# Elaborato Fondamenti di Informatica - Consumi Idrici

## Analisi dei requisiti

Lo scopo del programma è quello di analizzare i consumi idrici di un gruppo di utenti relativi esclusivamente all'anno 2015 in maniera interattiva. I dati sono forniti in un file formato .csv strutturato in questo modo: ogni riga del file corrisponde ad una lettura di un contatore, ciascuna riga riporta in ordine data, valore e codice utente della lettura separati da una virgola.

* La data è in formato "yyyy-MM-dd hh:mm:ss"
* Il valore della lettura è un numero decimale progressivo che indica quanti metri cubi sono stati consumati da quell'utente a quella data
* Il codice utente è un identificativo alfanumerico univoco che si riferisce ad una utenza domestica.

Non è garantito alcun ordine nell'insieme dei dati e non è garantita nessuna continuità cronologica: ogni utente può avere un numero variabile di letture e tra lettura successive può essere passata una quantità di tempo variabile.

Il programma deve fornire tre funzionalità principali: visualizzazione dei consumi, interrogazione dei dati ed analisi dei consumi.

La visualizzazione dei consumi permette, inserito un codice cliente, di verificare il consumo totale dello stesso, ossia il valore dell'ultima lettura in ordine cronologico. Inoltre fornisce un istogramma dei consumi. Tale grafico ha quattro modalità di visualizzazione:

* Annuale: il grafico mostra le 12 barre dei consumi di ciascun mese dell'anno 2015
* Mensile per giorni: il grafico mostra da 28 a 31 barre relative ai consumi di ciascun giorno di un mese scelto dall'utente
* Mensile per settimane: il grafico mostra da 4 a 6 barre relative ai consumi delle settimane di un mese scelto dall'utente. Le settimane si considerano da lunedì a domenica e possono ricadere parzialmente all'interno del mese stesso
* Giornaliero: il grafico mostra i 24 consumi orari di una data scelta dall'utente.

in ogni caso il grafico deve indicare il valore minimo, medio e massimo.

L'interrogazione dei dati è una funzionalità del programma che deve richiedere all'utente un codice identificativo, una data di inizio ed una di fine periodo. In base all'input deve mostrare il consumo dell'utente cercato all'interno del periodo ed i valori medi di consumo orario, giornaliero, settimanale e mensile. Devono essere indicati solo i dati ritenuti validi in base all'estensione del periodo.

L'analisi dei consumi, al contrario delle altre sezioni, riguarda l'intero insieme di utenze e non una singola, e si suddivide in due parti. La prima deve mostrare un elenco di utenti che si ritenga possano avere perdite, ossia utenti con consumi superiori ad una certa soglia in orari notturni (compresi tra le 00:00 e le 05:00). Per ogni utente devono essere indicati i periodi in cui tali consumi sospetti si sono verificati. La seconda parte è quella che ricerca le utenze devianti, ossia un elenco di tutte le utenze con consumi medi superiori di più del doppio rispetto alla media di tutte le utenze in un certo intervallo di tempo.

## Progettazione della soluzione

Il programma è strutturato in una sola finestra suddivisa in tre tab, una per ogni funzionalità principale richiesta. Nella parte alta della finestra è presente un pulsante che permette di selezionare un file .csv. Una volta caricato il file, solo se ciò è avvenuto correttamente, viene abilitato il resto dell'interfaccia. In ogni tab è presente una parte in cui l'utente inserisce i dati necessari ed una in cui vengono mostrate le informazioni richieste. Nella tab di visualizzazione l'output è l'istogramma dei consumi, i consumi medi, il consumo totale e la data di ultimo aggiornamento e dipende da tre input: codice utente, modalità del grafico e data. Nella parte di interrogazione, dati un codice utente e le date di inizio e fine del periodo, si possono visualizzare il consumo totale nel periodo indicato ed i valori medi. La parte di analisi è suddivisa in due sezioni totalmente indipendenti. La prima, data in input una threshold, riempie una lista con l'elenco degli utenti con perdite notturne e, selezionato uno di questi utenti, mostra una tabella con i relativi consumi anomali. La seconda, invece, dato un periodo di tempo, mostra un elenco di tutti gli utenti che abbiano consumi superiori al doppio della media. Per ogni utente sono elencati codice identificativo e consumo medio giornaliero, settimanale e mensile.

### Classi e strutture dati

Il programma è realizzato secondo il paradigma della programmazione a oggetti. Le classi realizzate sono:

* MainWindow: classe standard per la gestione dell'interfaccia grafica. Essendo il programma formato da una sola finestra è l'unica classe di questo tipo. Contiene tutte le interazioni con l'utente gestite in base agli eventi dell'interfaccia e richiama gli altri oggetti per la manipolazione e la lettura dei dati. Il suo metodo show() viene invocato dal main.
* InputFile: oggetto che rappresenta il file contenente i dati. Il suo metodo principale read() consente di estrapolare dal file i dati in una struttura ideale per essere analizzati facilmente.
* Consumption: rappresenta una lettura di un contatore indipendentemente dall'utente cui viene associata. La parte privata contiene quindi solo data e valore della lettura. Sono implementati gli operatori di (dis)uguaglianza e ordinamento. Garantisce la correttezza dei dati.
* ConsumptionSet: basato sulla std::set, rappresenta una collezione di consumption ordinate. Gestisce l'inserimento delle consumption nella collezione, garantendo la coerenza tra tutti i dati. Permette inoltre di calcolare i dati necessari alla realizzazione del programma.
* Plot: rappresenta il grafico della parte di visualizzazione. Permette di disegnare e cancellare l'istogramma e di ottenere i valori minimo, medio e massimo.
* AvgTableModel e LeaksTableModel: derivano dalla classe QAbstractModel e sono i modelli delle due tabelle che visualizzano gli utenti con perdite e devianti.

La struttura dati più importante del programma è quella che contiene tutti i dati ricavati dal file. È una std::map<QString, ConsumptionSet> che associa ad ogni codice utente (QString) l'elenco di tutte le sue letture (ConsumptionSet). Questa soluzione semplifica la rappresentazione dei dati perché elimina ogni informazione non necessaria (il codice utente ripetuto ad ogni lettura), li organizza in maniera strutturata e ne velocizza l'accesso perché le chiavi (codici utente) sono ordinate alfabeticamente e gli oggetti a cui si riferiscono (ConsumptionSet) contengono al loro interno le letture ordinate cronologicamente. La ConsumptionSet sfrutta tutti i vantaggi della std::set, in particolare l'ordinamento ed il rapido accesso, estendendo le sue funzioni con quelle di calcolo dei consumi.

### Algoritmi

***InputFile::read() : legge il file e produce una std::map contenente i dati***

Legge il file csv una riga alla volta, da cui ricava i tre parametri data, valore e codice utente. Utilizza i primi due per costruire un oggetto Consumption ed aggiunge quest'ultimo alla ConsumptionSet indicizzata nella mappa dalla chiave uguale al codice utente. Robusto ad errori causati da dati errati, può ignorare righe invalide chiedendo conferma all'utente.

***ConsumptionSet::getLast() : restituisce il consumo totale dell'utente e la data di ultimo aggiornamento***

Sfruttando l'ordinamento dei dati già descritto, il consumo totale e la sua data di lettura sono semplicemente l'ultima Consumption nella collezione di dati.

***ConsumptionSet::getConsAtDate(date) : restituisce il consumo in un momento preciso***

I dati non garantiscono una continuità ed indicano il consumo totale solo in alcuni momenti. Nei periodi compresi tra due letture l'andamento del consumo è ignoto e viene quindi considerato lineare. Perciò, il consumo in un determinato momento è dato da una proporzione. Si cercano l'ultima registrazione precedente alla data richiesta e la prima successiva e la formula per calcolarlo è:

= consumo ad una data

= differenza di tempo tra le registrazioni più vicine

= differenza di consumi tra le registrazioni più vicine

= differenza di tempo tra la data cercata e la registrazione precedente più vicina

C = consumo alla registrazione precedente più vicina

Nel caso in cui non sia presente una registrazione successiva, significa che il consumo è rimasto invariato dall'ultima registrazione. Il consumo alla data richiesta è quindi lo stesso di quello dell'ultima lettura. Nel caso in cui manchi una lettura precedente vale il discorso inverso.

***ConsumptionSet::getPeriodConsumption(firstDate, lastDate) : restituisce il consumo totale in un periodo***

È semplicemente la differenza tra il consumo totale alla seconda data e il consumo totale alla prima data. Grazie ai controlli sull'inserimento dei dati non può risultare negativo.

***ConsumptionSet::getHistogramData(begin, end, step) : restituisce i valori delle barre del grafico***

Richiede un periodo di tempo ed uno step tra un intervallo e l'altro.

Per ogni intervallo calcola la differenza di consumi tra il suo inizio e la sua fine. Ad ogni iterazione il consumo alla fine di un intervallo diventa quello all'inizio dell'intervallo successivo. Ripete finché la data di inizio incrementata ogni volta non raggiunge la data di fine. L'insieme dei consumi dei vari intervalli sono i valori delle barre del grafico.

***ConsumptionSet::getNightLeaks(threshold) : restituisce l'insieme di consumi notturni superiori alla soglia***

Per ogni giorno del 2015 calcola il consumo nel periodo tra le 00:00 e le 05:00. Se tale consumo è superiore alla threshold lo aggiunge (insieme alla data di riferimento) alla lista delle possibili perdite notturne.

## Note di realizzazione: aspetti particolari della codifica

***Conversione in CET (Central European Time)***

Il tipo scelto per rappresentare le date e ore è il QDateTime. Durante le lettura del file, però, questo causa problemi in quanto esistono letture dei contatori nelle notti in cui avviene il cambio ora legale-solare e ciò risulta in oggetti QDateTime invalidi. Per risolvere questo problema è necessario convertire il tutto in UTC tramite il metodo apposito e poi tornare in CET tenendo però conto dell'ora legale. Alle date UTC vengono quindi aggiunte due ore nel caso di date estive (comprese tra l'ultima domenica di marzo e l'ultima di ottobre) o un'ora sola nel caso di date invernali.

***ConsumptionSet::insert ( cons )***

Una ConsumptionSet è sostanzialmente una std::set<Consumption> e pertanto una collezione ordinata di Consumption. L'ordinamento tra Consumption è definito tramite l'operatore < il quale stabilisce che una Consumption è minore di un'altra se lo è la sua data. In caso di date uguali l'ordinamento viene stabilito dal valore. Può accadere che un oggetto Consumption abbia data inferiore a quella di un altro ma consumo maggiore. Perciò è necessario escludere questi casi quando viene aggiunta una nuova Consumption ad una ConsumptionSet. Questi controlli sono implementati nella ConsumptionSet::insert() e, nel caso in cui non vengano superati, la nuova Consumption non viene aggiunta. In caso di dati incoerenti vengono quindi mantenuti quelli inseriti per primi. Nei dati del file consumption\_all.csv questo avviene più volte anche in utenti diversi.

Qualche esempio in cui una lettura ha un consumo inferiore rispetto alla lettura precedente e/o superiore rispetto a quella successiva:

*"2015-05-28 20:00:00",45.099,16*

*"2015-05-28 21:00:00",44.739,16*

*"2015-05-28 22:00:00",45.099,16*

*"2015-06-10 20:00:00",54.893,16*

*"2015-06-10 21:00:00",54.535,16*

*"2015-06-10 22:00:00",54.893,16*

*"2015-06-18 20:00:00",57.356,16*

*"2015-06-18 21:00:00",56.988,16*

*"2015-06-18 22:00:00",57.356,16*

*"2015-06-25 20:00:00",62.356,16*

*"2015-06-25 21:00:00",61.995,16*

*"2015-06-25 22:00:00",62.356,16*

*"2015-06-30 20:00:00",66.695,16*

*"2015-06-30 21:00:00",66.341,16*

*"2015-06-30 22:00:00",66.695,16*

*"2015-02-04 08:25:00",23.445,112*

*"2015-02-04 08:28:00",9.726,112*

*"2015-04-09 09:00:00",20.331,206*

*"2015-04-09 10:00:00",20.330,206*

***Visualizzazione grafico per settimane***

Il metodo ConsumptionSet::getHistogramData() funziona solo se il periodo considerato è formato da un numero intero di intervalli (cioè all'inizio del periodo inizia anche il primo intervallo e alla fine del periodo finisce l'ultimo intervallo). Questo non crea problemi nella visualizzazione annuale (12 mesi, il primo giorno dell'anno è il primo di Gennaio e l'ultimo è il 31 Dicembre), in quella mensile per giorni (28-31 giorni) e in quella giornaliera (24 ore). Nella visualizzazione settimanale, però, può capitare che la prima e/o l'ultima settimana siano comprese solo parzialmente nel mese scelto. Per questo motivo quando viene scelta la visualizzazione settimanale si tratta in realtà di una visualizzazione del mese a giorni, i cui valori vengono però accorpati da lunedì (o dal primo giorno del mese per la prima settimana) a domenica (o all'ultimo giorno del mese per l'ultima). L'algoritmo che fa ciò è molto semplice: una volta ottenuti i dati della visualizzazione del mese per giorni, li scorre tutti e li somma fino a quando non raggiunge una domenica o la fine del mese; a quel punto memorizza la somma in un altro vettore, la azzera e ricomincia dal lunedì.

***Interrogazione - periodi non definiti***

Nella parte di interrogazione vengono mostrati i valori medi di consumo orario, giornaliero, settimanale e mensile di un determinato utente in un determinato periodo.

Il periodo è indicato dall'utente tramite due date, entrambe comprese nello stesso, e viene considerato partire dalle 00:00 della prima data e finire alle 23:59 della seconda. Per questo motivo è possibile inserire la stessa data nei due controlli: il periodo in questione sarà il giorno indicato. Non essendo possibile, invece, scegliere periodi di durata inferiore alle 24 ore, sia il consumo medio orario sia quello giornaliero sono sempre definiti. Il consumo settimanale è considerato definito quando tra le due date è passata almeno una settimana. Ciò si verifica in due casi: le due date appartengono alla stessa settimana e sono rispettivamente lunedì e domenica (il periodo in questione è una settimana esatta), oppure le due date appartengono a due settimane diverse (può esserci un numero variabile di settimane intere comprese nel mezzo, anche 0, e le settimane a cui appartengono le due date possono essere incluse solo parzialmente). Per il consumo mensile vale lo stesso discorso: è definito quando le date appartengono a mesi diversi o quando appartengono allo stesso mese e siano rispettivamente il primo e l'ultimo giorno del mese.

***Utenze devianti - consumo medio***

Nella ricerca delle utenze devianti si chiede di riportare le utenze con valor medio del consumo mensile, settimanale o giornaliero superiore di più del doppio rispetto alla media dell'intero insieme di utenze. Dato che i valori medi mensile, settimanale e giornaliero sono calcolati dividendo lo stesso consumo totale per il numero di mesi, settimane e giorni del periodo, questi sono multipli tra loro. Perciò non ha senso confrontare consumo medio giornaliero, settimanale e mensile di un utente con la media giornaliera, settimanale e mensile dell'intero insieme, ma è sufficiente confrontarne uno solo di questi, o addirittura confrontare i due consumi totali. Una volta trovate le utenze devianti si è comunque scelto di mostrare nella relativa tabella i valori medi giornalieri, settimanali e mensili.

***Utenze devianti - periodo***

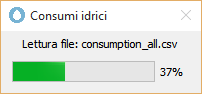
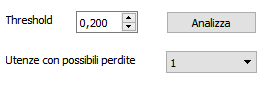
Essendo una ricerca effettuata sui valori medi all'interno di un periodo, non ha senso parlare di "tutti i periodi in cui il consumo ha superato il doppio della media" perché l'unico periodo in cui ciò è avvenuto è quello in cui si è cercato.

***Classe Plot***

Il costruttore della classe Plot richiede solo un puntatore ad un QCustomPlot. Dato che i costruttori devono verificare la correttezza dei dati, bisogna controllare che non sia un nullptr. In questo caso sarebbe possibile continuare con l'elaborazione mettendo l'oggetto in uno stato di inattività, ma la soluzione migliore è quella di lanciare un'eccezione[[1]](#footnote-1). In questo modo l'unico metodo per costruire un oggetto Plot è avere a disposizione un puntatore a QCustomPlot. Questo puntatore esiste solo dopo che l'interfaccia viene inizializzata e ciò avviene all'interno del costruttore di MainWindow. La conseguenza è che non è possibile costruire Plot con una lista di inizializzazione prima del costruttore di MainWindow, ma va fatto al suo interno. Non essendo inoltre possibile implementare un costruttore di default per Plot, l'oggetto andrà allocato dinamicamente. Nel distruttore ~MainWindow() Plot viene deallocato.

***Progress bar***

A causa della grande quantità di dati le operazioni di lettura del file e di analisi dei consumi si rivelano piuttosto lunghe. Per questo si è scelto di mostrarne l'avanzamento tramite delle progress bar, che evitano anche l'effetto di maschera bloccata. Generalmente, quando l'utente cambia uno dei dati di input, l'output cambia in tempo reale. Ad esempio nella tab di visualizzazione, mentre l'utente scrive il codice "111", verranno mostrati i grafici dell'utente 1, poi dell'utente 11 ed infine quello del 111. Ciò è possibile grazie alla velocità con cui vengono elaborate le informazioni. Nel caso dell'analisi dei dati, invece, non è possibile aggiornare immediatamente l'elenco degli utenti con perdite quando viene cambiato il valore della threshold. Per questo motivo è stato necessario aggiungere un pulsante di avvio analisi.

***Utente NULL***

Nei dati sono presenti alcune letture del 7 e 8 luglio riferite ad un utente NULL. Essendo i codici utente alfanumerici non c'è alcun motivo per escludere "NULL" (teoricamente potrebbe essere un codice valido), anche se in questo caso è evidente che si siano verificati errori nella raccolta dei dati. Le letture di questo utente sono quindi diventate un'ottima prova per verificare la correttezza dei vari controlli implementati. Infatti i valori (probabilmente riferiti in origine ad utenti distinti) sono molto diversi tra loro ed hanno un andamento quasi casuale. Vengono quindi correttamente scartate tutte le letture che genererebbero consumi negativi.

***Rispetto paradigma***

Si è cercato di rispettare ogni regola di buona programmazione:

* Le classi sono strutturate in maniera corretta
* Sono state usate le liste di inizializzazione
* Gli errori non sono comunicati con valori di ritorno particolari ma con eccezioni, le quali vengono intercettate e gestite. La stessa cosa viene fatta nel costruttore della classe Consumption nel caso in cui i dati non siano validi.
* Avendo definito gli operatori == e < per la classe Consumption, sono stati definiti anche !=, <=, > e >=
* Si è evitato di causare memory leak. L'unica volta che viene allocata memoria dinamica viene fatto in un costruttore (new) e nel corrispettivo distruttore la memoria viene deallocata (delete)
* È rispettata la const correctness

## Note di realizzazione: librerie utilizzate

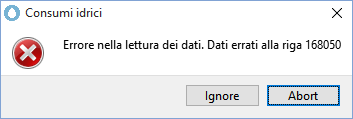
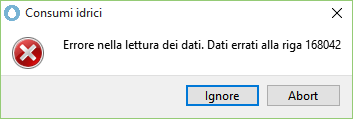
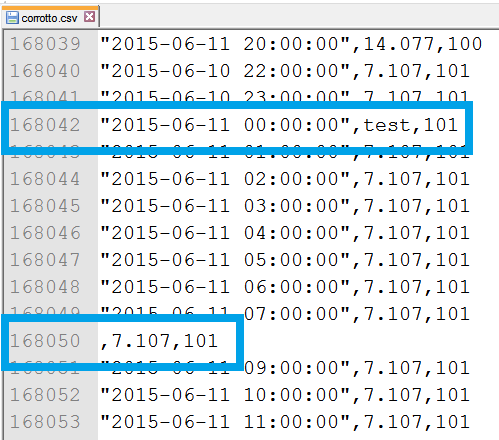
***QCustomPlot***

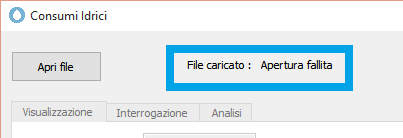
QCustomPlot[[2]](#footnote-2) è l'unica libreria esterna utilizzata e fornisce un widget con cui è possibile disegnare grafici. L'utilizzo è molto semplice, una volta aggiunti al progetto i file qcustomplot.cpp e qcustomplot.h è possibile promuovere un widget a qcustomplot. Questa libreria è utilizzata dalla classe plot, i cui metodi principali sono draw() e clear(). Questi sfruttano la libreria per disegnare e cancellare il grafico, gestendo assi, griglia, valori sugli assi, colori, dati, ed ogni aspetto necessario a visualizzare l'istogramma.

## Test

***Apertura file corrotto***

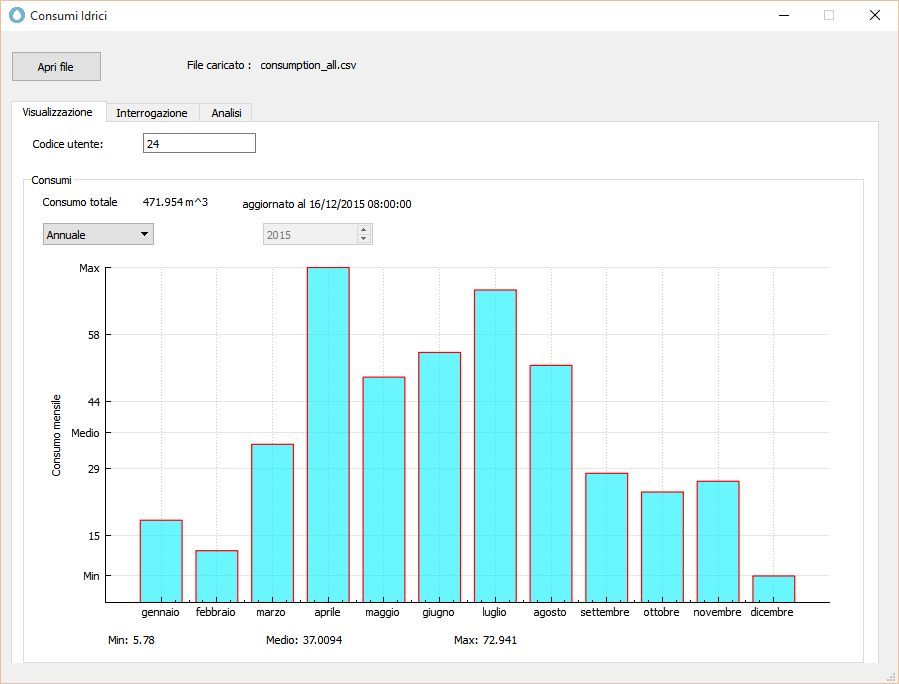
La lettura del file è in grado di rilevare errori nella formattazione o nel valore dei dati. Se succede il programma chiede all'utente se ignorare l'errore (la riga indicata verrà saltata) o abortire l'apertura del file corrotto. In questo esempio il file contiene due righe con errori diversi [1], alla riga 168042 è stato detto di ignorare l'errore [2], la lettura è continuata correttamente fino alla riga 168050, quando si è detto di abortire la lettura [3]. Il problema segnala correttamente "Apertura fallita" [4].

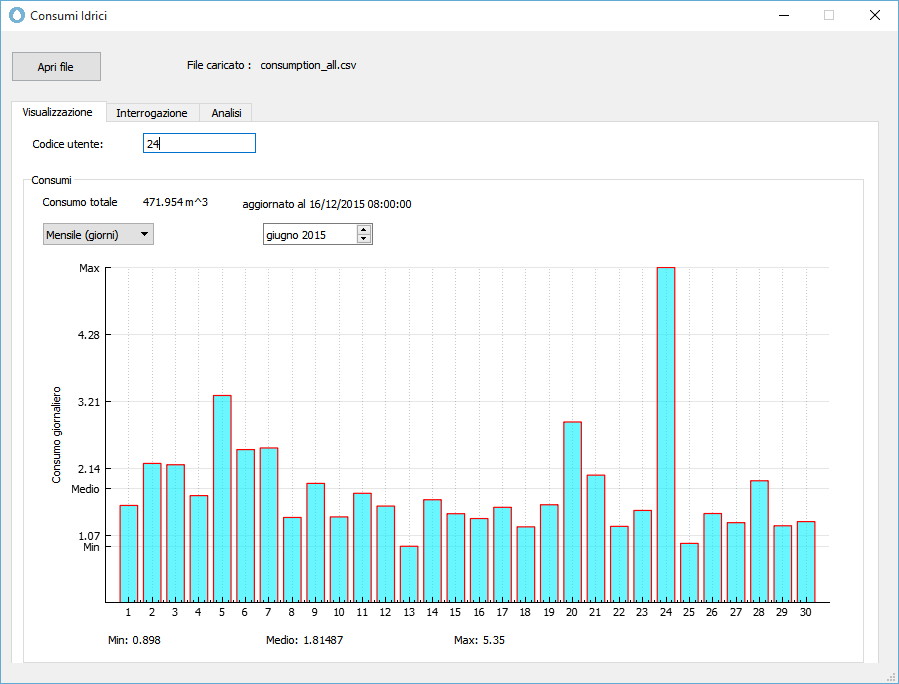
******

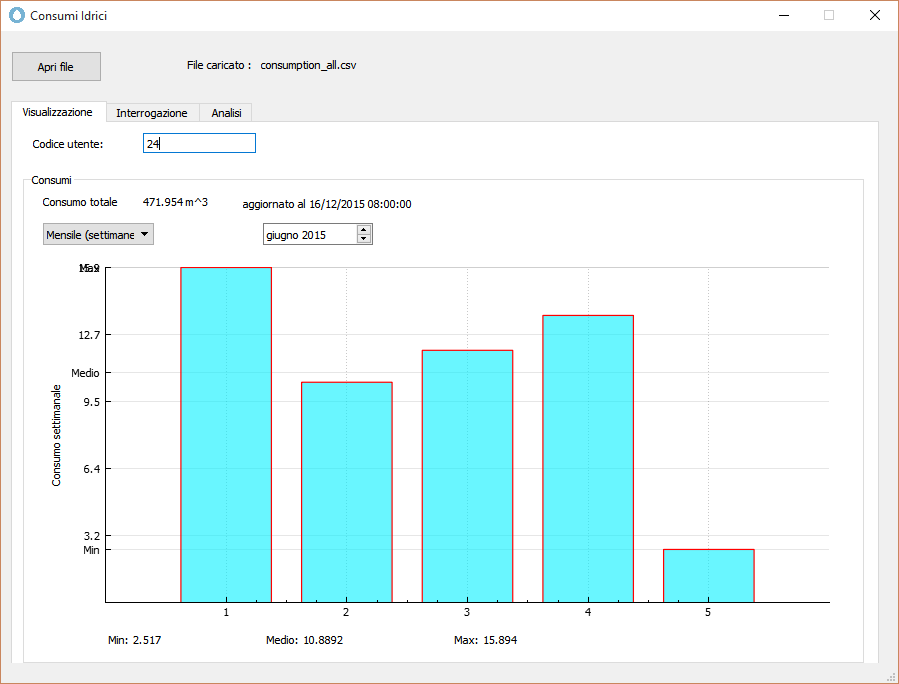
******

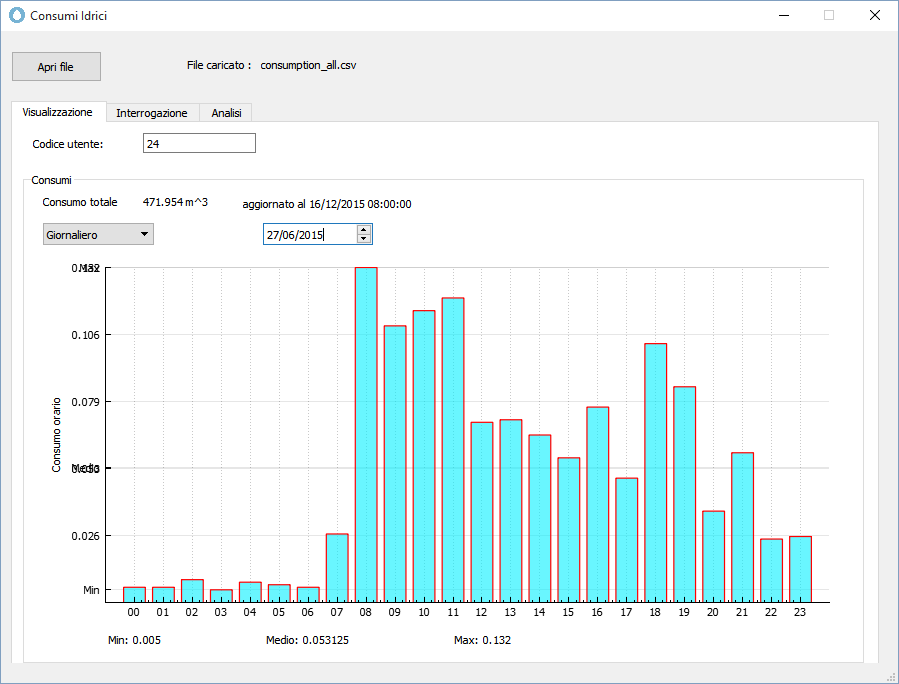
***Visualizzazione consumi***

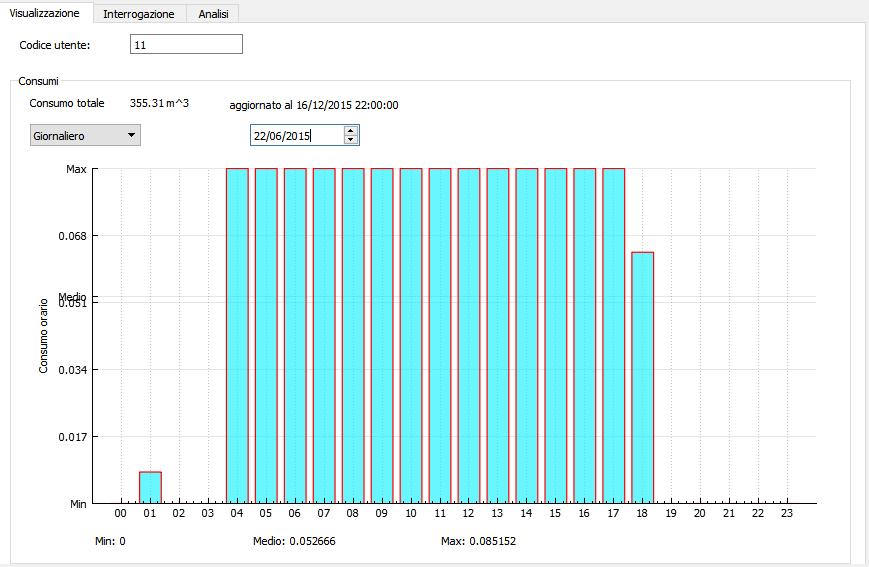
Le diverse modalità di visualizzazione dell'istogramma

******

******

******

******

***Visualizzazione consumi - casi particolari***

*"2015-06-21 23:00:00",148.282,11*

*"2015-06-22 01:00:00",148.282,11*

*"2015-06-22 01:05:00",148.290,11*

*"2015-06-22 02:00:00",148.290,11*

*"2015-06-22 03:05:00",148.290,11*

*"2015-06-22 04:00:00",148.290,11*

*"2015-06-22 18:45:00",149.546,11*

*"2015-06-22 19:00:00",149.546,11*

*"2015-06-22 20:45:00",149.546,11*

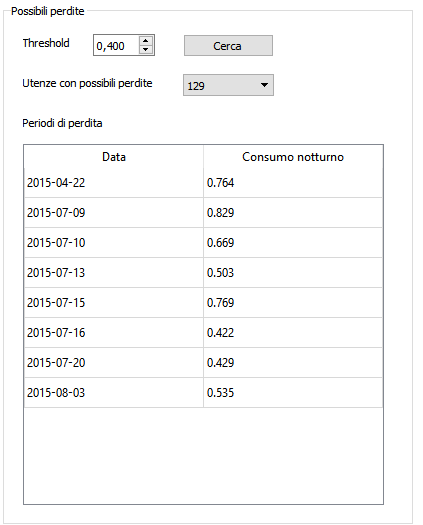
In questo caso particolare si notano benissimo il consumo all'una di notte (passaggio 148.282-148.290), la mancanza dei dati dalle 04 alle 18:45 che si traduce in un consumo costante nelle ore del giorno (la differenza di consumo tra la sera e la mattina viene ripartita equamente tra le ore comprese tra le due letture). Il fatto che la colonna delle 18 sia 3/4 delle altre è dovuto all'ora della lettura, le ore 18:45. Infine si vede che la mancanza di consumi nel resto della giornata è dovuta al fatto che il valore rimane invariato per alcune ore.

***Interrogazione***

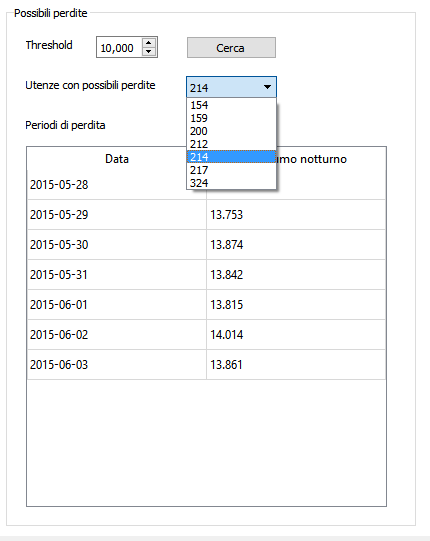
Calcolo dei consumi medi in periodi diversi

|  |  |
| --- | --- |
| Tutto l'anno come periodo: tutti i consumi sono definiti | Un solo giorno: sono definiti solo consumo orario e giornaliero. Quest'ultimo è pari al consumo nel periodo |
| Una settimana intera da lunedì a domenica: il consumo settimanale equivale al consumo nel periodo | Da sabato a mercoledì: non è compresa una settimana intera, ma i giorni appartengono a due settimane diverse. Il consumo settimanale è metà del consumo nel periodo |
| Un mese completo: consumo mensile e del periodo sono identici | Più mesi |

***Analisi dei consumi - perdite notturne***

******

Tutte le notti in cui l'utente 129 ha consumato più di 0,400 m3 d'acqua



I pochi utenti che hanno consumato almeno 10 m3 d'acqua per almeno una notte.

Nella tabella sono presenti i consumi dell'utente 214 ed i dati confermano l'utilizzo anomalo:

*"2015-06-02 00:00:00",425.847,214*

*"2015-06-02 01:00:00",428.58,214*

*"2015-06-02 02:00:00",431.426,214*

*"2015-06-02 03:00:00",434.307,214*

*"2015-06-02 04:00:00",437.085,214*

*"2015-06-02 05:00:00",439.861,214*

***Analisi dei consumi - utenze devianti***

|  |  |
| --- | --- |
|  |  |
| Gli utenti che hanno consumo medio superiore al doppio della media nell'arco dell'intero anno. In basso è presente il consumo medio di tutte le utenze. | Utenti con consumo superiore al doppio della media nel mese di Agosto |

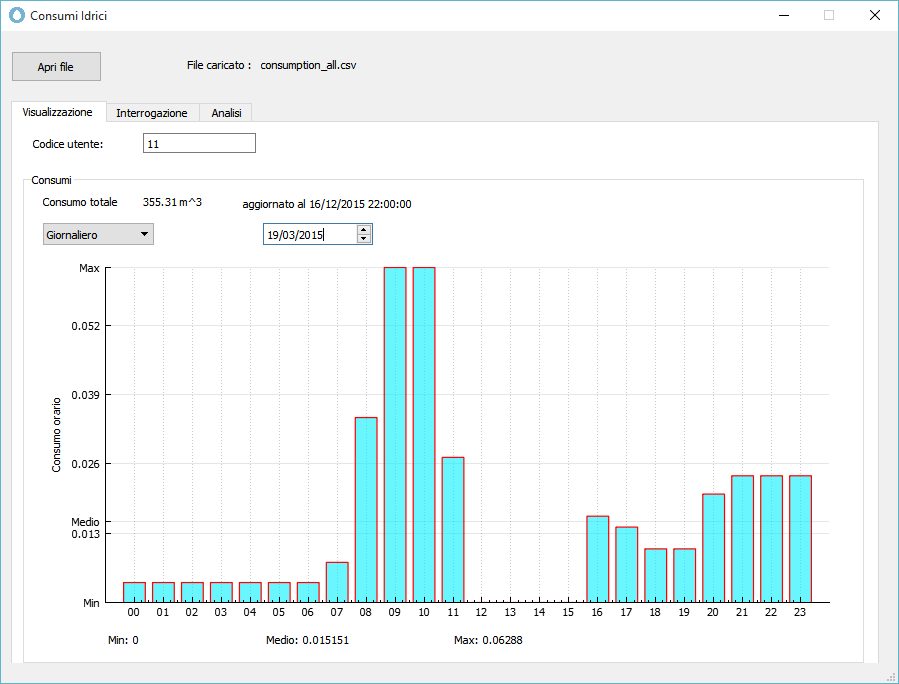
***Calcolo consumo in un dato momento con la proporzione***

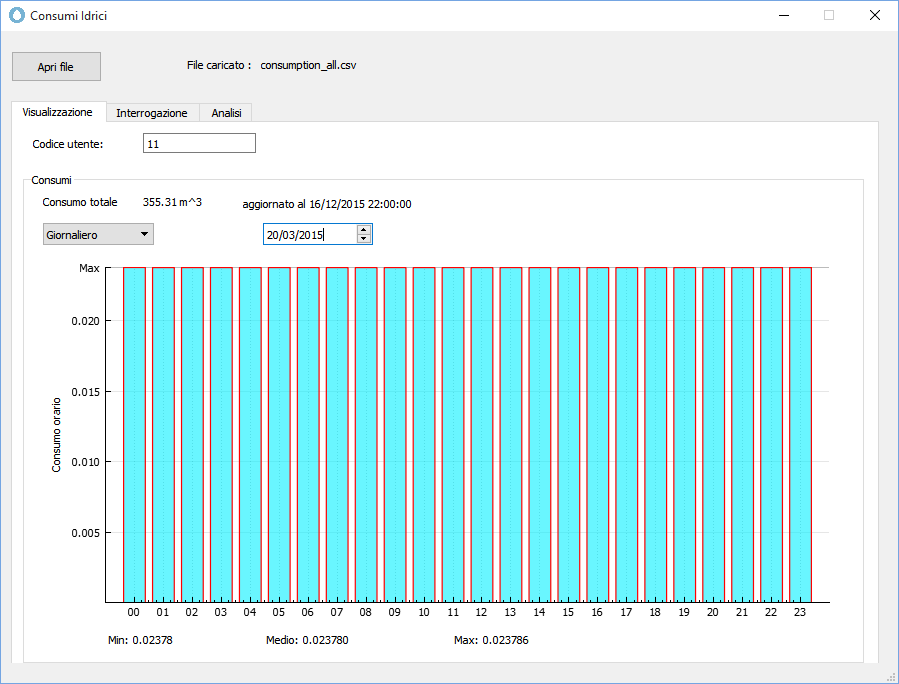
Tra queste letture dell'utente 11 non ne sono presenti altre

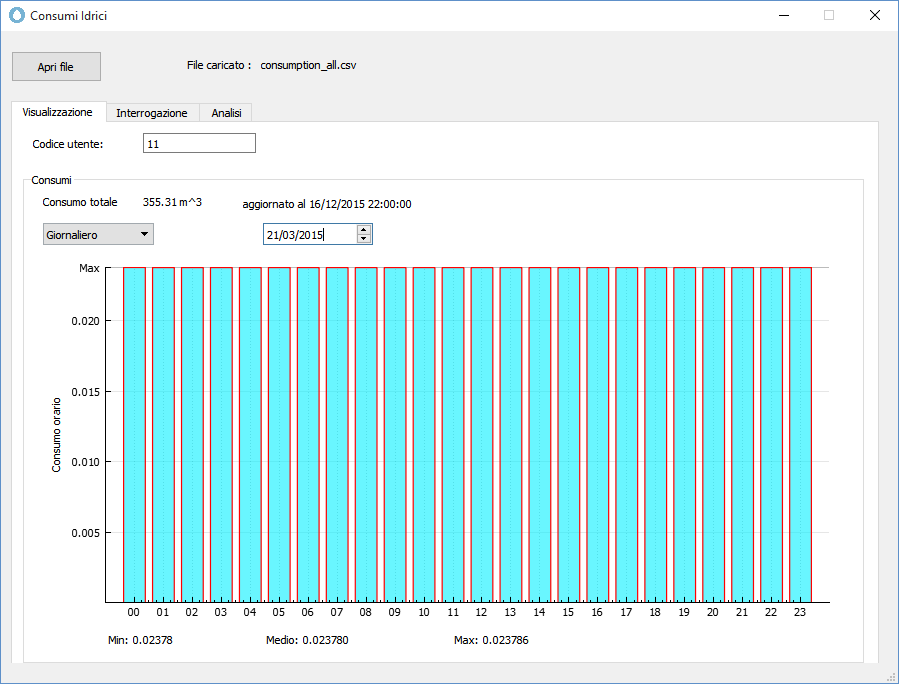
"2015-03-19 20:15:00",55.432,11

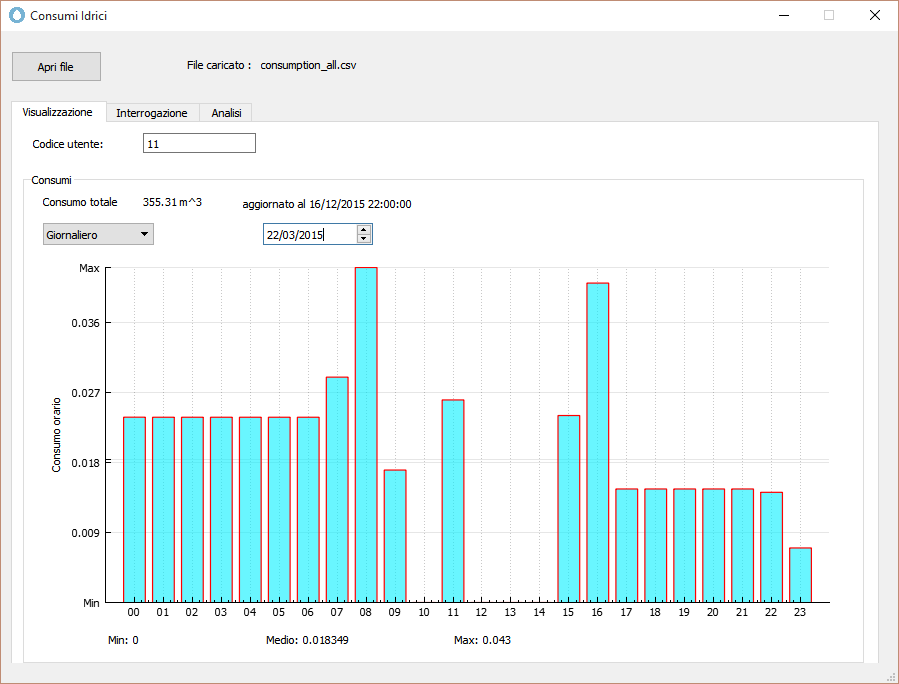
"2015-03-22 07:20:00",56.837,11

Il consumo dato dalla differenza tra le due è quindi suddiviso equamente in ogni momento compreso tra la sera del 19/03 e la mattina del 22/03. Questo si nota molto bene nel grafico, dove il consumo rimane costante in questo periodo di tempo. In particolare, le ore 20 del 19/03 e le ore 07 del 22/03 hanno consumi diversi perché il loro valore dipende anche dai consumi registrati tra le 20:00 e le 20:15 e da quelli registrati dalle 07:20 alle 8:00.









1. https://isocpp.org/wiki/faq/exceptions#ctors-can-throw [↑](#footnote-ref-1)
2. http://www.qcustomplot.com/ [↑](#footnote-ref-2)