

Università di Pisa  
  
  
TASK 1  
LARGE-SCALE AND MULTI-STRUCTURED DATABASES

**Documentazione del progetto “*PisaFlix”***  
Anno Accademico 2019-2020  
  
  
  
  
stefano petrocchi, Andrea Tubak, Francesco Ronchieri, Alsessandro Madonna

Sommario

[Analysis Document 3](#_Toc23335307)

[Introduction 3](#_Toc23335308)

[Requirements 3](#_Toc23335309)

[Functional 3](#_Toc23335310)

[Non-Functional 3](#_Toc23335311)

[Software Architecture 3](#_Toc23335312)

[Use Cases Diagram 4](#_Toc23335313)

[Class DIagram 4](#_Toc23335314)

[Database Main Entities Diagram 4](#_Toc23335315)

[Documento di Progetto 5](#_Toc23335316)

[Architettura del Software 5](#_Toc23335317)

[Responsabilità Classi 5](#_Toc23335318)

[Back-end 5](#_Toc23335319)

[Middleware 6](#_Toc23335320)

[Front-end 6](#_Toc23335321)

[Documento di Collaudo 7](#_Toc23335322)

[Schermata dell’Applicazione 7](#_Toc23335323)

[Elementi di Back-End 7](#_Toc23335324)

[File di configurazione Locale in Xml 7](#_Toc23335325)

[Archivio 8](#_Toc23335326)

[File di Log Remoto in Xml 9](#_Toc23335327)

[Casi di Test 9](#_Toc23335328)

[Inserimento Nuovo Affitto 9](#_Toc23335329)

[Visualizzazzione Registro e Statistiche degli Affitti 10](#_Toc23335330)

[Rimozione di una Prenotazione 10](#_Toc23335331)

[Informazioni Aggiuntive Statistiche 10](#_Toc23335332)

# Analysis Document

## Introduction

Have you ever found yourself in a gloomy day? Everyone is at home, no one knows what to do and time seems to slow down. That’s the perfect time for a movie! If you live within the Pisan suburb and you want to enjoy the best experience, PisaFlix is what you need.

PisaFlix is a platform in which you’ll find all of the information regarding movies and cinemas in the Pisa area. It gives you the possibility to know which cinema is available, which film you could watch and at what time all of the projections are due. PisaFlix has also a rating/comment system both for cinemas and movies. This allows people to express their opinion, and, by doing so, providing others some really valuable information. Everyone who’s still unsure about what to do next will receive a great deal of help by this functionality. We believe PisaFlix offers a complete package of services, that will have a huge impact on the quality of the decisions made by our customers. Proving you everything you need to have a well informed choice is not only our goal, but also a pleasure.

## Requirements

### Functional

1. *Users* can **view** the list of **Movies/Cinemas** available on the platform.
2. *Users* can **view** the specific information about a *Movie* (es. category, publish date ecc…).
3. *Users* can **view** the specific information about a *Cinema* (es. Name, Address).
4. *Users* can **view** the *Projections* scheduled in a *Cinema*.
5. *Users* can **view** the *Projections* scheduled for a *Film*.
6. *Users* can **register** an account on the platform.
7. *Users* can **log in** on the platform in order to do some specific operations:
   1. If logged a *User* can **add to favorite** a *Movie/Cinema*.
   2. If logged a *User* can **comment** a *Movie*/*Cinema* and express his opinion about it.
   3. If logged a *User* can **change** a *Movie* *Comment* that he expressed before or delete it.
   4. A *User* can **delete** his account.

### Non-Functional

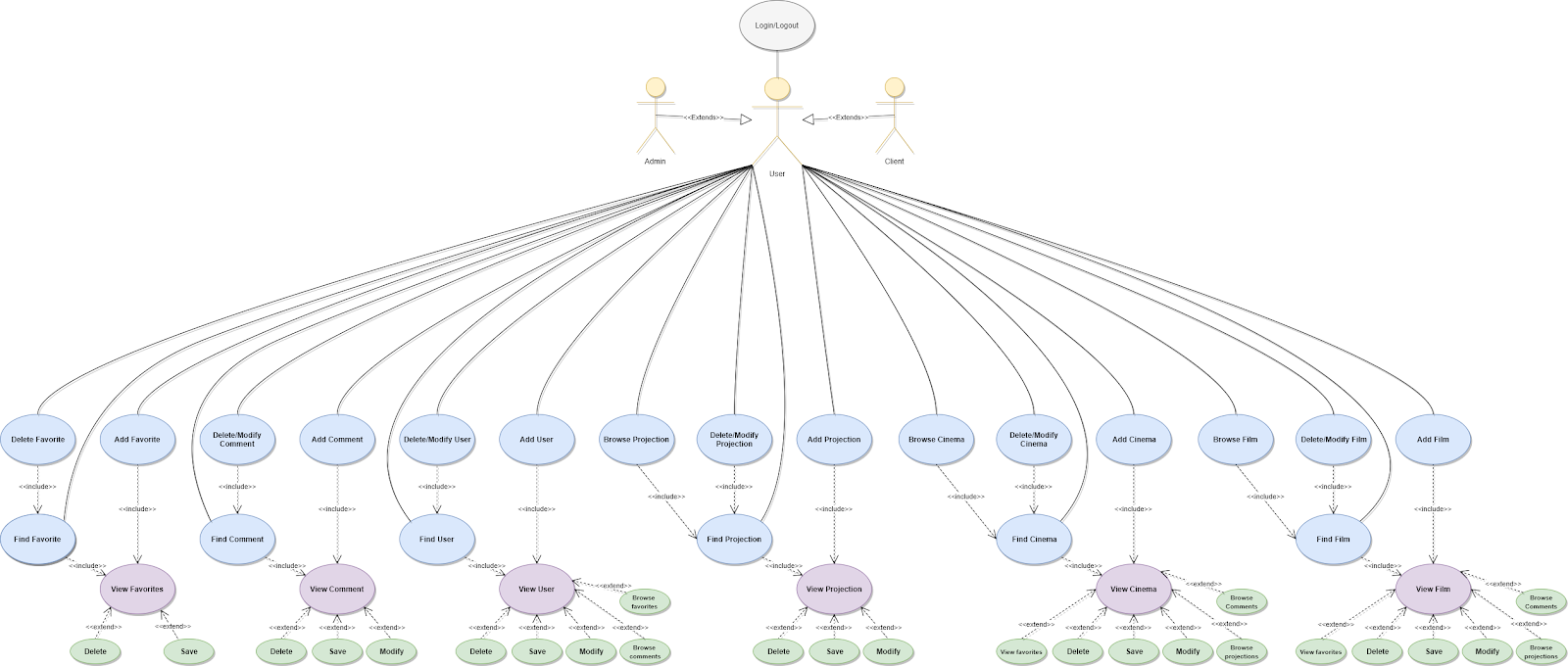
1. The systems must be on 24/24.
2. The system must support hundred of concurrent access.
3. The response time must be in the order of 1-10 ms.
4. The password must be protected and stored encrypted for privacy issues.

## Software Architecture

The aim of this project is to build up the platform PisaFlix, in order to do that, we need a relational Database where storing all the informations about movies, cinemas and users.

Users instead can use a java application with a basic GUI for using all functionalities of the platform (register, see movies list ecc…)

## Use Cases Diagram

****

## Class DIagram

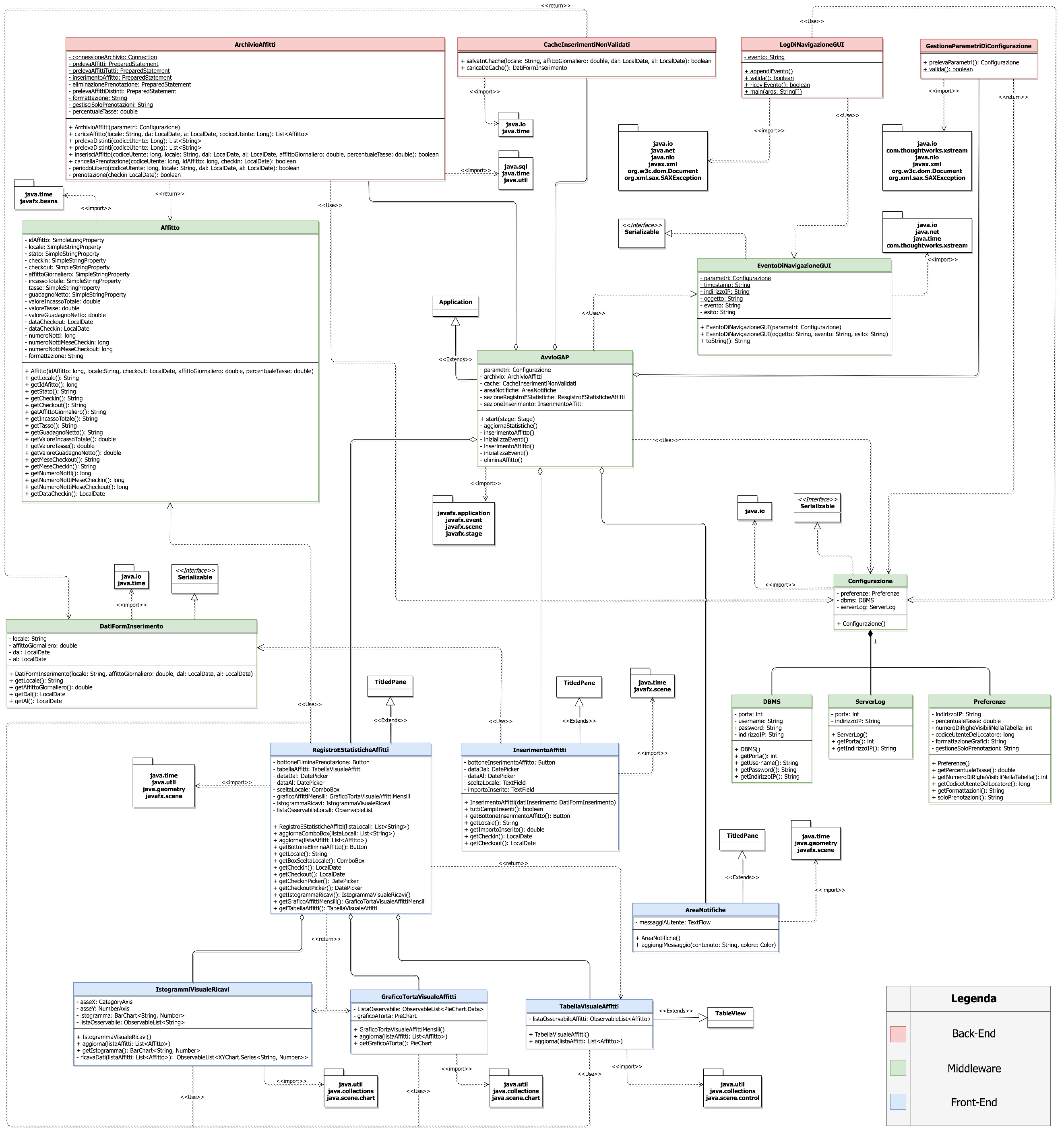


## Database Main Entities Diagram



# Documento di Progetto

## Architettura del Software



## Responsabilità Classi

### Back-end

* Classe **CacheInserimentiNonValidi**: contiene su file binario gli ultimi dati inseriti nel form di inserimento nuovi affitti; reinserisce i suddetti dati nei form attraverso la classe **DatiFormInserimento**.
* Classe **LogDiNavigazioneGUI**: (*Server*) riceve un evento di log XML; invoca la validazione; aggiunge la riga XML al file di log in modo incrementale.
* Classe **GestioneParametriDiConfigurazione**: legge il file di configurazione XML; invoca la validazione del file di configurazione XML e deserializza il contenuto XML come oggetto **Configurazione**.
* Classe **ArchivioAffitti**: preleva dati dall’Archivio e restituisce una lista di oggetti **Affitto** e una lista con i nomi dei locali; inserisce all’interno dell’archivio nuovi dati; cancella dall’archivio prenotazioni.

### Middleware

* Classe **AvvioGAP**: inizializza la GUI; è il controller applicativo che risponde agli eventi invocando le opportune classi.
* Classe **DatiFormInserimento**: implementa *Serializable*; contiene i dati relativi alla cache e li rende disponibili alla classe **InserimentoAffitti**.
* Classe **EventoDiNavigazioneGUI**: implementa *Serializable*; contiene le informazioni di un evento del log; si serializza in XML; spedisce l’evento al server **LogDiNavigazioneGUI**.
* Classe **Configurazione**: implementa *Serializable*; contiene i dati relativi al file di configurazione e li rende disponibili alle altre classi.
* Classe **Affitto**: classe bean; viene inizializzata mediante i dati prelevati dall’**ArchivioAffitti** e calcola i rimanenti necessari alle statistiche; utilizzata per inizzializzare la **TabellaVisualeAffitti,** il **GraficoTortaVisualeAffitti** e l’**IstogrammaVisualeRicavi**.

### Front-end

* Classe **InserimentoAffitti**: estensione di *TitledPane*; è il front-end della sezione di inserimento nuovi affitti;
* Classe **RegistroEStatisticheAffitti**: estensione di *TitledPane*; è il front-end della sezione che mostra il registro e le statistiche degli affitti;
* Classe **AreaNotifiche**: estensione di *TitledPane*; è il front-end della sezione che mostra tutte le notifiche a scorrimento per l’utente;
* Classe **TabellaVisualeAffitti**: estensione di *TableView*; è il front-end della tabella degli affitti; preleva i dati da una lista di elementi **Affitto**.
* Classe **GraficoTortaVisualeAffitti**: è il front-end del grafico a torta degli affitti; preleva i dati da una lista di elementi **Affitto** e calcola i rimanenti a partire dalla stessa.
* Classe **IstogrammaDeiRicavi**: è il front-end dell’istogramma dei ricavi; preleva i dati da una lista di elementi **Affitto** e calcola i rimanenti a partire dalla stessa.

# Documento di Collaudo

## Schermata dell’Applicazione

Figura 2: contenuto della cartella *src* del progetto

Figura 1: schermata dell’applicazione, formattazione *MMMM* dei grafici, aggiornamento delle statistiche

## Elementi di Back-End

### File di configurazione Locale in Xml

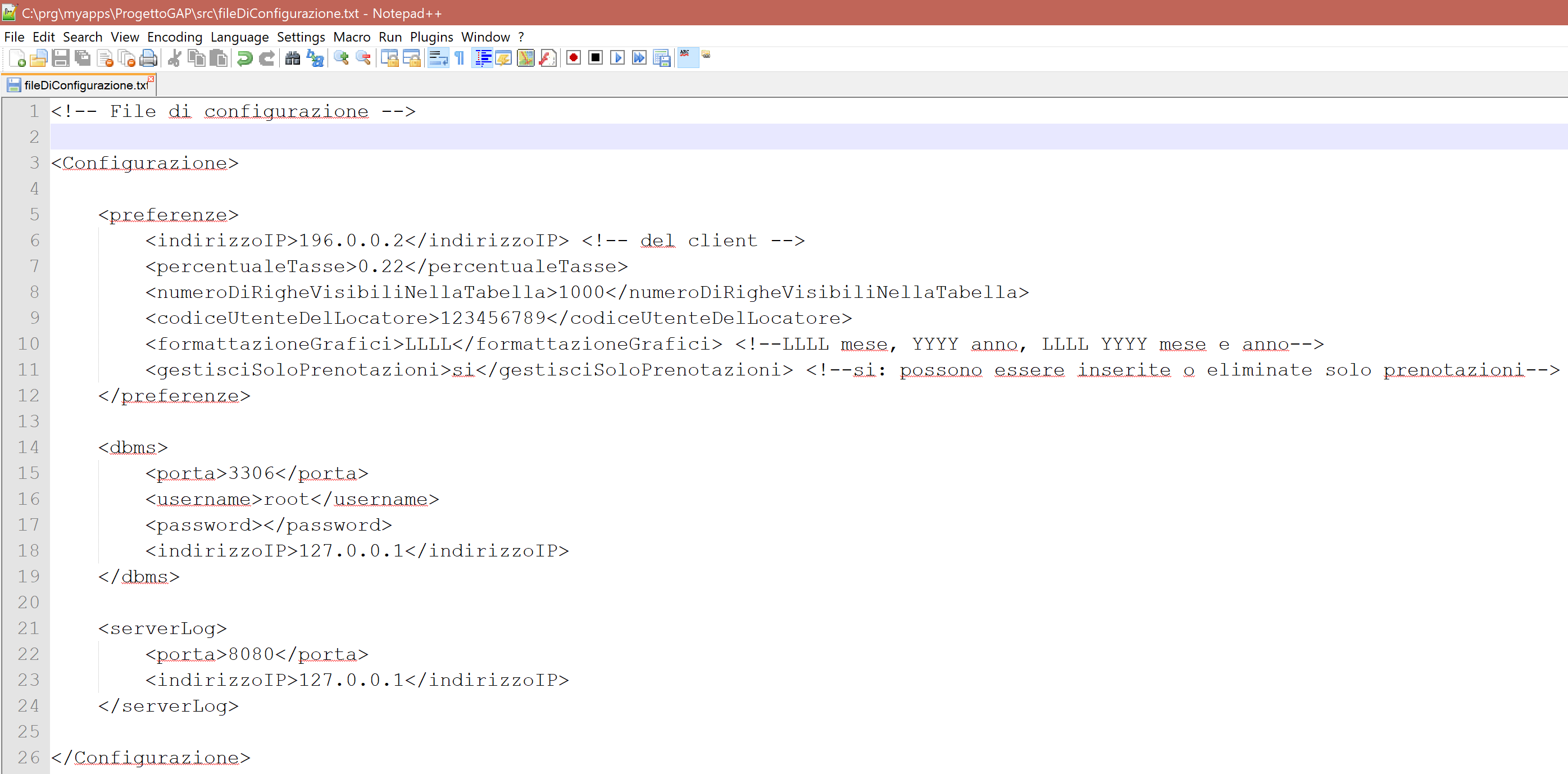
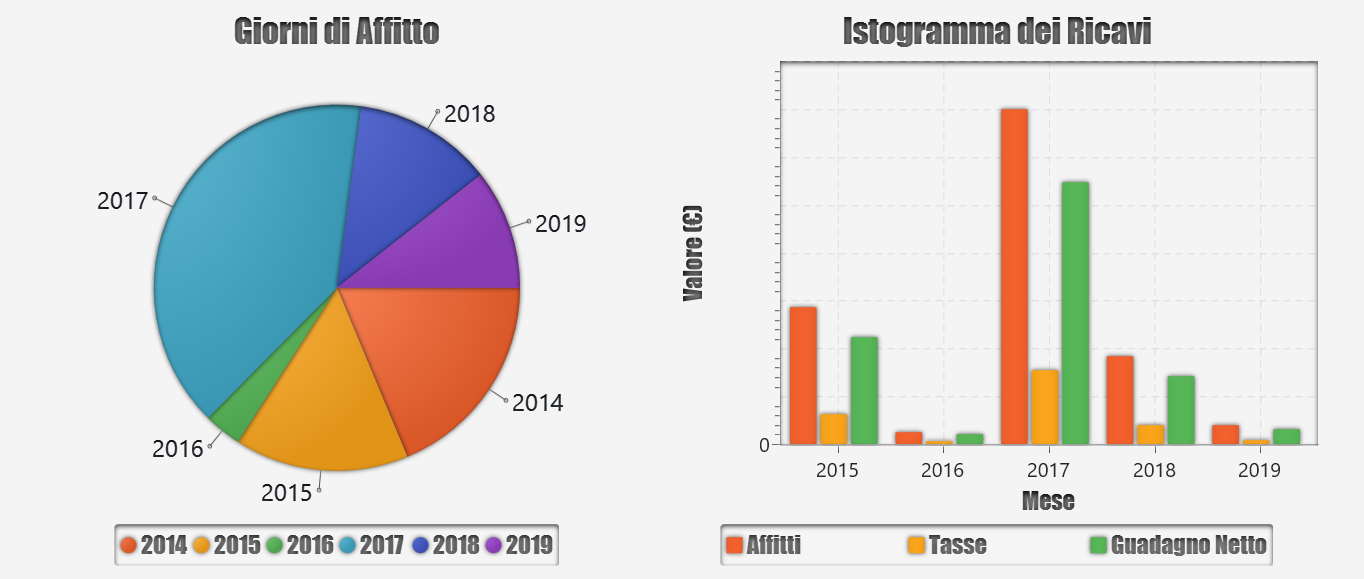


Figura 3: file di configurazione locale

Il file di configurazione è suddiviso in tre sezioni:

* **Preferenze**: riguarda le preferenze dell’utenente
  + L’indirizzo IP del client e il codice utente del locatore
  + La percentuale di “tasse” da apllicare all’affitto totale per ottenere il guadagno netto
  + Il tipo di formattazione dei grafici che può essere: *YYYY* per gli anni (figura 4), *MMMM* per mesi (figura 1) o *MMMM* *YYYY* per mesi e anni (figura 5)
  + Se è possibile gestire anche affitti *passati* o *in* *corso* oltre che *prenotati*
* **DBMS**: riguarda l’accesso all’archivio
* **ServerLog**: riguarda l’accesso al server di log remoto

****

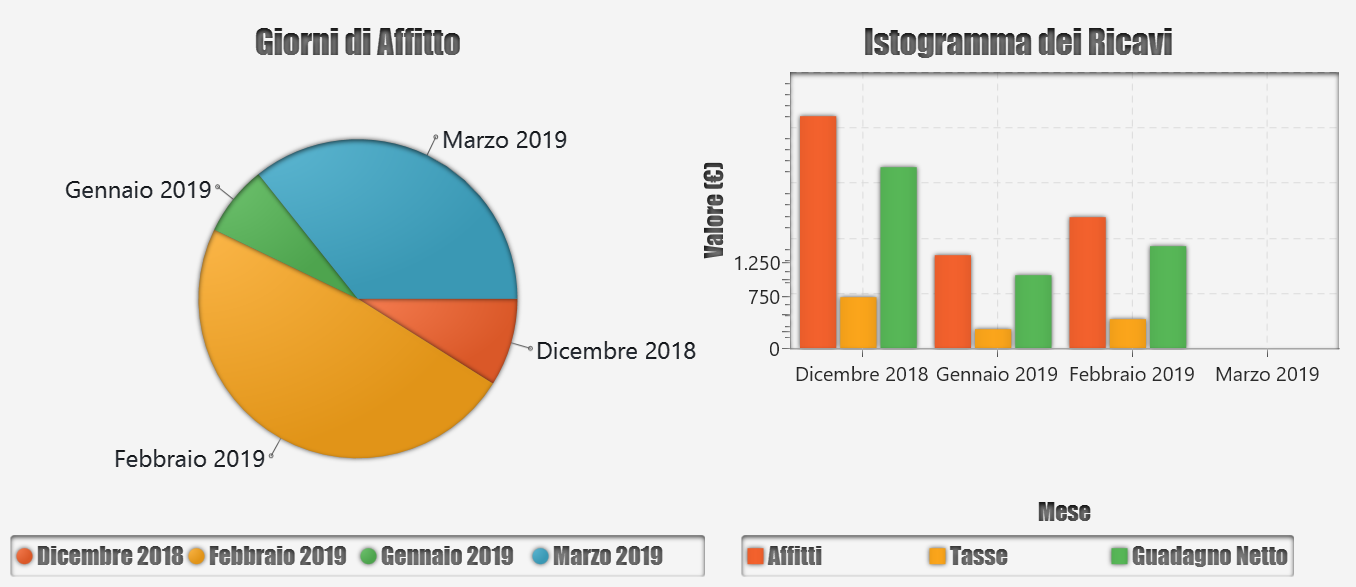
 Figura 4: formattazione *YYYY*

Figura 5: formattazione *MMMM YYYY*

### Archivio

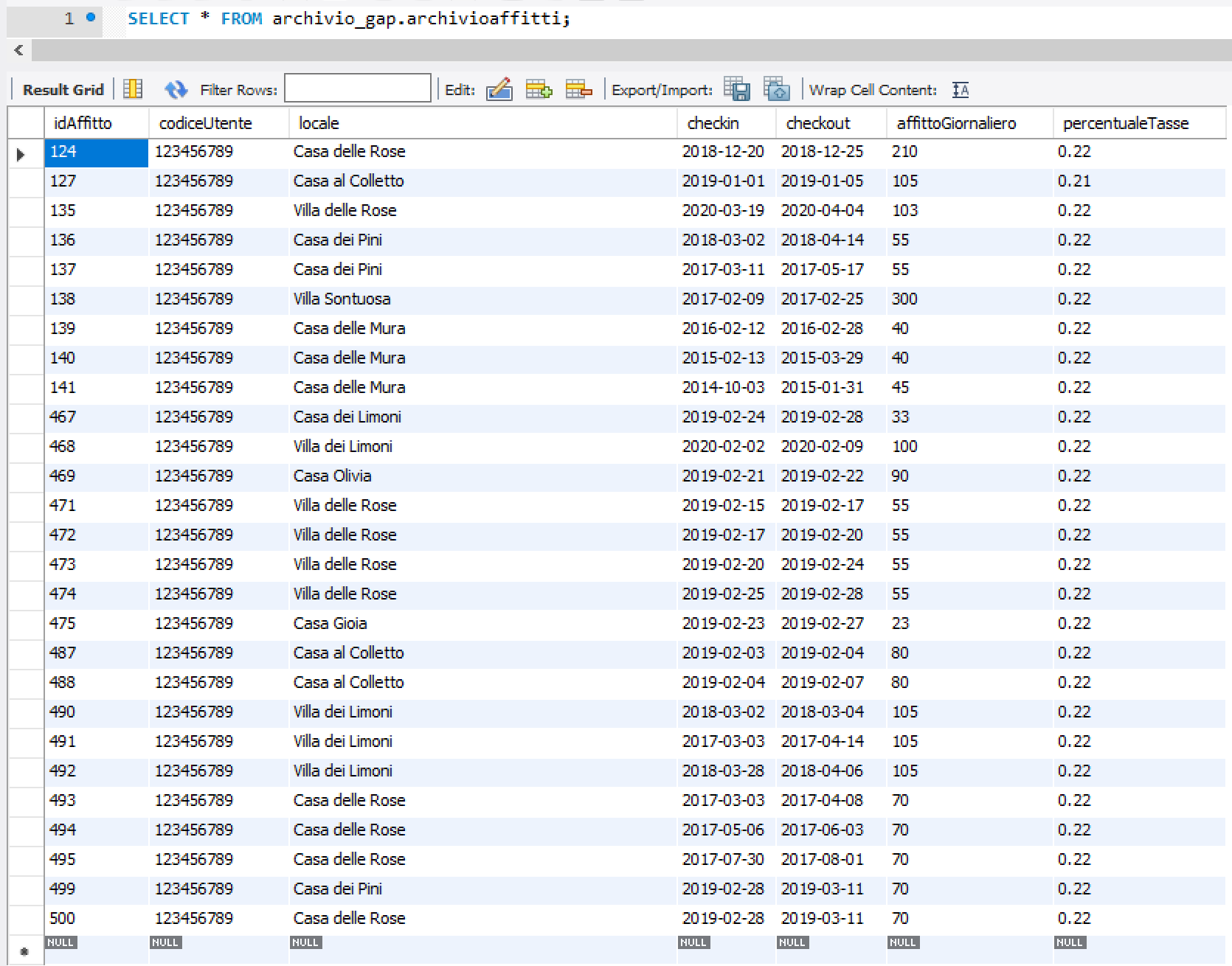
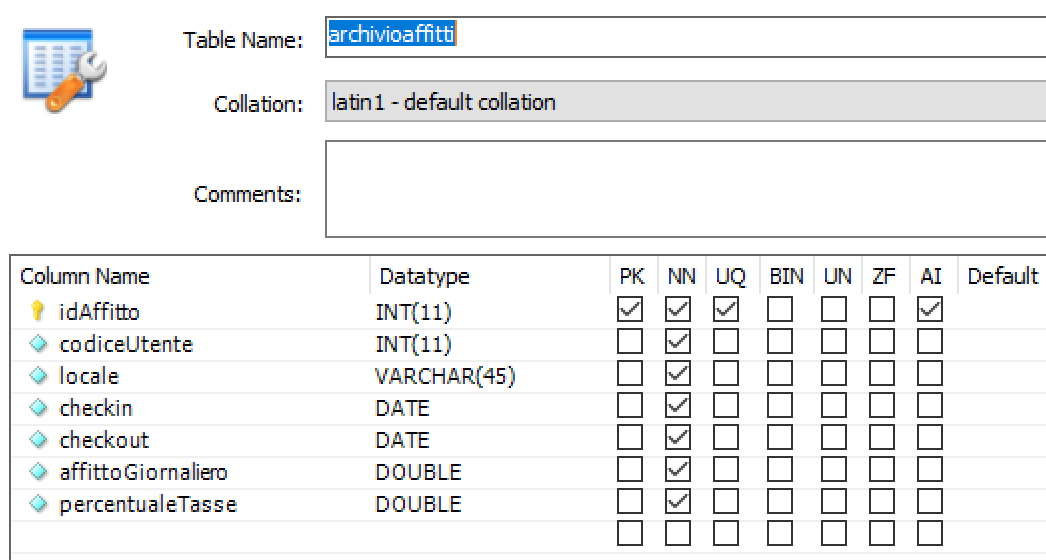


Figura 7: struttura della tabella *archivioaffitti*

Figura 6: elementi precaricati nel DB

Nell’archivio sono memorizzati i dati relativi agli affitti di vari utenti:

* L’ID univoco per ogni affitto
* Il codice dell’utente a cui appartengono gli affitti
* Il nome del locale la data di checkin, la data di checkout e l’affitto giornaliero
* La percentuale di tasse da applicare all’affitto

### File di Log Remoto in Xml

Nel file di log remoto sono contenute le informazioni relative ad ogni evento di navigazione eseguito sull’applicazione, per ogni evento sono inviati l’indirizzo IP del client, un timestamp che indica il momento in cui è avvenuta l’interazione, l’oggetto con cui è avvenuta l’interazione, la tipologia di evento e l’esito di tale evento, a fini statistici.

Figura 9: file di log remoto

Figura 8: schermata del server di log remoto

## Casi di Test

### Inserimento Nuovo Affitto

1. L’utente inserisce il nome di un locale preesistente o di un nuovo locale (vedere figura 10)
2. L’utente inserisce un valore numerico relativo all’affitto giornaliero
3. L’utente inserisce la data di checkin e la data di checkout
4. L’utente clicca sul pulsante “Inserisci Affitto”
   1. Il sistema controlla:
      1. Che l’utente abbia completato tutti i campi
      2. Che le date siano valide (checkin prima di checkout)
      3. Che le date non si sovrappongano ad una prenotazione precedentemente inserita
      4. Che il campo “Affitto Giornaliero” sia numerico
   2. Il sistema carica sull’in base all’esito dei controlli precedenti:
5. Segnala all’utente l’errore pertinente ad uno dei controlli sopra citati
6. Carica nell’archivio i dati inseriti e segnala all’utente l’avvenuto inserimento

Figura 0: sezione d’inserimento affitti e area notifiche

### Visualizzazzione Registro e Statistiche degli Affitti

1. Il sistema preleva dall’archivio i nome dei locali da inserire nella lista di selezione dei medesimi
2. L’utente seleziona (figura 1 sezione “Registro e Statistiche Affitti”):
3. Un locale presente nell’archivio dalla tendina a scorrimento
4. Una data da cui far partire le statistiche
5. na data a cui far arrivare le statistiche
6. Il sistema seguito ad una qualsiasi delle azioni sovracitate:
   1. Preleva dall’archivio i dati relativi al periodo e al locale richieso e li inserisce nella tabella
   2. Calcola le varie statistiche in base alla configurazione richiesta e le inserisce all’interno dei grafici

### Rimozione di una Prenotazione

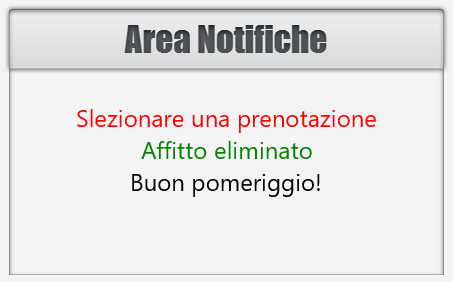
1. L’utente seleziona una riga della tabella (figura 12)
2. L’utente clicca sul bottone “Elimina Prenotazione”
3. Il sistema controlla:
   1. Che la riga appartenga ad una prenotazione (se così stabilito nelle preferenze)
   2. Che sia selezionata almeno una riga
4. Il sistema in base all’esito dei controlli precedenti (figura 11):
5. Elimina la prenotazione dall’archivio e segnala l’operazione all’utente

Figura 11: notifiche relative all’eliminazione di una prenotazione

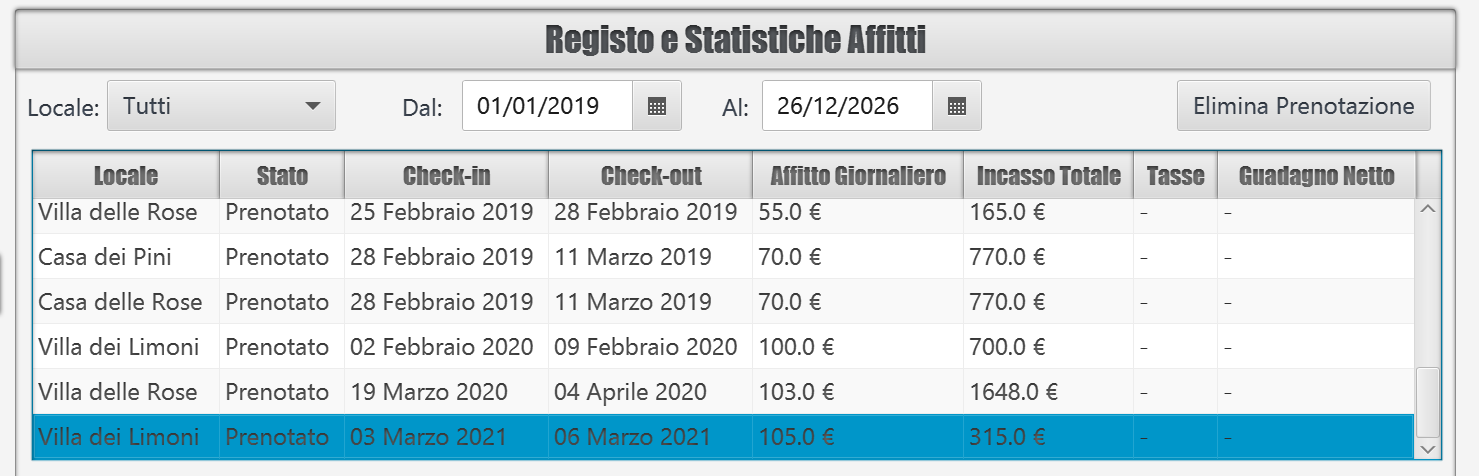
1. Segnala all’utente un opportuno messaggio di errore

Figura 12: eliminazione di una prenotazione

### Informazioni Aggiuntive Statistiche

1. L’utente preme in una qualsiasi area relativa ad uno dei due grafici (figura 1, 4 o 5)

Figura 13: notifiche relative alle informazione aggiuntive sui grafici

1. Il sistema mostra all’utente nell’area notifiche i valori numerici e l’identificatore dell’area che è stata cliccata (figura 13)