UNIVERSITÀ DI PISA

Corso di Laurea in Ingegneria Informatica

Documentazione del progetto per il corso di programmazione avanzata

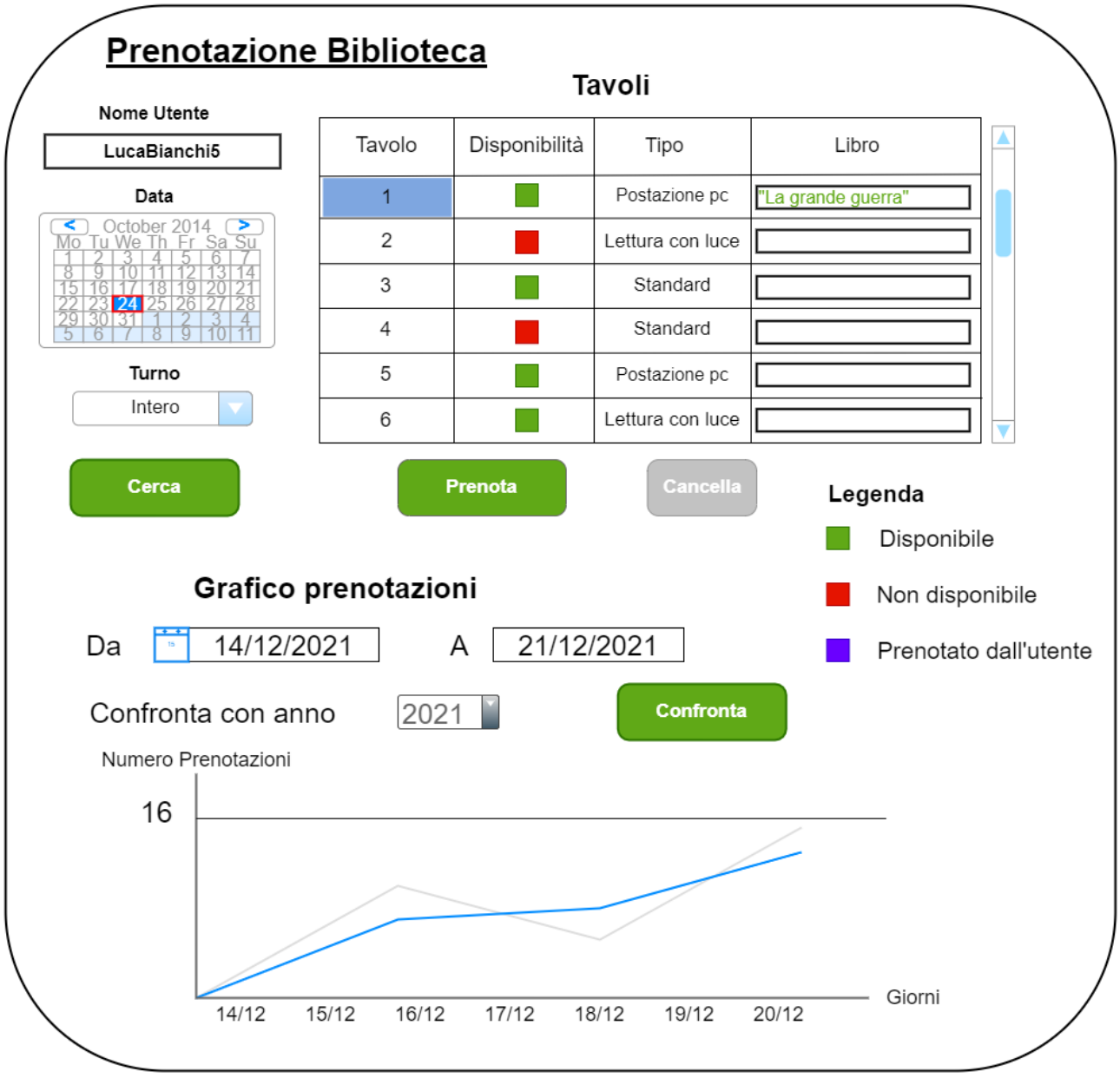
Anno accademico 2021/2022

Applicazione per prenotazioni in biblioteca

Luca Chiocchetti

Analisi

Mockup



Vista dinamica: Prenotazione

1) L'utente inserisce il Nome Utente
2) L'utente sceglie la Data
3) L'utente seleziona il Turno
4) L'utente preme su Cerca
5) Il Sistema visualizza la tabella dei Tavoli
6) L'utente clicca su una riga della tabella dei Tavoli
7) L'utente inserisce Libro
8) L'utente preme su Prenota
9) Il Sistema verifica la disponibilità e aggiorna il colore di Libro
10) IF Libro è disponibile
    10.1) Il Sistema aggiorna il colore di Disponibilità della riga con Prenotato dall'utente
11) IF l'utente preme su dataDa e/o l'utente preme su annoConfronto e l'utente Preme su Confronta
    11.1) il Sistema aggiorna il Grafico Prenotazioni

Vista dinamica: Cancellazione

1) L'utente inserisce il Nome Utente usato nella prenotazione
2) L'utente inserisce la Data della prenotazione
3) L'utente seleziona il Turno della prenotazione
4) L'utente preme su Cerca
5) Il Sistema visualizza la tabella dei Tavoli
6) L'utente clicca su una riga della tabella dei Tavoli con Disponibilità di colore Prenotato dall'utente
7) L'utente preme su Cancella
8) Il Sistema aggiorna il colore di Disponibilità della riga con Disponibile.
9) IF l'utente preme su dataDa e/o l'utente preme su annoConfronto e l'utente Preme su Confronta
    9.1) il Sistema aggiorna il Grafico Prenotazioni

Specifica di altre parti Java I/O

File di configurazione locale in XML

All'avvio il sistema legge dal file di configurazione i seguenti dati:

- L'indirizzo IP del client
- L'indirizzo IP e porta del server di log
- Indirizzo IP e porta, username e password del DBMS.
- Font e colore del background
- Altezza e larghezza del Grafico Prenotazioni
- I valori che può assumere Turno: Intero, Mattina, Pomeriggio

- Il numero di giorni di default dell'intervallo [da,a] con cui viene Inizializzato il campo dataDa
- L'anno di default per il confronto delle prenotazioni
- I valori esadecimali dei colori della legenda in formato stringa

Cache locale degli input

Alla chiusura, se l'utente stava tentando di effettuare una prenotazione, il sistema salva su file binario tutte le informazioni per riprendere la prenotazione da dove è stata interrotta: Nome Utente, Data, Turno e Tavolo selezionato (se presente), il nome del Libro (se inserito). Se modificate, salva anche le informazioni relative al Grafico prenotazioni: data del campo dataDa e il valore di annoConfronto

All'avvio il sistema carica dal file di binario i suddetti dati.

Archivio

Il Sistema Archivia i seguenti Dati:

- Nome utente
- Data
- Turno
- Numero del tavolo
- Tipo del tavolo
- Titolo del libro

File di LOG remoto in XML

Il sistema invia una riga di log ad ogni evento di seguito:

- Avvio dell'applicazione. ("AVVIO")
- Pressione dei pulsanti "Cerca", "Prenota" o "Cancella".
- Pressione del tasto "Confronta"
- Termine dell'applicazione. ("TERMINE")

La riga di log contiene: nome dell'applicazione, indirizzo IP del client, Data e ora corrente e l'etichetta associata all'evento.

Progetto

Descrizione delle classi

Classe **PrenotazioneBibliotecaGui**: è la classe che si occupa della gestione dell'interfaccia grafica dell'applicazione, la sua inizializzazione avviene disponendo gli elementi nell'ordine corretto tramite i suoi metodi.

Classe **GestoreBiblioteca:** è la classe che si occupa di aggiornare l'interfaccia grafica in base all'interazione dell'utente. Si occupa di rispondere agli eventi, come la pressione dei bottoni, chiamando le opportune funzioni delle altre classi come *EseguiRichiestaUtente*.

Classe **StatoTavoloGiorno**: è la classe che rappresenta il concetto di stato di un tavolo in un dato giorno, contiene le seguenti informazioni: Numero tavolo, disponibilità, tipo del tavolo ed il libro prenotato.

Classe **DialogaConDB**: è la classe che si occupa di creare le connessioni con il database Mysql, gestire gli eventuali errori legati ad essa e di fornire i metodi per recuperare i dati dal database alle altre classi.

Classe **EseguiRichiestaUtente**: è la classe che si occupa di fornire i metodi per rispondere alle effettive richieste dell'utente utilizzando i metodi di *DialogaConDB* per ottenere le informazioni necessarie dal database. Offre i suoi metodi a *GestoreBiblioteca* per rispondere agli eventi generati dall'utente.

Classe **CachePrenotazioneBiblioteca**: è la classe che si occupa di salvare i dati in fase di chiusura dell'applicazione e quando viene riaperta, li recupera riportando l'applicazione nello stato in cui era quando è stata chiusa.

Classe **ParametriDiConfigurazione**: è la classe che contiene i parametri di configurazione dell'applicazione che verranno utilizzati dalle altre classi. Utilizza la classe *GestoreXML* per ottenere i parametri XML.

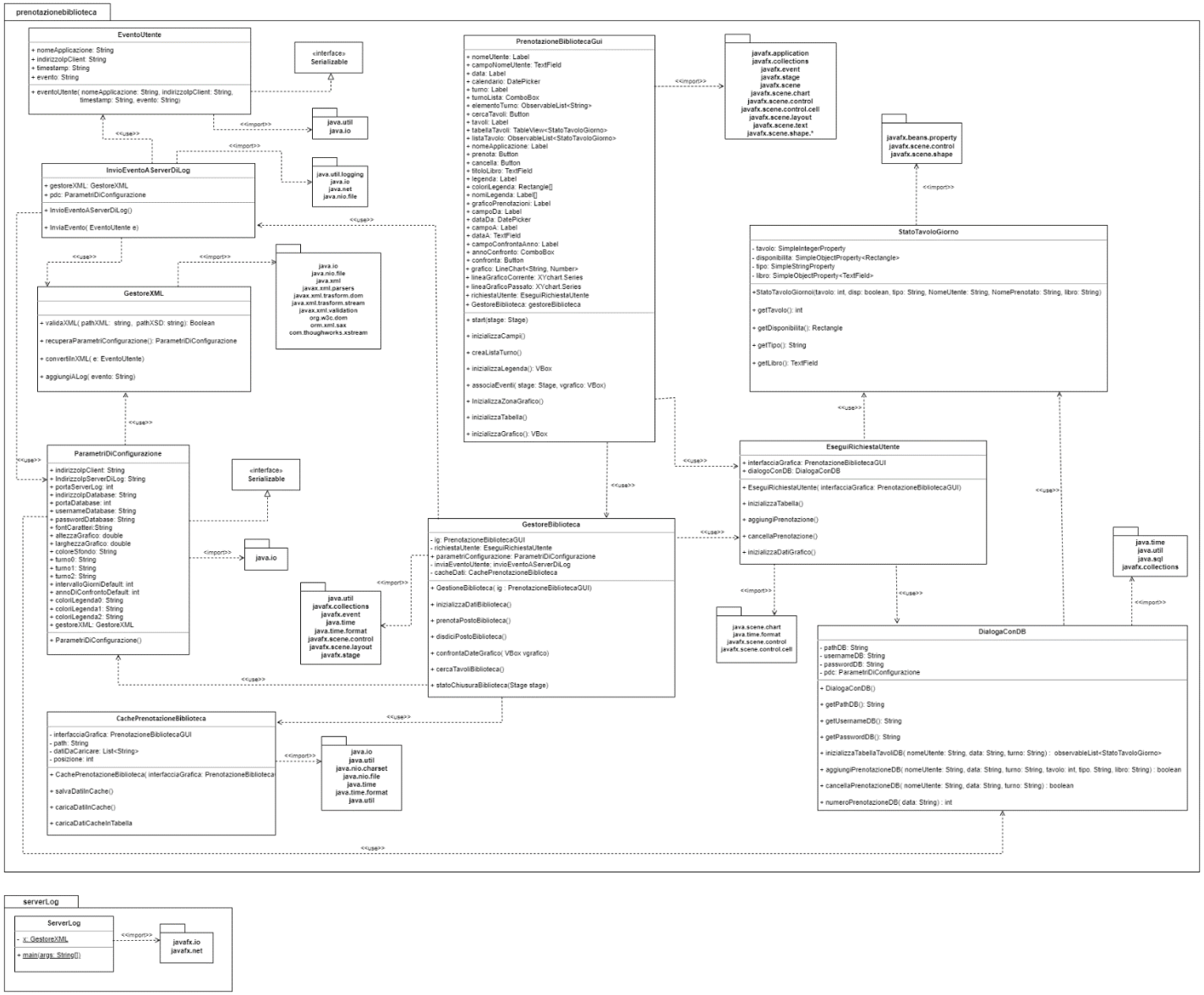
Classe **EventoUtente**: è la classe che rappresenta l'evento generato dall'utente all'interno dell'applicazione che verrà poi inviata al Server di Log. Ha come attributi il nome dell'applicazione, l'indirizzo IP del Client, data e ora al momento dell'invio dell'evento e l'evento svolto dall'utente. È una classe Serializable e viene utilizzata dalla classe *InvioEventoAServerLog*.

Classe **InviaEventoAServerDiLog**: è la classe che si occupa di notificare un nuovo evento utente creando un oggetto di tipo *EventoUtente* e di inviarlo al *Server di Log.*

Classe **GestoreXML:** è la classe che serve per realizzare tutte le operazioni che riguardano l'XML come la validazione, deserializzazione e conversione in stringa.

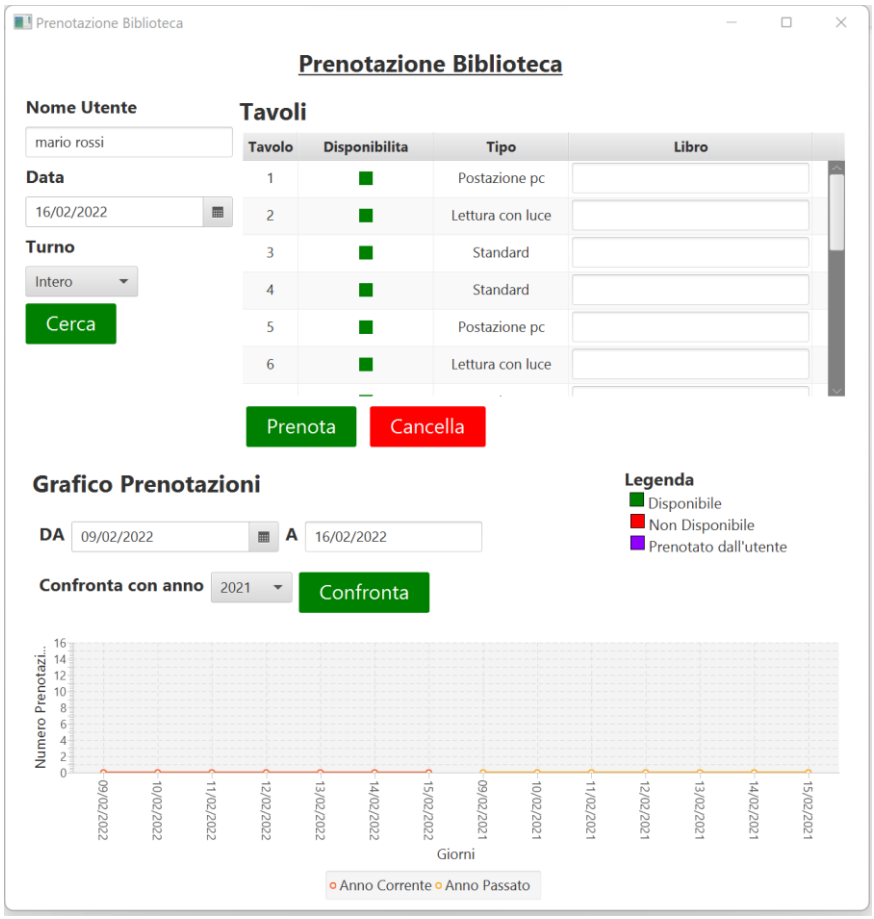
Classe **ServerLog:** è la classe che riceve il messaggio di log in formato XML e lo salva su file di testo locale usando la classe *GestoreXML.*

Diagramma di Classe



Collaudo

Una volta avviata l'applicazione, viene mostrata la seguente schermata:

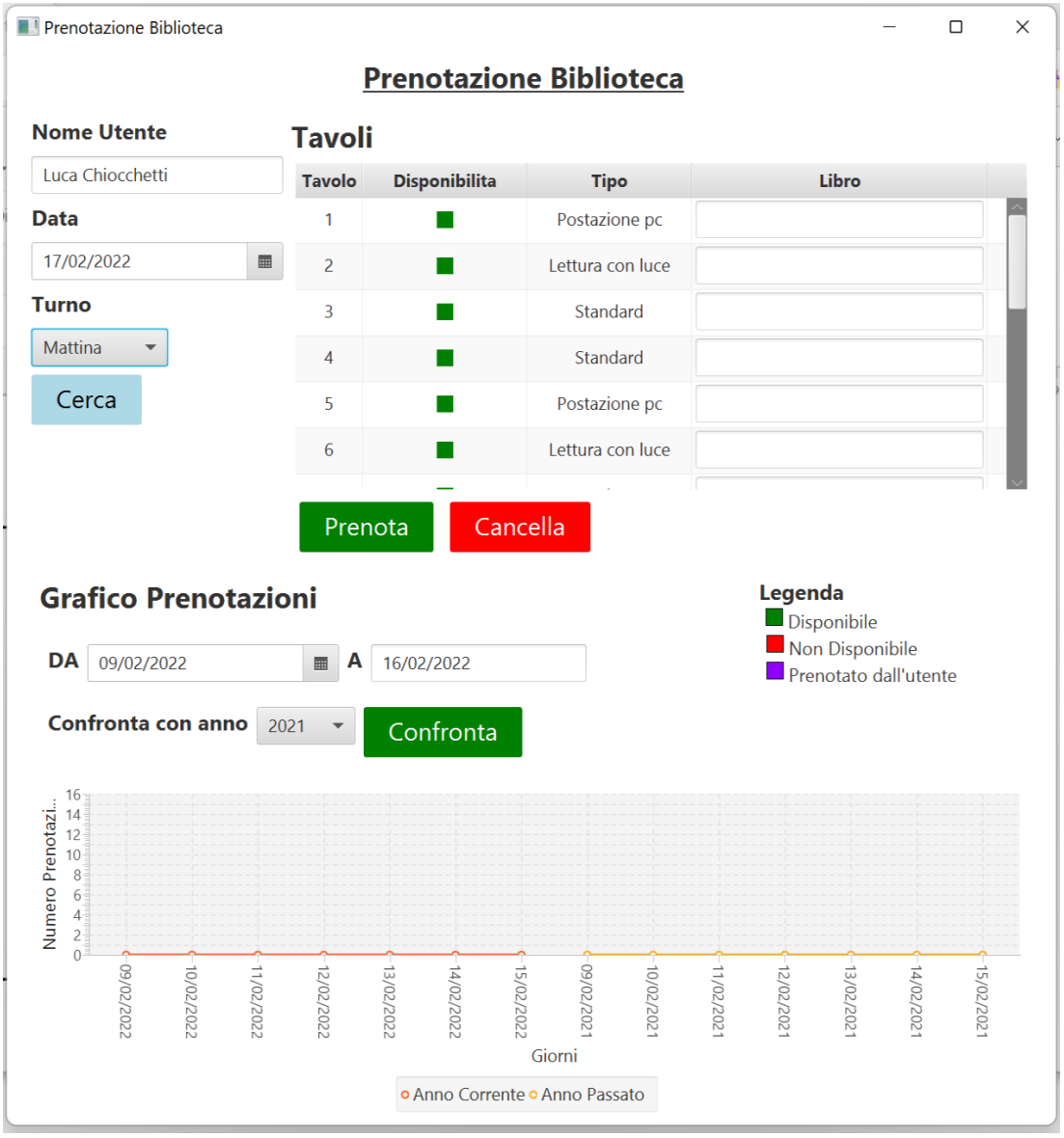


L'interfaccia si presenta divisa in 3 sezioni:

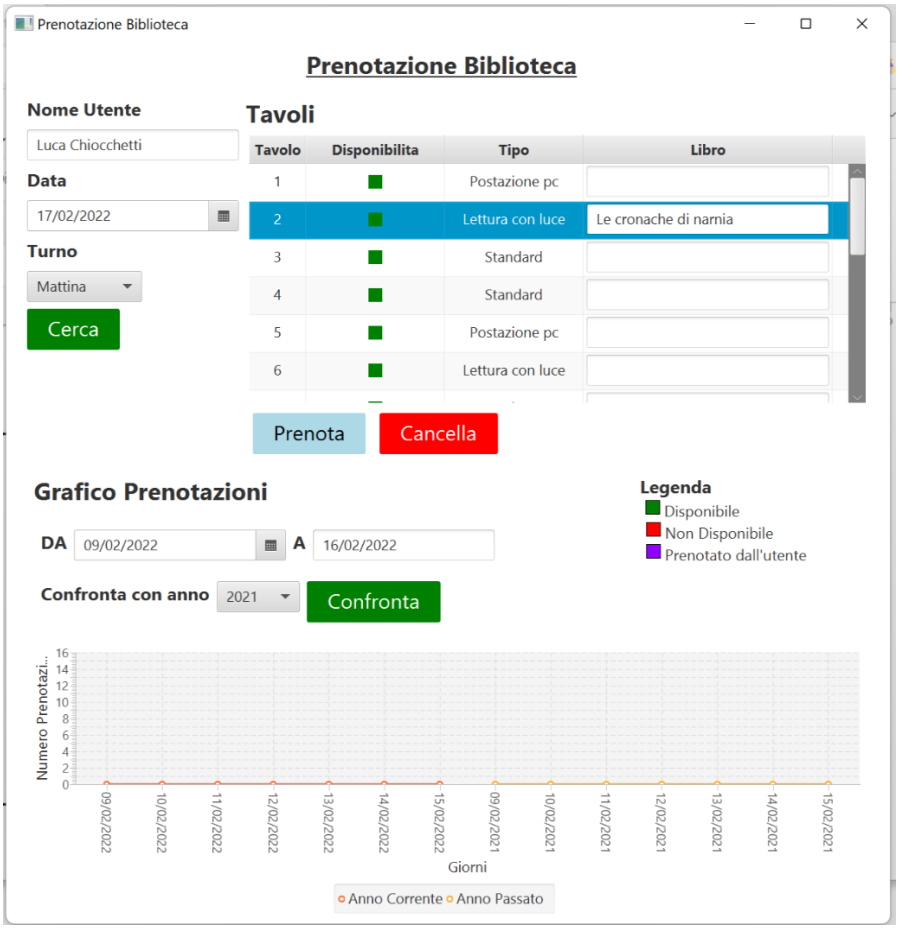
1) In alto a sinistra è presente la sezione che contiene gl'informazioni che deve inserire l'utente al fine di poter prenotare.
2) In alto a destra è presente la sezione che contiene la tabella dello stato dei tavoli per il giorno scelto dall'utente.
3) L'ultima sezione, quella in basso, contiene il grafico delle prenotazioni e permette un confronto delle prenotazioni di un intervallo di tempo fra l'anno corrente e un anno a scelta dell'utente.

   All'avvio i campi della sezione 1) vengono inizializzati con dei valori di default che hanno lo scopo di esempio per l'utente.

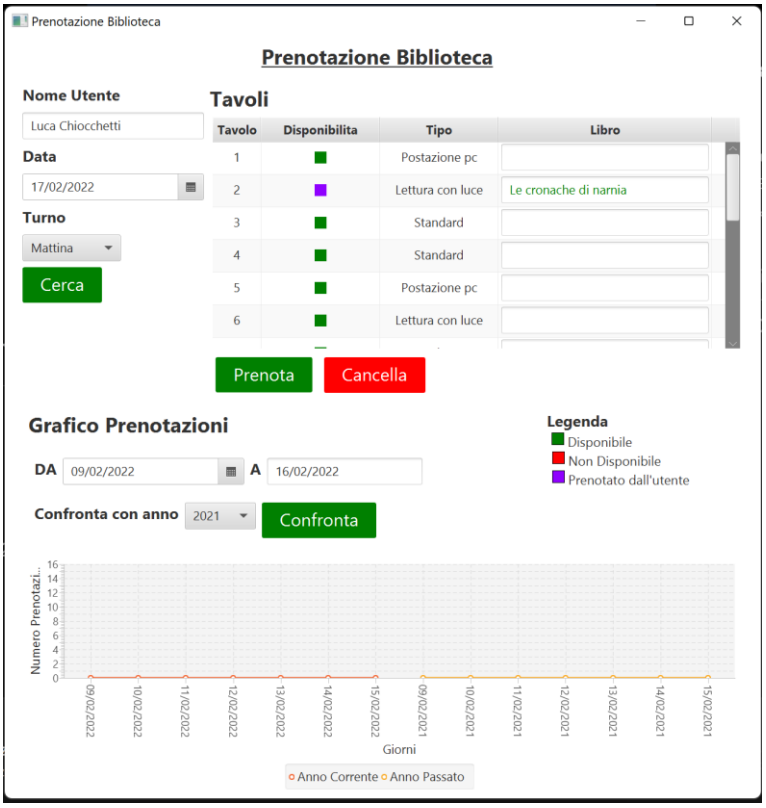
Dopo l'avvio, l'utente inserisci il nome Utente con cui decide di prenotare il posto, la data di prenotazione, il turno della giornata per cui vuole occupare il tavolo e preme sul bottone "Cerca", l'applicazione visualizzerà lo stato dei tavoli per quel giorno.



Una volta visualizzato lo stato dei tavoli, l'utente seleziona il tavolo da prenotare, inserisce il titolo del libro che è interessato a noleggiare e preme su prenota.



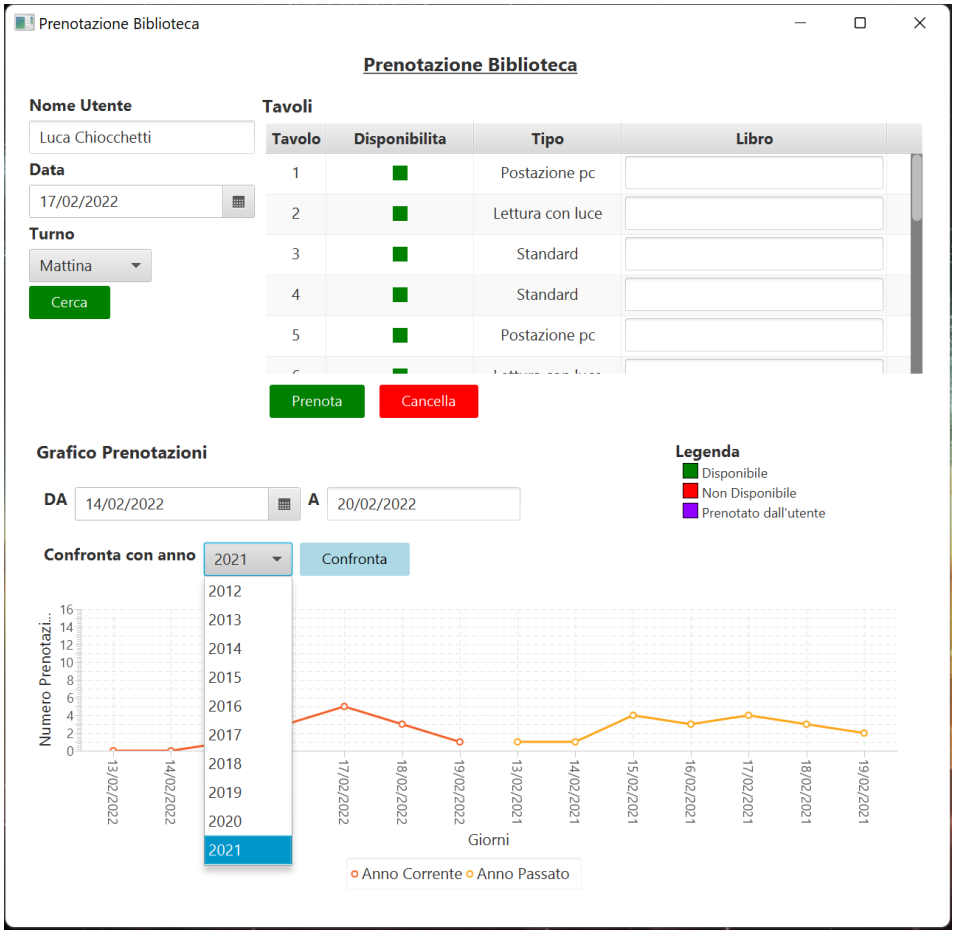
Se il libro è disponibile per quel giorno, la prenotazione andrà a buon fine e il colore Di “Disponibilità” cambierà con “Prenotato dall’utente”.



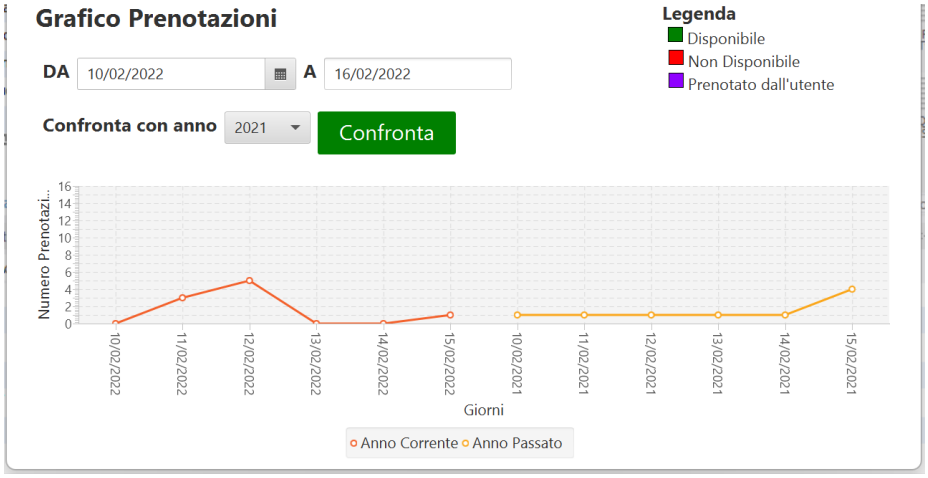
Per cancellare basta inserire tutti i dati messi al momento della prenotazione selezionare il tavolo prenotato che ha “Disponibilità” con colore “Prenotato dall’utente” e premere su Cancella.

Dopo di che, il colore di “Disponibilità” del tavolo selezionato tornerà ad essere “Disponibile”.

Per quanto riguarda l'utilizzo del grafico per confrontare lo stesso periodo in anni diversi: tramite il calendario fornito dopo l'etichetta Da, l'utente seleziona la data di inizio del periodo e sceglie l'anno con cui confrontarlo, dopo di che preme sul bottone Confronta e il grafico verrà aggiornato.



Una volta premuto sul pulsante Confronta, il grafico assumerà questo tipo di aspetto



Altre informazioni

Parametri Di Configurazione: Le funzionalità e l'aspetto dell'applicazione sono influenzati dai Parametri di configurazione presenti nel file parametriDiConfigurazione.xml, il contenuto è di seguito riportato:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<ParametriDiConfigurazione xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <indirizzoIpClient>localhost</indirizzoIpClient>
 <indirizzoIpServerDiLog>localhost</indirizzoIpServerDiLog>
 <portaServerLog>4243</portaServerLog>
 <indirizzoIpDatabase>localhost</indirizzoIpDatabase>
 <portaDatabase>3306</portaDatabase>
 <usernameDatabase>root</usernameDatabase>
 <passwordDatabase></passwordDatabase>
 <fontCaratteri>Helvetica</fontCaratteri>
 <coloreSfondo>white</coloreSfondo>
 <altezzaGrafico>400</altezzaGrafico>
 <larghezzaGrafico>1200</larghezzaGrafico>
 <turni0>Mattina</turni0>
 <turni1>Pomeriggio</turni1>
 <turni2>Intero</turni2>
 <intervalloGiorniDefault>7</intervalloGiorniDefault>
 <annoDiConfrontoDefault>1</annoDiConfrontoDefault>
 <coloriLegenda0>green</coloriLegenda0>
 <coloriLegenda1>red</coloriLegenda1>
 <coloriLegenda2>'0x8F00FF'</coloriLegenda2>
</ParametriDiConfigurazione>
```

Per validarne dinamicamente il contenuto è stato realizzato il file parametriDiConfigurazione.xsd che ne contiene la grammatica, anche esso di seguito riportato:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="ParametriDiConfigurazione">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="indirizzoIpClient" type="xs:string"/>
                <xs:element name="indirizzoIpServerDiLog" type="xs:string"/>
                <xs:element name="portaServerLog" type="xs:int"/>
                <xs:element name="indirizzoIpDatabase" type="xs:string"/>
                <xs:element name="portaDatabase" type="xs:int"/>
                <xs:element name="usernameDatabase" type="xs:string"/>
                <xs:element name="passwordDatabase" type="xs:string"/>
                <xs:element name="fontCaratteri" type="xs:string"/>
                <xs:element name="coloreSfondo" type="xs:string"/>
                <xs:element name="altezzaGrafico" type="xs:double"/>
                <xs:element name="larghezzaGrafico" type="xs:double"/>
                <xs:element name="turni0"  type="xs:string"/>
                <xs:element name="turni1"  type="xs:string"/>
                <xs:element name="turni2"  type="xs:string"/>
                <xs:element name="intervalloGiorniDefault" type="xs:int"/>
                <xs:element name="annoDiConfrontoDefault" type="xs:int"/>
                <xs:element name="coloriLegenda0" type="xs:string"/>
                <xs:element name="coloriLegenda1" type="xs:string"/>
                <xs:element name="coloriLegenda2" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

File di Log: Ogni azione compiuta dall'utente sull'applicazione si tramuta in un evento che viene inviato al server di log e salvata su file di testo.

L'evento viene convertito in una stringa xml che ha il seguente tipo di formato:

```xml
<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>07:17:46 13/02/2022</timeStamp>
  <evento>cancella</evento>
</EventoUtente>
```

Per validare dinamicamente il file xml prima dell'invio è stato realizzato il file eventoUtente.xsd che contiene la sua grammatica.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="EventoUtente">
      <xs:complexType>
          <xs:sequence>
            <xs:element name="nomeApplicazione" type="xs:string"/>
            <xs:element name="indirizzoIpClient" type="xs:string"/>
            <xs:element name="timeStamp" type="xs:string"/>
            <xs:element name="evento" type="xs:string"/>
          </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Il file di log presente sul server è un file di testo che non presenta un tag xml di chiusura o di apertura e per tanto non potrà essere validato.

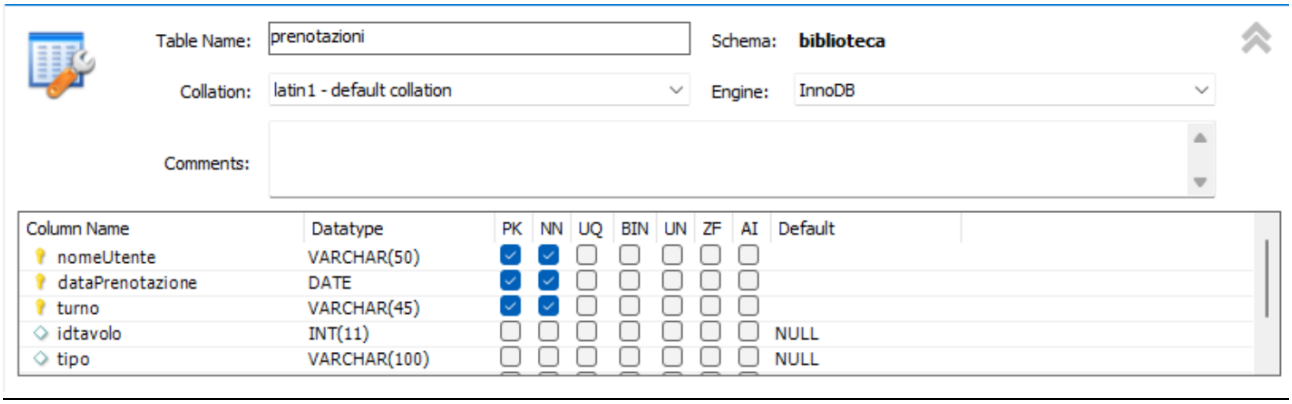
Un esempio del suo contenuto è il seguente:

```xml
<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 12:09:27 CET</timeStamp>
  <evento>AVVIO</evento>
</EventoUtente>

<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 12:35:49 CET</timeStamp>
  <evento>Confronta</evento>
</EventoUtente>

<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 12:35:50 CET</timeStamp>
  <evento>Confronta</evento>
</EventoUtente>

<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 12:35:51 CET</timeStamp>
  <evento>Confronta</evento>
</EventoUtente>

<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 12:47:34 CET</timeStamp>
  <evento>Prenota</evento>
</EventoUtente>

<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 14:49:30 CET</timeStamp>
  <evento>Cerca</evento>
</EventoUtente>

<EventoUtente>
  <nomeApplicazione>PrenotazioneBiblioteca</nomeApplicazione>
  <indirizzoIpClient>localhost</indirizzoIpClient>
  <timeStamp>2022-02-16 14:49:39 CET</timeStamp>
  <evento>Prenota</evento>
</EventoUtente>
```

Database: Per la gestione e la memorizzazione dell'informazioni utili all'applicazione come le prenotazioni e i tipi di tavoli, abbiamo fatto ricorso a mysql workbench.

Le tabelle realizzate che fanno parte del database "biblioteca" sono le seguenti:

- prenotazioni



- tavoli