RELAZIONE PROGETTO BIBLIOTECA COMUNALE

BocchiStore

[Schema Er/Logico 1](#_Toc1705126162)

[Index (Pagina principale) 1](#_Toc1564745827)

[Tutti i libri disponibili 2](#_Toc1051940676)

[Tutti gli iscritti in biblioteca 4](#_Toc604884554)

[Tutti i libri prestati ad un utente specifico in ordine cronologico 5](#_Toc1687406079)

[Individua i primi tre lettori che hanno letto più libri 7](#_Toc758024994)

[Individua tutti i possessori dei libri non ancora rientrati e il titolo degli stessi 9](#_Toc988282327)

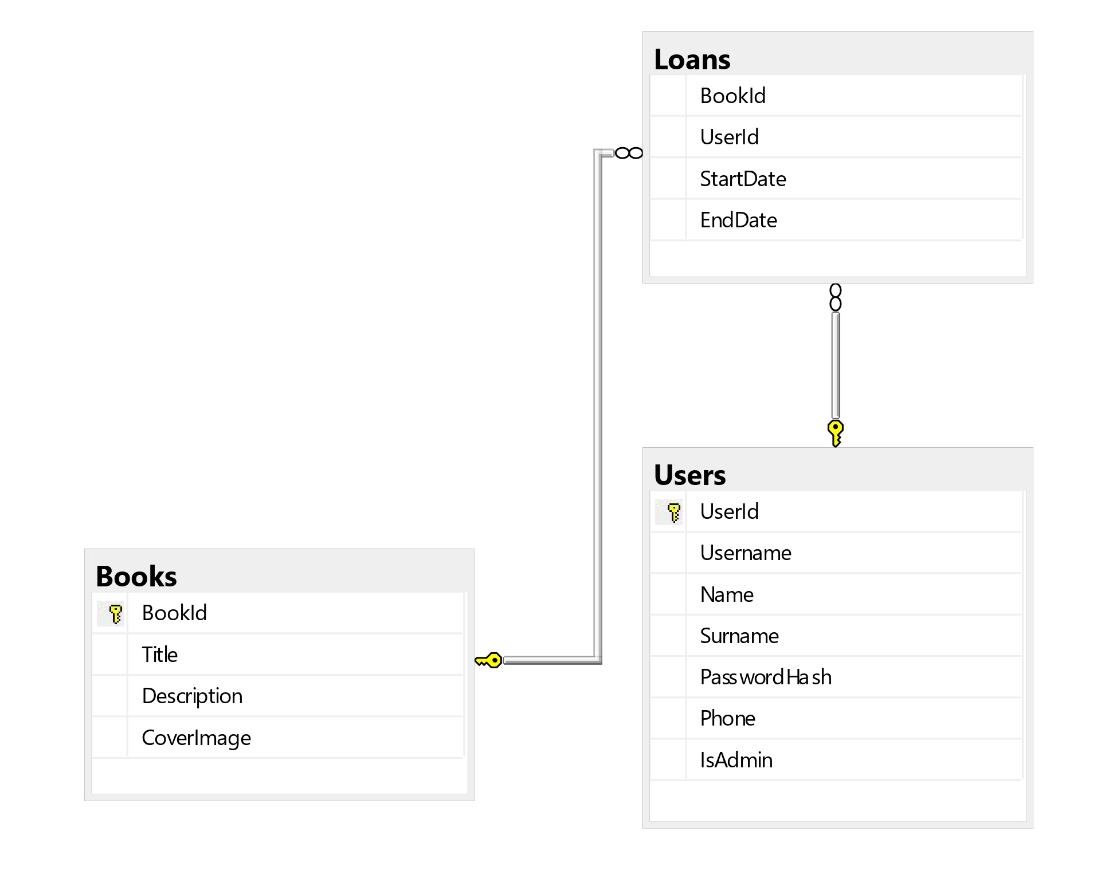
[Dare lo storico dei libri chiesti in prestito da un utente indicando il periodo 10](#_Toc409484722)

[Fai la classifica dei libri maggiormente prestati 10](#_Toc1360935360)

[Individua prestiti la cui data supera i 15gg 12](#_Toc2004763827)

# Schema Er/Logico

Questo è lo schema del database utilizzato per il progetto, l’abbiamo ideato in questa maniera



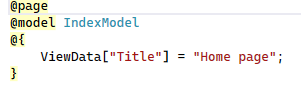
Schema logico

Books  (BookId, Title, Description, CoverImage)  
Users  (UserId, Username, Name, Surname, Phone)  
Loans  (BookId, UserId, StartDate, EndDate)

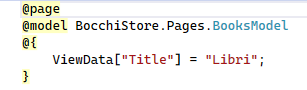
# Index (Pagina principale)

La pagina index contiene la top 3 utenti che hanno letto più libri e la classifica di libri più letti

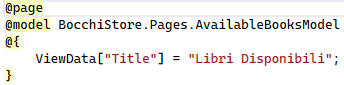
* Home

Home page dell’applicazione 

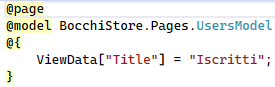
* Libri

Elenco libri complessivi

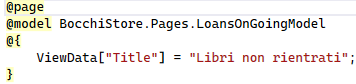
* Libri disponibili

Libri disponibili

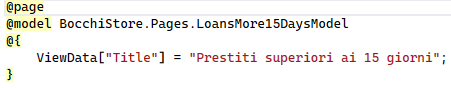
* Iscritti

Iscritti alla biblioteca

* Libri non rientrati

Elenco libri non tornati dal prestito

* Prestiti maggiore di 15 giorni

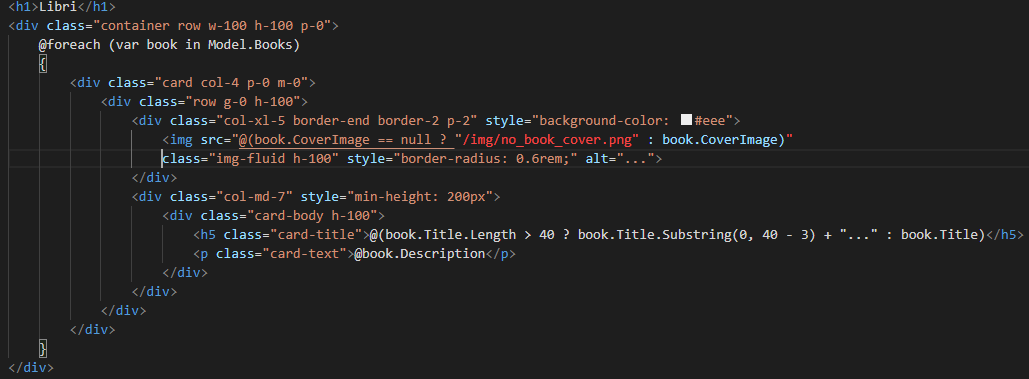
Elenco prestiti maggiore di 15 giorni

# Tutti i libri disponibili

Questa funzione serve per mostrare solo i libri disponibili questa è la schermata quando viene avviato



Nel Front end viene fatta una iterazione sulla lista restituita da model.books la lista viene generata dalla query sottostante al metodo storage. GetAvailableBooks che la esegue



public List<BookModel> Books => \_storage.GetAvailableBooks ().ToList();

La query viene eseguita dalla classe DapperStorage nel metodo GetAvailableBooks

public IEnumerable<BookModel> GetAvailableBooks()

{

    return \_connection.Query<BookModel>("SELECT \* FROM Books WHERE Books.BookId NOT IN (SELECT b.BookId FROM Books b INNER JOIN Loans ON Books.BookId = Loans.BookId WHERE Loans.EndDate IS NULL)");

}

La DapperStorage implementa la connessione dati e implementa la query e le trasforma in liste di dati.

Il costruttore della classe provvede a stabilire la connessione al db

public DapperStorage()

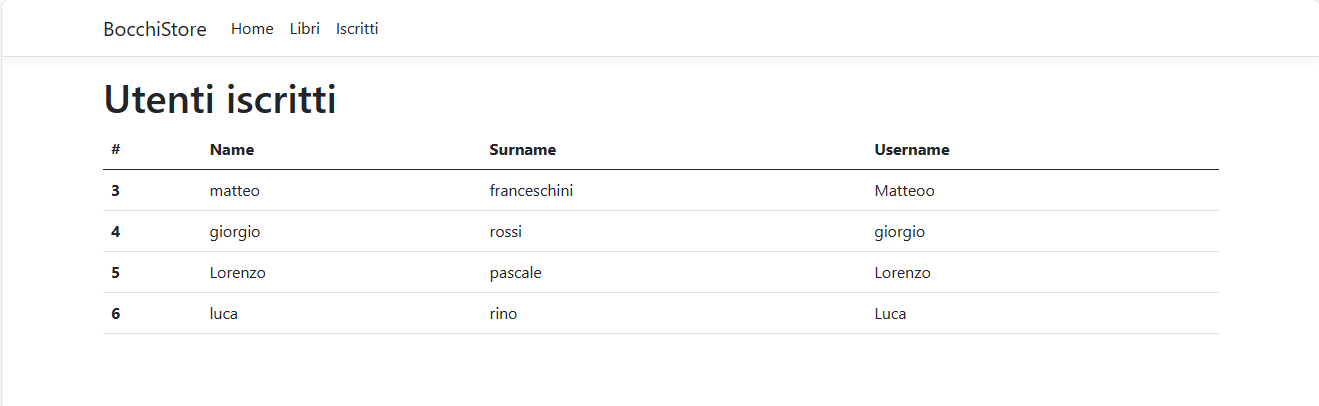
{

\_connection = new SqlConnection("Data Source=DESKTOP-7QH0QHD\\SQLEXPRESS;Initial Catalog=BocchiStore;User ID=sa;Password=Oracle#1;Integrated Security=True;Connect Timeout=30;Encrypt=False");

}

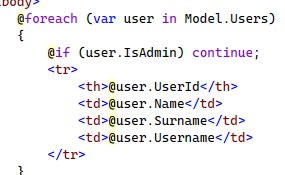
# Tutti gli iscritti in biblioteca

Abbiamo fatto una pagina che ci visualizzi tutti gli iscritti in biblioteca



Nella pagina Users.cshtml abbiamo creato una tabella che contiene tutti gli iscritti alla libreria

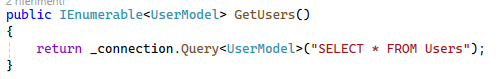
La pagina contiene del codice c# che richiama degli oggetti e dei metodi che alimentano le tabelle descritte.



public List<UserModel> Users => \_storage.GetUsers().ToList();

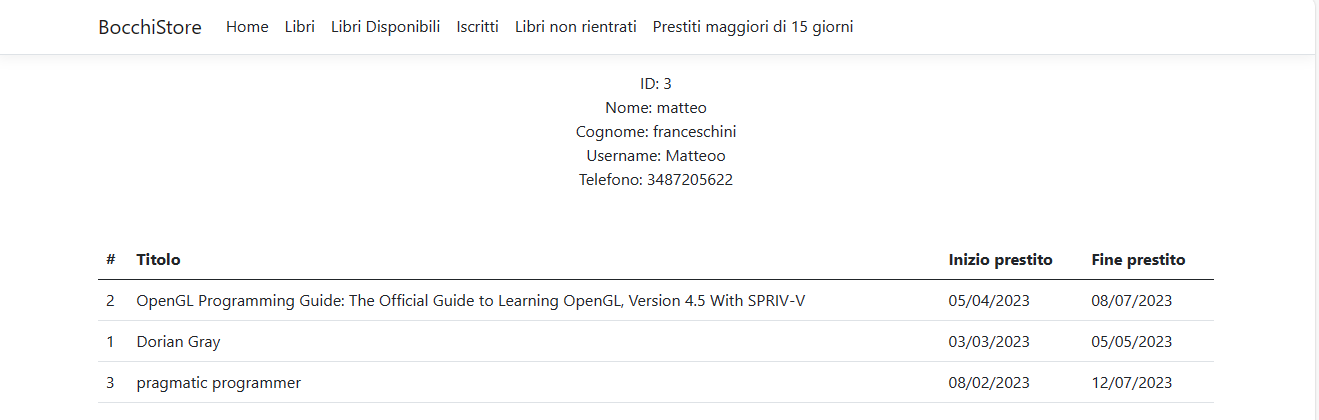
Per ogni utente contenuto nella lista Users aggiungo una riga di tabella che contiene Userid Name Surname Username.

I dati originano da una query definita da una classe DapperStorage che implementa il metodo di interfaccia GetUsers

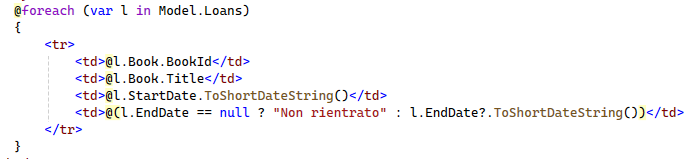


# Tutti i libri prestati ad un utente specifico in ordine cronologico

Dalla pagina degli iscritti per ogni iscritto avremmo un link che ci permette di visualizzare i dati dell’utente e visualizzare i prestiti in ordine cronologico decrescente dal più recente al più vecchio

Nella pagina vengono visualizzato i dati dell’Utente selezionato e viene creata una tabella che contiene una lista di presiti

La pagina contiene del codice c# che richiama degli oggetti e dei metodi che alimentano le tabelle descritte e se non c’è una fine del prestito stampa non rientrato



public List<LoanModelFull> Loans;

Per ogni prestito contenuto nella lista Loans aggiungo una riga di tabella che contiene BookId Title StartDate EndDate.

Al caricamento della pagina vengono instanziati gli oggetti che contengono i dati da visualizzare tramite id contenuto nel Url .

public void OnGet(int id)

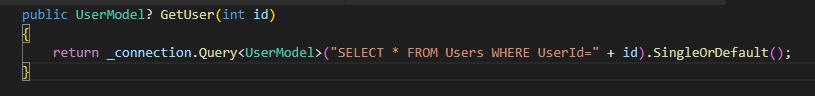
{

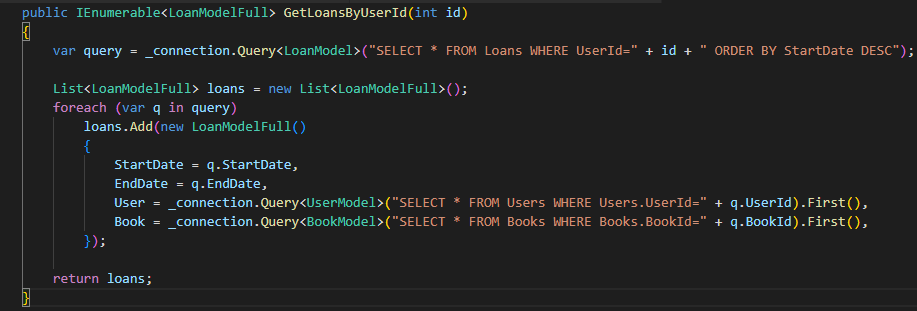
SelectedUser = \_storage.GetUser(id);

Loans = \_storage.GetLoansByUserId(id).ToList();

}

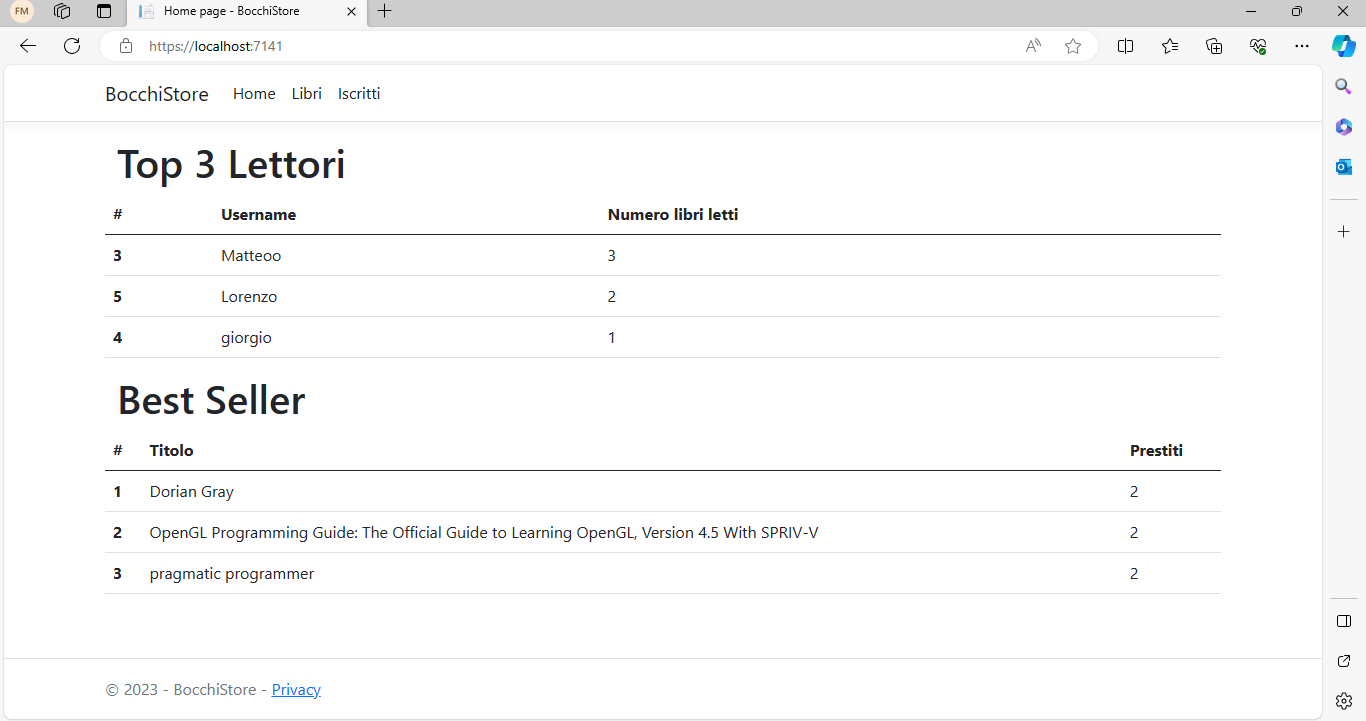
I dati originano da una query definita da una classe DapperStorage che implementa il metodo di interfaccia GetLoansByUserId





# Individua i primi tre lettori che hanno letto più libri

La pagina principale con cui si apre l’app ci mostra la top 3 lettori e anche la lista dei libri più prestati



Nel Front end abbiamo inserito una tabella che contiene la Top3 dei lettori per numero di prestiti eseguiti e dei libri più prestati

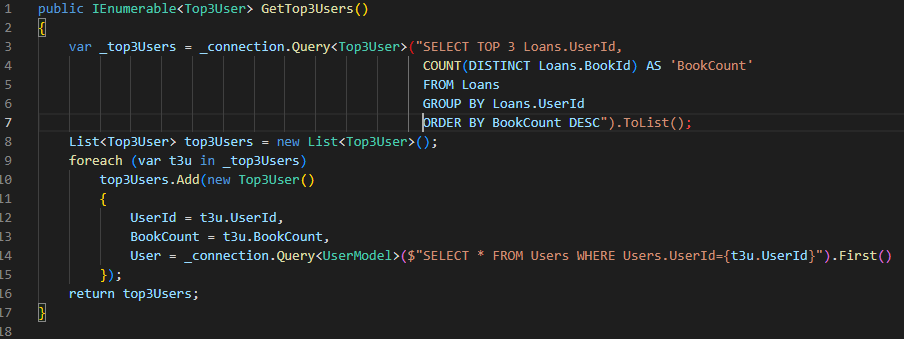
La pagina contiene del codice c# che richiama degli oggetti e dei metodi che alimentano le tabelle descritte.



public List<Top3User> Top3Users => \_storage.GetTop3Users().ToList();

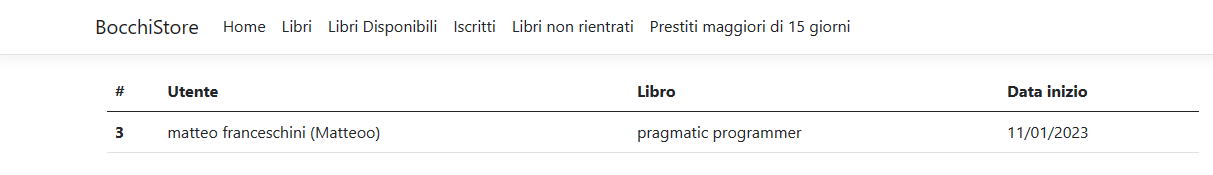
Per ogni utente contenuto nella lista Top3Users aggiungo una riga di tabella che contiene Userid UserUsername BookCount.

I dati originano da una query definita da una classe DapperStorage che implementa il metodo di interfaccia GetTop3Users



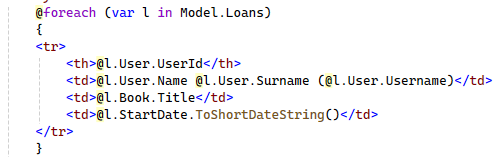
# Individua tutti i possessori dei libri non ancora rientrati e il titolo degli stessi

Questa pagina elenca i libri che non sono ancora rientrati dal prestito



Nel Front end abbiamo inserito una tabella che contiene la lista dei libri non restituiti con utente e data inizio prestito

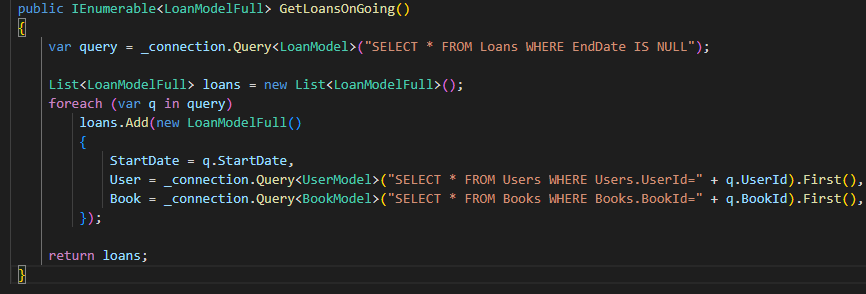
La pagina contiene del codice c# che richiama degli oggetti e dei metodi che alimentano le tabelle descritte.



public List<LoanModelFull> Loans => \_storage.GetLoansOnGoing().ToList();

Per ogni prestito contenuto nella lista prestiti aggiungo una riga di tabella che contiene Userid Name Title StartDate.

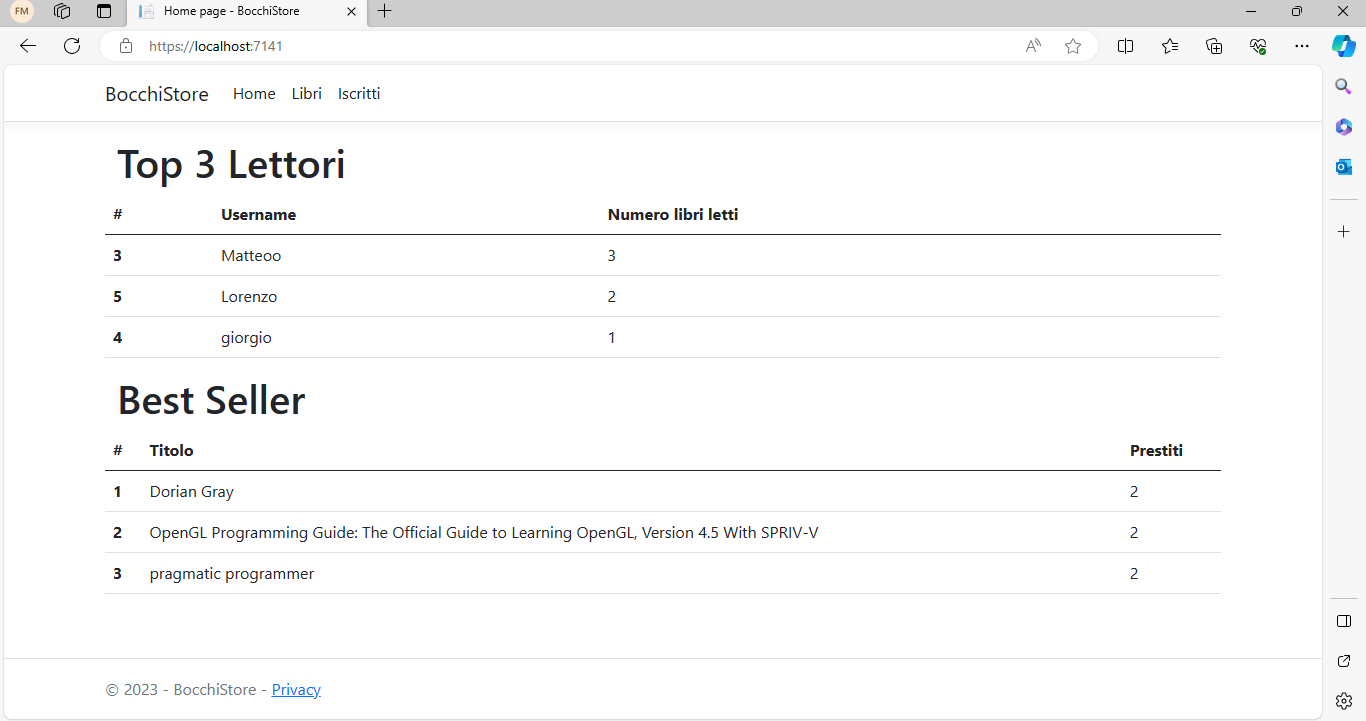
I dati originano da una query definita da una classe DapperStorage che implementa il metodo di interfaccia GetLoansOnGoing



# Dare lo storico dei libri chiesti in prestito da un utente indicando il periodo

Fai la classifica dei libri maggiormente prestati

La lista dei libri più prestati si trova nella pagina principale



Nel Front end abbiamo inserito una tabella che contiene la Top3 dei lettori per numero di prestiti eseguiti e dei libri più prestati

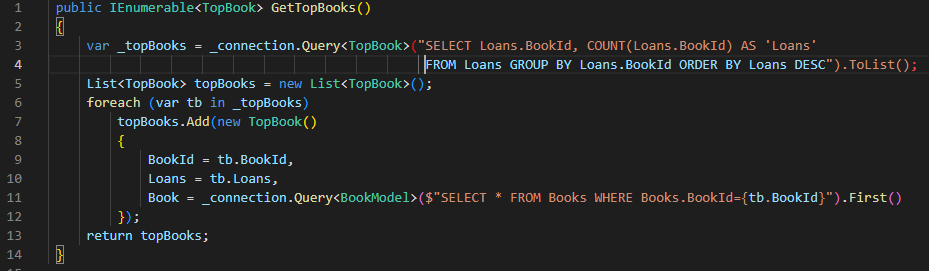
La pagina contiene del codice c# che richiama degli oggetti e dei metodi che alimentano le tabelle descritte.



public List<TopBook> TopBooks => \_storage.GetTopBooks().ToList();

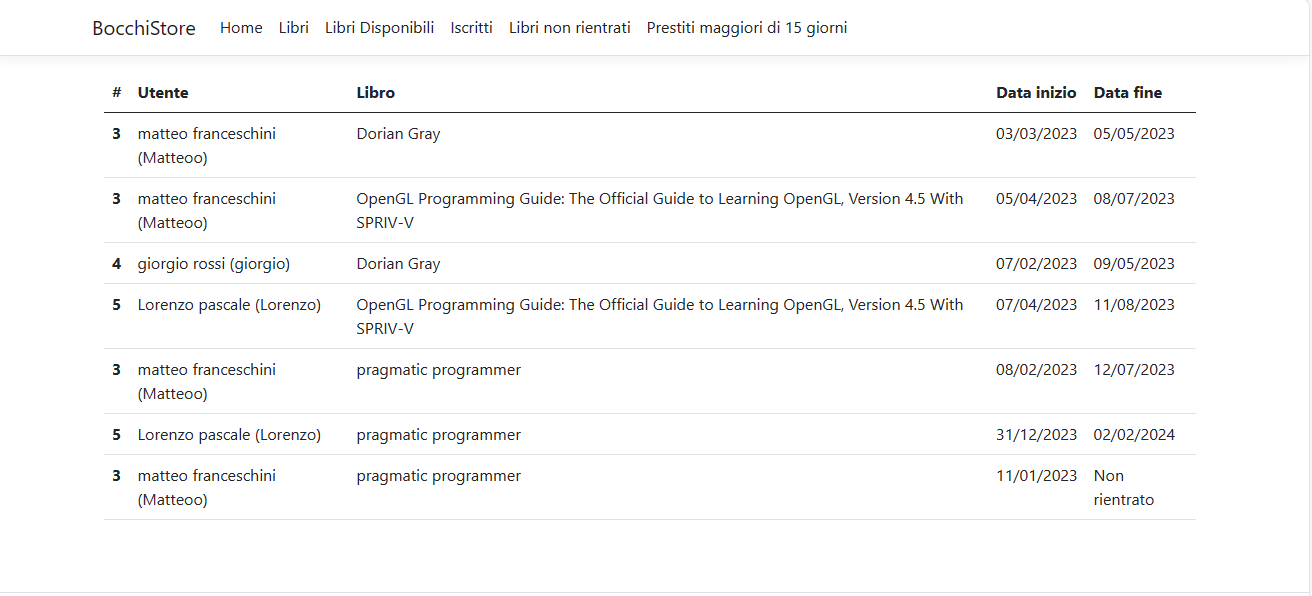
Per ogni book contenuto nella lista TopBook aggiungo una riga di tabella che contiene Bookid Title Loans

I dati originano da una query definita da una classe DapperStorage che implementa il metodo di interfaccia GetTopBooks



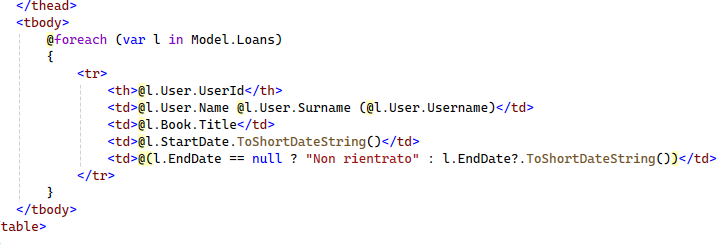
# Individua prestiti la cui data supera i 15gg

In questa pagina vedremo i prestiti che sono durati 15 giorni o più. Per i prestiti non ancora rientrati si considera come data termine la data di sistema.



Nel Frontend abbiamo inserito una tabella che mi permette di visualizzare i libri con prestiti maggiori di 15 giorni con i dati Utente e libro e il periodo del prestito

La pagina contiene del codice c# che richiama degli oggetti e dei metodi che alimentano le tabelle descritte.



public List<LoanModelFull> Loans => \_storage.GetLoansMore15Days().ToList();

Per ogni prestito contenuto nella lista Prestiti aggiungo una riga di tabella che contiene UserId Name Title StartDate EndDate

I dati originano da una query definita da una classe DapperStorage che implementa il metodo di interfaccia GetLoansMore15Days

