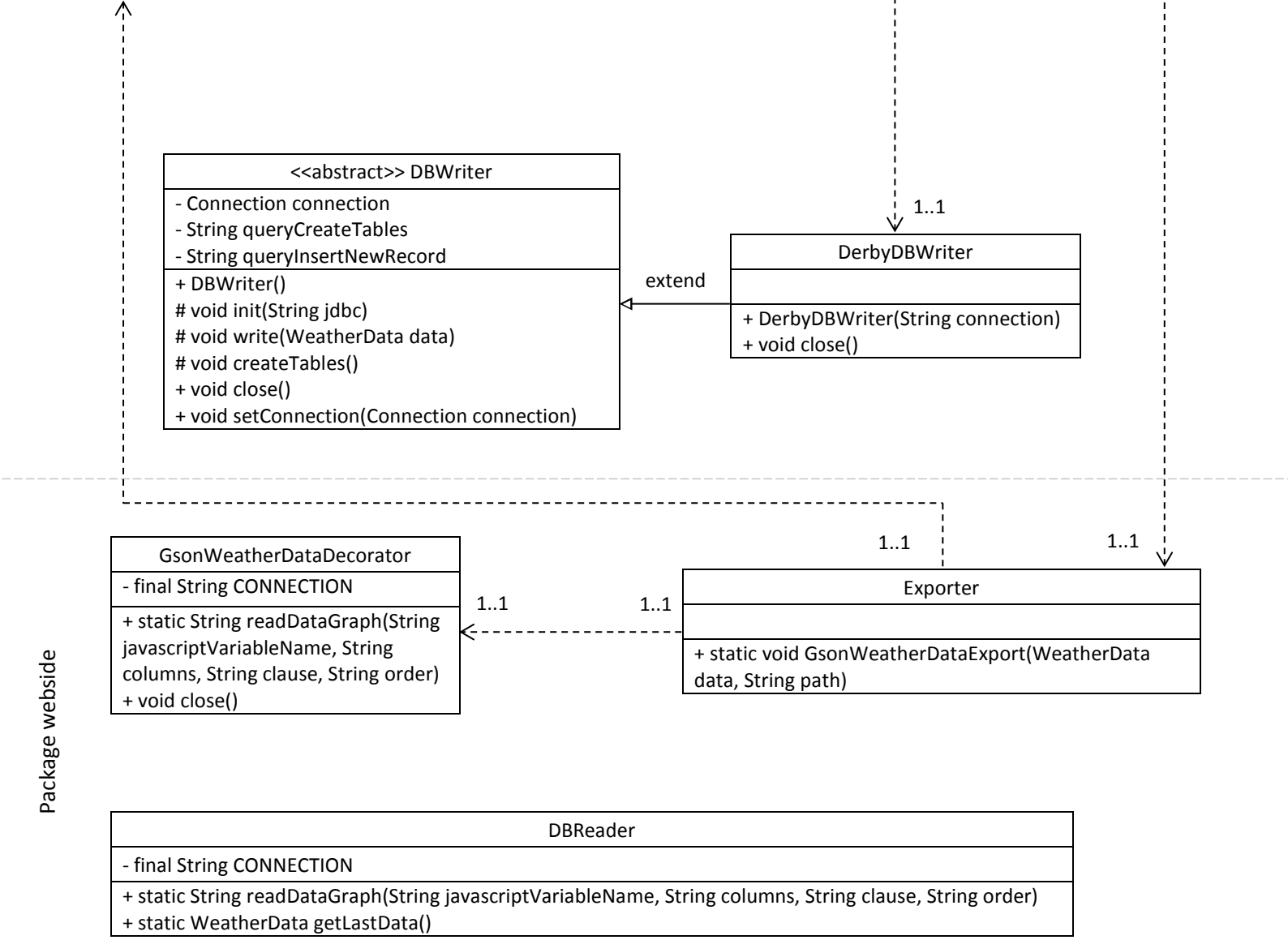
Dopo una breve analisi del sito è stato riscontrato che la versione base della pagina dei dati non riesce a fornire le misurazioni attuali siccome viene chiamata una funzione javascript che scarica dati da un'altra sorgente eccetto per i dati dell'altezza della neve. L'ipotesi fatta seguendo un informazione divulgata dai gestori del rifugio è che l'altezza della neve viene calcolata con la media dell'altezza fra i due versanti e che quindi utilizza un sistema diverso da quello delle altre misurazioni. Quindi l'altezza della neve viene scaricata dal sito http://www.meteocimagrappa.it/wd/tabella.html dove c'è la tabella di tutto e viene estratto usando un'espressione regolare mentre gli altri dati sono estratti da un file txt contenente solo valori da http://www.meteocimagrappa.it/wd/clientraw.txt. Per il corretto funzionamento le corrispondenze dei valori sono state estratte dal codice javascript della pagina della tabella.

Il codice java si divide in due package il core e il workers. In ordine di esecuzione, il workers contiene le classi che servono per lanciare la servlet mentre il core serve per la parte operativa di analisi del testo ed inserimento nel database dei dati. La scelta della divisione è stata per mantenere un buon livello di scalabilità dell'applicazione perché il package core può lavorare da solo e il package workers può cambiare i compiti cambiando solamente package d'importazione e classe di avviamento.

Come extra della servlet viene generato un file JSON dei dati raccolti che serviranno all'app per mostrare l'ultimo aggiornamento. La funzione che lo crea viene richiamata dal parser prima dell'inserimento e le dichiarazioni si trovano nel package webside dove sono contenute tutte le funzionalità riguardanti le pagine web.

2.2 Diagramma UML delle classi





DBReader
- final String CONNECTION
+ static String readDataGraph(String javascriptVariableName, String columns, String clause, String order) + static WeatherData getLastData()

2.3 Sito web

Il sito web è composto di due pagine, la home che mostra l'ultimo rilevamento scaricato e la pagina dei grafici che mostra i grafici relativi alle temperature, all'umidità, alla velocità del vento e l'intensità della pioggia. Ci sono altri due file visitabili che però servono all'app: il file json con l'ultimo rilevamento e una pagina jsp che mostra solamente un grafico e che interrogandola opportunamente mostra tutti i grafici disponibili nella pagina del sito dei grafici. Per il sito è stato usato un template standard offerto da Adobe Dreamweaver CC che fornisce anche un'interfaccia mobile.

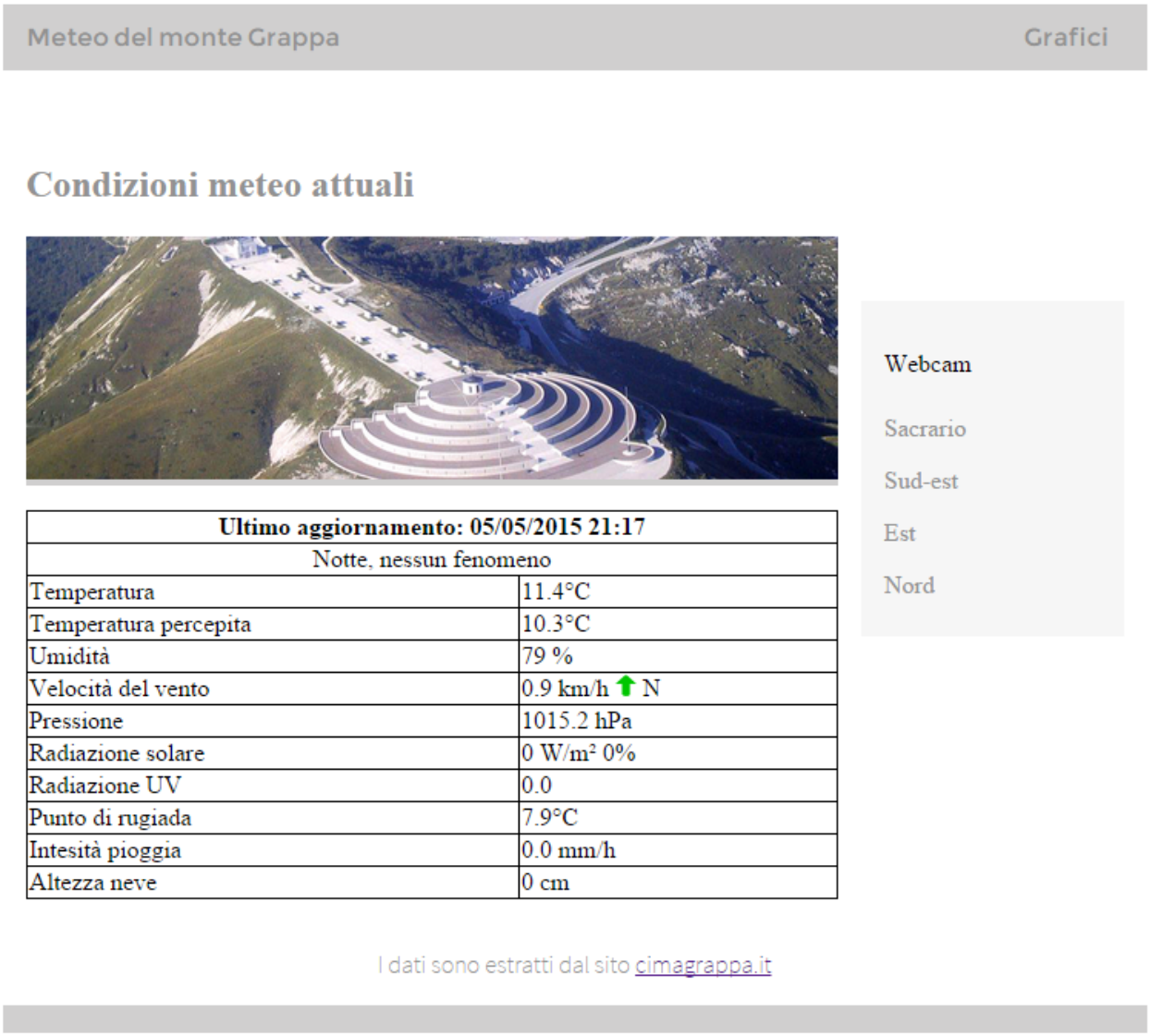


Figura 2.1 - Home del sito

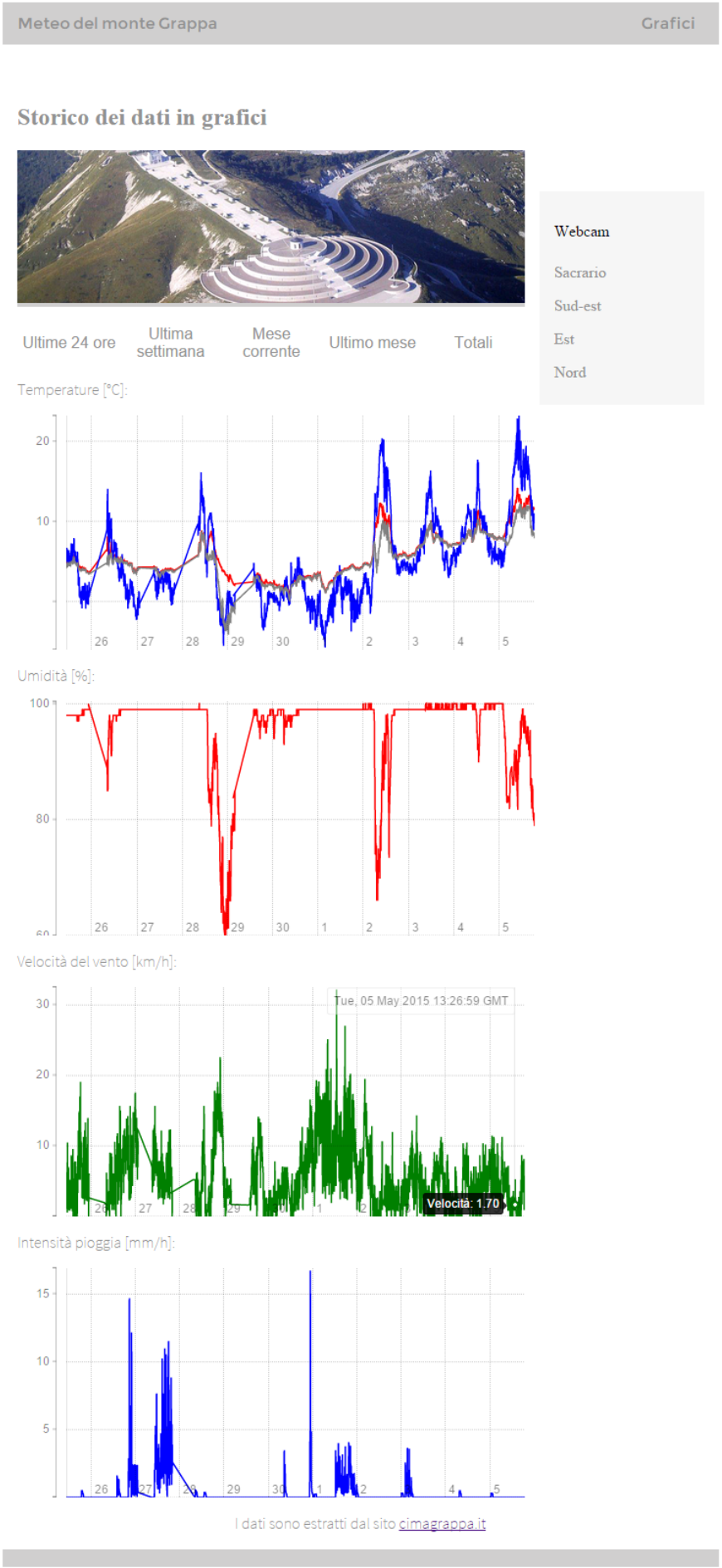


Figura 2.2 - Pagina dei grafici

2.3.1 Parte operativa server side

Sostanzialmente la parte di codice server side sulle pagine serve per leggere dal database le informazioni del meteo utilizzando la classe DBReader che contiene dei metodi statici per estrarre le varie informazioni evitando di esporre una connessione al database.

2.3.2 Parte operativa client side

La parte client side, in questo caso quella di codice javascript, ha il compito di creare i grafici e adattarli alla dimensione della finestra. I grafici vengono generati usando il toolkit Rickshaw http://code.shutterstock.com/rickshaw/.

2.4 Javadoc

2.4.1 Package core

2.4.1.1 Class DBWriter

java.lang.Object
core.DBWriter

Direct Known Subclasses:
`DerbyDBWriter`

```
public abstract class DBWriter
extends java.lang.Object
```

Operate between the database and the data.

Constructor Summary	
Constructor and Description	
`DBWriter()` Set null the connection	

Method Summary	
Modifier and Type	**Method and Description**
`abstract void`	`close()` Close the connection with the database.
`protected void`	`createTables()` Create initial tables on the database.
`protected void`	`init(java.lang.String jdbc)` Initialize the database if it's empty.
`void`	`setConnection(java.sql.Connection connection)` Set connection of the database.
`protected void`	`write(WeatherData data)` Write the data stored in a WeatherData object.
Methods inherited from class java.lang.Object	

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString,
wait, wait, wait
```

Constructor Detail

DBWriter

```
public DBWriter()
```

Set null the connection

Method Detail

init

```
protected void init(java.lang.String jdbc)
```

Initialize the database if it's empty. It creates a table in the database MeteoGrappa named MeteoGrappa_data where there will be stored all the information when the main class download the data.

Parameters:

```
jdbc - odbc string for connect and create the database
```

write

```
protected void write(WeatherData data)
```

Write the data stored in a WeatherData object.

Parameters:

```
data - WeatherData object to use for give the data to be written on
the database
```

createTables

```
protected void createTables()
```

Create initial tables on the database. The sql for creating table is:

```
CREATE TABLE MeteoGrappa_Data (datetime TIMESTAMP NOT NULL, condition
VARCHAR(30), temperature REAL, humidity INTEGER, wind_speed
REAL,wind_direction VARCHAR(10), pressure REAL, solar_radiation
INTEGER,solar_percentage INTEGER, uv REAL, deaf_temperature REAL, rain
REAL, feel_temperature REAL, snow INTEGER, PRIMARY KEY (datetime));
```

close

```
public abstract void close()
```

Close the connection with the database.

setConnection

```
public void setConnection(java.sql.Connection connection)
```

Set connection of the database.

Parameters:

```
connection - Connection to the database
```

2.4.1.2 Class DerbyDBWriter

java.lang.Object
    core.DBWriter
        core.DerbyDBWriter

```
public class DerbyDBWriter
extends DBWriter
```
DBWriter for Derby databases.

Constructor Summary

Constructor and Description
`DerbyDBWriter(java.lang.String connection)` Open the database or create a new one with default tables.

Method Summary

Modifier and Type	Method and Description
`void`	`close()` Close the connection with the database.

Methods inherited from class core.DBWriter
`createTables, init, setConnection, write`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DerbyDBWriter

```
public DerbyDBWriter(java.lang.String connection)
```
Open the database or create a new one with default tables.

Parameters:

`connection - path for connecting to the database`

Method Detail

close

```
public void close()
```
Close the connection with the database.

Specified by:

`close in class DBWriter`

2.4.1.3 Class GrappaWeatherParser

java.lang.Object

core.GrappaWeatherParser

All Implemented Interfaces:

`java.lang.Runnable`

```
public class GrappaWeatherParser
extends java.lang.Object
```

implements java.lang.Runnable

This class elaborates the web page choosed for the parsing of the data and putted on the database.

Constructor Summary

Constructor and Description
GrappaWeatherParser(javax.servlet.ServletContext settings)

Method Summary

Modifier and Type	Method and Description
WeatherData	extractWeatherDataFromString(java.lang.String text, java.lang.String[] rawDataArray) This method extract data from a text.
void	run() Runner method of the servlet.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GrappaWeatherParser

public GrappaWeatherParser(javax.servlet.ServletContext settings)

Method Detail

run

public void run()

Runner method of the servlet.

In order it does:

- Download the web page.
- Parse page for extract data.
- Export the parsed data.
- Add data to new record in the database.

Specified by:

run in interface java.lang.Runnable

extractWeatherDataFromString

public WeatherData extractWeatherDataFromString(java.lang.String text, java.lang.String[] rawDataArray)

This method extract data from a text. It uses a regular expression for extract the variables and them are combined in a WeatherData object.

Parameters:

text - string of the web page where extract data

rawDataArray - contains the second document downloaded by the page with raw data

Returns:

WeatherData object that contains the data extract

2.4.1.4 Class ResourceDownloader

java.lang.Object
core.ResourceDownloader

---

public class **ResourceDownloader**
extends java.lang.Object
This class download the content of a web page.

Constructor Summary
Constructor and Description
`ResourceDownloader(java.lang.String link)` It downloads the web page and store in a String variable.

Method Summary	
Modifier and Type	**Method and Description**
`java.lang.String`	`getPageHtml()`
`java.lang.String`	`getPageText()`
Methods inherited from class java.lang.Object	
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`	

Constructor Detail
ResourceDownloader

public ResourceDownloader(java.lang.String link)
It downloads the web page and store in a String variable.
Parameters:
link - url of the page to download

Method Detail
getPageText

public java.lang.String getPageText()

getPageHtml

public java.lang.String getPageHtml()

2.4.1.5 Class WeatherData

java.lang.Object
core.WeatherData

---

public class **WeatherData**
extends java.lang.Object

This class contains the data of weather from a page.

Field Summary

Modifier and Type	Field and Description
`static java.lang.String[]`	**WINDDIRECTIONS** Array of the wind directions.

Constructor Summary

Constructor and Description
WeatherData()
WeatherData(java.lang.String date, java.lang.String condition, double temperature, int humidity, double windSpeed, java.lang.String windDirection, double pressure, int solar, int solarPercentage, double uv, double dewTemperature, double rain, double feelTemperature, int snow)

Method Summary

Modifier and Type	Method and Description
java.lang.String	**getCondition**()
java.sql.Timestamp	**getDate**()
double	**getDewTemperature**()
double	**getFeelTemperature**()
int	**getHumidity**()
double	**getPressure**()
double	**getRain**()
int	**getSnow**()
int	**getSolar**()
int	**getSolarPercentage**()
double	**getTemperature**()
double	**getUv**()
java.lang.String	**getWindDirection**()
double	**getWindSpeed**()
void	**setCondition**(java.lang.String condition)
void	**setDate**(java.lang.String date)
void	**setDate**(java.sql.Timestamp date)

void	setDewTemperature(double dewTemperature)
void	setDewTemperature(java.lang.String dewTemperature)
void	setFeelTemperature(double feelTemperature)
void	setFeelTemperature(java.lang.String feelTemperature)
void	setHumidity(int humidity)
void	setHumidity(java.lang.String humidity)
void	setPressure(double pressure)
void	setPressure(java.lang.String pressure)
void	setRain(double rain)
void	setRain(java.lang.String rain)
void	setSnow(int snow)
void	setSnow(java.lang.String snow)
void	setSolar(int solar)
void	setSolar(java.lang.String solar)
void	setSolarPercentage(int solarPercentage)
void	setSolarPercentage(java.lang.String solarPercentage)
void	setTemperature(double temperature)
void	setTemperature(java.lang.String temperature)
void	setUv(double uv)
void	setUv(java.lang.String uv)
void	setWindDirection(java.lang.String windDirection)
void	setWindSpeed(double windSpeed)
void	setWindSpeed(java.lang.String windSpeed)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

WINDDIRECTIONS

public static final java.lang.String[] WINDDIRECTIONS

Array of the wind directions.

Constructor Detail

WeatherData

```
public WeatherData()
```

WeatherData

```
public WeatherData(java.lang.String date,
                   java.lang.String condition,
                   double temperature,
                   int humidity,
                   double windSpeed,
                   java.lang.String windDirection,
                   double pressure,
                   int solar,
                   int solarPercentage,
                   double uv,
                   double dewTemperature,
                   double rain,
                   double feelTemperature,
                   int snow)
```

Method Detail

getDate

```
public java.sql.Timestamp getDate()
```

setDate

```
public void setDate(java.sql.Timestamp date)
```

setDate

```
public void setDate(java.lang.String date)
throws java.lang.IllegalArgumentException
```

Throws:

`java.lang.IllegalArgumentException`

getCondition

```
public java.lang.String getCondition()
```

setCondition

```
public void setCondition(java.lang.String condition)
```

getTemperature

```
public double getTemperature()
```

setTemperature

```
public void setTemperature(double temperature)
```

getHumidity

```
public int getHumidity()
```

setHumidity

```
public void setHumidity(int humidity)
```

getWindSpeed

```
public double getWindSpeed()
```

setWindSpeed

```
public void setWindSpeed(double windSpeed)
```

getWindDirection

```
public java.lang.String getWindDirection()
```

setWindDirection

```
public void setWindDirection(java.lang.String windDirection)
```

getPressure

```
public double getPressure()
```

setPressure

```
public void setPressure(double pressure)
```

getSolar

```
public int getSolar()
```

setSolar

```
public void setSolar(int solar)
```

getSolarPercentage

```
public int getSolarPercentage()
```

setSolarPercentage

```
public void setSolarPercentage(int solarPercentage)
```

getUv

```
public double getUv()
```

setUv

```
public void setUv(double uv)
```

getDewTemperature

```
public double getDewTemperature()
```

setDewTemperature

```
public void setDewTemperature(double dewTemperature)
```

getRain

```
public double getRain()
```

setRain

```
public void setRain(double rain)
```

getFeelTemperature

```
public double getFeelTemperature()
```

setFeelTemperature

```
public void setFeelTemperature(double feelTemperature)
```

getSnow

```
public int getSnow()
```

setSnow

```
public void setSnow(int snow)
```

setTemperature

```
public void setTemperature(java.lang.String temperature)
```

setHumidity

```
public void setHumidity(java.lang.String humidity)
```

setWindSpeed

```
public void setWindSpeed(java.lang.String windSpeed)
```

setPressure

```
public void setPressure(java.lang.String pressure)
```

setSolar

```
public void setSolar(java.lang.String solar)
```

setSolarPercentage

```
public void setSolarPercentage(java.lang.String solarPercentage)
```

setUv

```
public void setUv(java.lang.String uv)
```

setDewTemperature

```
public void setDewTemperature(java.lang.String dewTemperature)
```

setRain

```
public void setRain(java.lang.String rain)
```

setFeelTemperature

```
public void setFeelTemperature(java.lang.String feelTemperature)
```

setSnow

```
public void setSnow(java.lang.String snow)
```

2.4.2 Package webside

2.4.2.1 Class DBReader

java.lang.Object
webside.DBReader

```
public class DBReader
extends java.lang.Object
```

Support class for reading from the database.

Constructor Summary

Constructor and Description
`DBReader()`

Method Summary

Modifier and Type	Method and Description
`static WeatherData`	`getLastData()` Makes a query for extract the last record inserted.

static java.lang.String	**readDataGraph**(java.lang.String javascriptVariableName, java.lang.String columns, java.lang.String clause, java.lang.String order) Makes a query to the database and extract results.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DBReader

```
public DBReader()
```

Method Detail

readDataGraph

```
public static java.lang.String readDataGraph(
                java.lang.String javascriptVariableName,
                java.lang.String columns,
                java.lang.String clause,
                java.lang.String order)
```

Makes a query to the database and extract results.

Parameters:

javascriptVariableName - Name of the javascript variable
columns - Columns of the table to extract
clause - Where statement for the query
order - Order statement for the query

Returns:

A javascript matrix. Each element has a couple x, y where x is the date in unix format and y is the value of the columns.

getLastData

```
public static WeatherData getLastData()
```

Makes a query for extract the last record inserted.

Returns:

A WeatherData object where are stored all the information extracted from the last record.

2.4.2.2 Class Exporter

java.lang.Object
webside.Exporter

```
public class Exporter
extends java.lang.Object
```

Export data in a file using a codification class

Constructor Summary

Constructor and Description
`Exporter()`

Method Summary

Modifier and Type	Method and Description
`static void`	`GsonWeatherDataExport(WeatherData data, java.lang.String path)` Export the weather data taken once in a JSON file.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Exporter

```
public Exporter()
```

Method Detail

GsonWeatherDataExport

```
public static void GsonWeatherDataExport(WeatherData data,
java.lang.String path)
```

Export the weather data taken once in a JSON file.

Parameters:

```
data - WeatherData object where are stored all the information to
export
path - Path where saves the file
```

2.4.2.3 Class GsonWeatherDataDecorator

java.lang.Object
webside.GsonWeatherDataDecorator

```
public class GsonWeatherDataDecorator
extends java.lang.Object
```

Decorator class for WeatherData objects that decorate to JSON syntax using gson library.

Constructor Summary

Constructor and Description
`GsonWeatherDataDecorator()`

Method Summary

Modifier and Type	Method and Description
`static java.lang.String`	`json(WeatherData data)` Convert a WeatherData object in JSON.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GsonWeatherDataDecorator

```
public GsonWeatherDataDecorator()
```

Method Detail

json

```
public static java.lang.String json(WeatherData data)
```

Convert a WeatherData object in JSON.

Parameters:

data - WeatherData object to extract in JSON

Returns:

A string in JSON syntax with the array of the wind directions and the informations of a WeatherData object.

2.4.3 Package workers

2.4.3.1 Class ApplicationContextListener

java.lang.Object
    workers.ApplicationContextListener

All Implemented Interfaces:

java.util.EventListener, javax.servlet.ServletContextListener

---

```
public class ApplicationContextListener
extends java.lang.Object
implements javax.servlet.ServletContextListener
```

Servlet for parsing the Grappa weather site.

Constructor Summary

Constructor and Description
ApplicationContextListener()

Method Summary

Modifier and Type	Method and Description
void	contextDestroyed(javax.servlet.ServletContextEvent settings) It stops the servlet.
void	contextInitialized(javax.servlet.ServletContextEvent settings) Initialize the servlet.
javax.servlet.ServletCo	getServletContext()

`ntext`	Get the settings in the web.xml

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ApplicationContextListener

```
public ApplicationContextListener()
```

Method Detail

getServletContext

```
public javax.servlet.ServletContext getServletContext()
```

Get the settings in the web.xml

Returns:

`servlet context`

contextInitialized

```
public void contextInitialized(javax.servlet.ServletContextEvent setti
ngs)
```

Initialize the servlet.

Specified by:

`contextInitialized in interface javax.servlet.ServletContextListener`

Parameters:

`settings`

contextDestroyed

```
public void contextDestroyed(javax.servlet.ServletContextEvent
settings)
```

It stops the servlet.

Specified by:

`contextDestroyed in interface javax.servlet.ServletContextListener`

Parameters:

`settings`

2.4.3.2 Class DataWorker

java.lang.Object
workers.DataWorker

All Implemented Interfaces:

`java.lang.Runnable`

```
public class DataWorker
extends java.lang.Object
implements java.lang.Runnable
```

This class launch the thread for download the page and write the data parsed on the database.

Field Summary

Modifier and Type	Field and Description
protected javax.servlet.ServletContext	servletContext

Constructor Summary

Constructor and Description
DataWorker(javax.servlet.ServletContext servletContext) Constructor

Method Summary

Modifier and Type	Method and Description
void	run() Runner method for store the data of the weather in the database.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

servletContext

protected javax.servlet.ServletContext servletContext

Constructor Detail

DataWorker

public DataWorker(javax.servlet.ServletContext servletContext)

Constructor

Parameters:

servletContext - Servlet context where are stored the server settings

Method Detail

run

public void run()

Runner method for store the data of the weather in the database.

Specified by:

run in interface java.lang.Runnable

3 App android

L'app android è stata sviluppata usando un'activity Navigation Drawer. L'app è stata fatta per essere compatibile con android 4.0 e superiori, ed è stata testata sulle versioni 4.4.2 e 5.1.1. La prima schermata (fig. 3.1) mostra i dati dell'ultimo rilevamento, mentre navigando sul menù laterale (fig. 3.2) si può raggiungere la schermata per la visualizzazione dei grafici (fig 3.3).

Figura 3.1 - Home

Figura 3.2 - Menù laterale

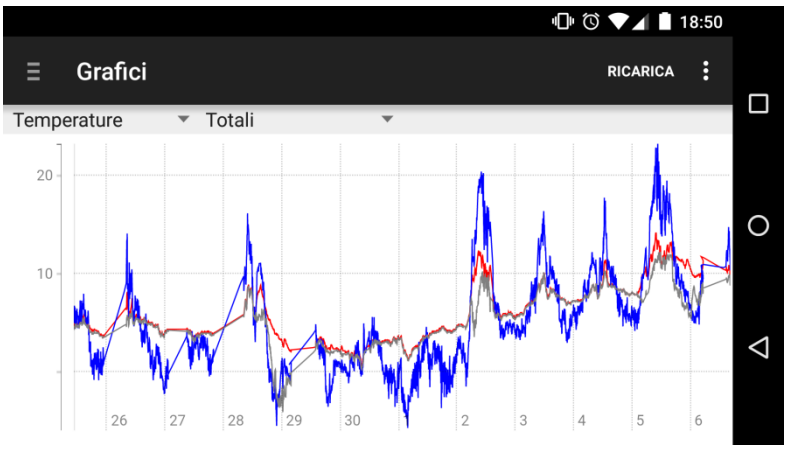


Figura 3.3 - Schermata dei grafici

3.1 Struttura

La struttura dell'app si basa in un'unica activity che contiene un LinearLayout con la schermata del grafico e un ScrollView con la home. In figura 3.4 si può osservare l'albero dei componenti grafici che compongono l'app.

- Device Screen
  - **drawer_layout** (CustomView) - android.support.v4.widget.DrawerLayout
    - **container** (FrameLayout)
      - **grafici** (LinearLayout) (vertical)
        - LinearLayout (horizontal)
          - **variable** (Spinner)
          - **type** (Spinner)
        - **graphBrowser** (WebView)
      - **home** (ScrollView)
        - LinearLayout (vertical)
          - **home_table** (TableLayout)
            - TableRow
              - **datetime_label** (TextView) - "Ultimo aggiornamento: "
              - **datetime_value** (TextView)
            - TableRow
            - TableRow
              - **condition_value** (TextView) - "Condizione"
            - TableRow
              - **temperature_label** (TextView) - "Temperatura"
              - **temperature_value** (TextView)
            - TableRow
              - **feel_temperature_label** (TextView) - "Temperatura percepita"
              - **feel_temperature_value** (TextView)
            - TableRow
              - **humidity_label** (TextView) - "Umidità"
              - **humidity_value** (TextView)
            - TableRow
              - **wind_label** (TextView) - "Velocità del vento"
              - TableLayout
                - TableRow
                - **wind_speed_value** (TextView)
                - **wind_direction_image** (ImageView) - @drawable/arrow
                - **wind_direction_value** (TextView)
            - TableRow
              - **pressure_label** (TextView) - "Pressione"
              - **pressure_value** (TextView)
            - TableRow
              - **solar_label** (TextView) - "Radiazione solare"
              - **solar_value** (TextView)
            - TableRow
              - **uv_label** (TextView) - "Radiazione UV"
              - **uv_value** (TextView)
            - TableRow
              - **dew_temperature_label** (TextView) - "Punto di rugiada"
              - **dew_temperature_value** (TextView)
            - TableRow
              - **rain_label** (TextView) - "Intesità pioggia"
              - **rain_value** (TextView)
            - TableRow
              - **snow_label** (TextView) - "Altezza neve"
              - **snow_value** (TextView)

Figura 3.4 - Albero dei componenti

Il codice si compone di 4 classi: ManageData, MeteoGrappa, NavigationDrawerFragment e ResourceDownloader. L'activity principale è lanciata dalla classe MeteoGrappa, alla creazione viene lanciato un thread AsyncTask per scaricare i dati dal file JSON sul server e vengono inseriti nella home. Dopodiché viene caricato su un oggetto WebView la pagina col grafico standard che poi potrà essere modificata grazie ai due Spinner posti immediatamente sopra, uno per scegliere il tipo di grafico e uno per scegliere il lasso di tempo da visualizzare. La modifica comporterà ad un nuovo caricamento della pagina. Queste operazioni sono contenute in un metodo refresh che viene richiamato anche quando si esegue un tap sull'opzione "Ricarica" nella barra superiore. È stato predisposto un menù con le impostazioni per una possibile implementazione futura che al momento non è legato ad alcuna azione.

3.2 Javadoc

3.2.1 Package com.alessandro.meteograppa

3.2.1.1 Class ManageData

java.lang.Object
AsyncTask
com.alessandro.meteograppa.ManageData

```
public class ManageData
extends AsyncTask
```

Download and set last weather data.

Constructor Summary	
Constructor and Description	
`ManageData()`	

Method Summary	
Modifier and Type	**Method and Description**
`protected java.lang.Object`	`doInBackground(java.lang.Object... arg0)`
Methods inherited from class java.lang.Object	
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`	

Constructor Detail

ManageData

```
public ManageData()
```

Method Detail

doInBackground

```
protected java.lang.Object doInBackground(java.lang.Object... arg0)
```

3.2.1.2 Class MeteoGrappa

java.lang.Object

ActionBarActivity
com.alessandro.meteograppa.MeteoGrappa

All Implemented Interfaces:
NavigationDrawerFragment.NavigationDrawerCallbacks

```
public class MeteoGrappa
extends ActionBarActivity
implements NavigationDrawerFragment.NavigationDrawerCallbacks
```

Main activity.

Nested Class Summary

Modifier and Type	Class and Description
static class	**MeteoGrappa.PlaceholderFragment** A placeholder fragment containing a simple view.

Field Summary

Modifier and Type	Field and Description
java.lang.String	**URL**

Constructor Summary

Constructor and Description
MeteoGrappa()

Method Summary

Modifier and Type	Method and Description
void	**addListenerOnSpinnerItemSelection**() Add a listener on the graph page spinners
void	**onConfigurationChanged**(Configuration newConfig)
protected void	**onCreate**(Bundle savedInstanceState)
boolean	**onCreateOptionsMenu**(Menu menu)
void	**onNavigationDrawerItemSelected**(int position) Called when an item in the navigation drawer is selected.
boolean	**onOptionsItemSelected**(MenuItem item)
void	**onSectionAttached**(int number)
void	**refresh**() Load the content data
void	**restoreActionBar**()
void	**setGraphUrl**(java.lang.String graphUrl)

void	**watch**(int position) Show the page selected from the link on the navigation tapped.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

URL

public final java.lang.String URL

See Also:

Constant Field Values

Constructor Detail

MeteoGrappa

public MeteoGrappa()

Method Detail

onCreate

protected void onCreate(Bundle savedInstanceState)

onNavigationDrawerItemSelected

public void onNavigationDrawerItemSelected(int position)

Description copied from interface: NavigationDrawerFragment.NavigationDrawerCallbacks

Called when an item in the navigation drawer is selected.

Specified by:

onNavigationDrawerItemSelected in interface NavigationDrawerFragment.NavigationDrawerCallbacks

onSectionAttached

public void onSectionAttached(int number)

restoreActionBar

public void restoreActionBar()

onCreateOptionsMenu

public boolean onCreateOptionsMenu(Menu menu)

onOptionsItemSelected

public boolean onOptionsItemSelected(MenuItem item)

onConfigurationChanged

public void onConfigurationChanged(Configuration newConfig)

refresh

public void refresh()

Load the content data

watch

```
public void watch(int position)
```

Show the page selected from the link on the navigation tapped.

Parameters:

`position - id of the navigation link tapped`

addListenerOnSpinnerItemSelection

```
public void addListenerOnSpinnerItemSelection()
```

Add a listener on the graph page spinners

setGraphUrl

```
public void setGraphUrl(java.lang.String graphUrl)
```

3.2.1.3 Class MeteoGrappa.PlaceholderFragment

java.lang.Object
    Fragment
        com.alessandro.meteograppa.MeteoGrappa.PlaceholderFragment

Enclosing class:
`MeteoGrappa`

```
public static class MeteoGrappa.PlaceholderFragment
extends Fragment
```

A placeholder fragment containing a simple view.

Constructor Summary

Constructor and Description
`PlaceholderFragment()`

Method Summary

Modifier and Type	Method and Description
`static MeteoGrappa.PlaceholderFragment`	`newInstance(int sectionNumber)` Returns a new instance of this fragment for the given section number.
`void`	`onAttach(Activity activity)`
`View`	`onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PlaceholderFragment

```
public PlaceholderFragment()
```

Method Detail

newInstance

```
public static MeteoGrappa.PlaceholderFragment newInstance(int
sectionNumber)
```

Returns a new instance of this fragment for the given section number.

onCreateView

```
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container,
                         Bundle savedInstanceState)
```

onAttach

```
public void onAttach(Activity activity)
```

3.2.1.4 Class NavigationDrawerFragment

java.lang.Object
    Fragment
        com.alessandro.meteograppa.NavigationDrawerFragment

```
public class NavigationDrawerFragment
extends Fragment
```

Fragment used for managing interactions for and presentation of a navigation drawer. See the design guidelines for a complete explanation of the behaviors implemented here.

Nested Class Summary

Modifier and Type	Class and Description
static interface	**NavigationDrawerFragment.NavigationDrawerCallbacks** Callbacks interface that all activities using this fragment must implement.

Constructor Summary

Constructor and Description
NavigationDrawerFragment()

Method Summary

Modifier and Type	Method and Description
boolean	**isDrawerOpen**()
void	**onActivityCreated**(Bundle savedInstanceState)
void	**onAttach**(Activity activity)
void	**onConfigurationChanged**(Configuration newConfig)

void	onCreate(Bundle savedInstanceState)
void	onCreateOptionsMenu(Menu menu, MenuInflater inflater)
View	onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
void	onDetach()
boolean	onOptionsItemSelected(MenuItem item)
void	onSaveInstanceState(Bundle outState)
void	setUp(int fragmentId, DrawerLayout drawerLayout) Users of this fragment must call this method to set up the navigation drawer interactions.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NavigationDrawerFragment

```
public NavigationDrawerFragment()
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

onActivityCreated

```
public void onActivityCreated(Bundle savedInstanceState)
```

onCreateView

```
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container,
                         Bundle savedInstanceState)
```

isDrawerOpen

```
public boolean isDrawerOpen()
```

setUp

```
public void setUp(int fragmentId, DrawerLayout drawerLayout)
```

Users of this fragment must call this method to set up the navigation drawer interactions.

Parameters:

fragmentId - The android:id of this fragment in its activity's layout.

drawerLayout - The DrawerLayout containing this fragment's UI.

onAttach

```
public void onAttach(Activity activity)
```

onDetach

```
public void onDetach()
```

onSaveInstanceState

```
public void onSaveInstanceState(Bundle outState)
```

onConfigurationChanged

```
public void onConfigurationChanged(Configuration newConfig)
```

onCreateOptionsMenu

```
public void onCreateOptionsMenu(Menu menu,
                                MenuInflater inflater)
```

onOptionsItemSelected

```
public boolean onOptionsItemSelected(MenuItem item)
```

3.2.1.5 Class ResourceDownloader

java.lang.Object
com.alessandro.meteograppa.ResourceDownloader

```
public class ResourceDownloader
extends java.lang.Object
```

This class downloads the content of a web page.

Constructor Summary

Constructor and Description
`ResourceDownloader(java.lang.String link)` It downloads the web page and store in a String variable.

Method Summary

Modifier and Type	Method and Description
`java.lang.String`	`getPage()`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ResourceDownloader

```
public ResourceDownloader(java.lang.String link)
```

It downloads the web page and store in a String variable.

Parameters:

`link - url of the page to download`

Method Detail

getPage

```
public java.lang.String getPage()
```