

Library Management System

Laboratorio di Basi di Dati (A.A. 2023/2024)

Specifiche del progetto e documentation

Juan Barea Rojo



UNIVERSITÀ DEGLI STUDI DI MILANO



INDEX

| | |
|--|-----------|
| Introduction..... | 3 |
| Manuale utente..... | 3 |
| Schema concettuale (ER)..... | 4 |
| Schema logico (relazionale) della base di dati..... | 5 |
| Explanation of the Schema logico..... | 6 |
| SQL used for the creation of the tables..... | 8 |
| Esauriente descrizione delle funzioni realizzate..... | 10 |
| check_loan_extension ()..... | 10 |
| statistiche_per_ogni_sede (integer)..... | 12 |
| ritardi_per_ogni_sede (integer)..... | 14 |
| update_copy_status ()..... | 16 |
| update_copy_status_to_available..... | 18 |
| Prove di funzionamento..... | 18 |



Introduction

The goal of this project is to develop a comprehensive database application for managing a library system, distributed across multiple locations. This application is designed to provide essential functionalities for both registered library readers and the librarians responsible for managing the system. By using this application, readers will have the ability to view detailed information about the catalogs maintained by the library, which include various books, authors, and available copies. Additionally, readers can enter loan requests for a specific set of books directly through the application, streamlining the borrowing process. One of the key features of this system is its ability to handle loan requests efficiently.

Manuale utente

For the development and management of the database, I used phpPgAdmin, a popular tool for PostgreSQL database management. I chose phpPgAdmin because it was the tool used in our classes, providing familiarity and consistency in the learning process. However, during my attempt to export the database (SQL Dump), the software did not allow the operation. This posed a significant challenge as it hindered the ability to back up and share the database structure and data. To address this issue, I will include the SQL statements used to create the tables, as well as the code for the functions and triggers implemented in the database, ensuring that the entire setup can be recreated if necessary.

Furthermore, the deployment of the web interface is available at the following link, where all implemented functions can be tested.

<https://studenti.di.unimi.it/juan.barearojo@studenti.unimi.it>.

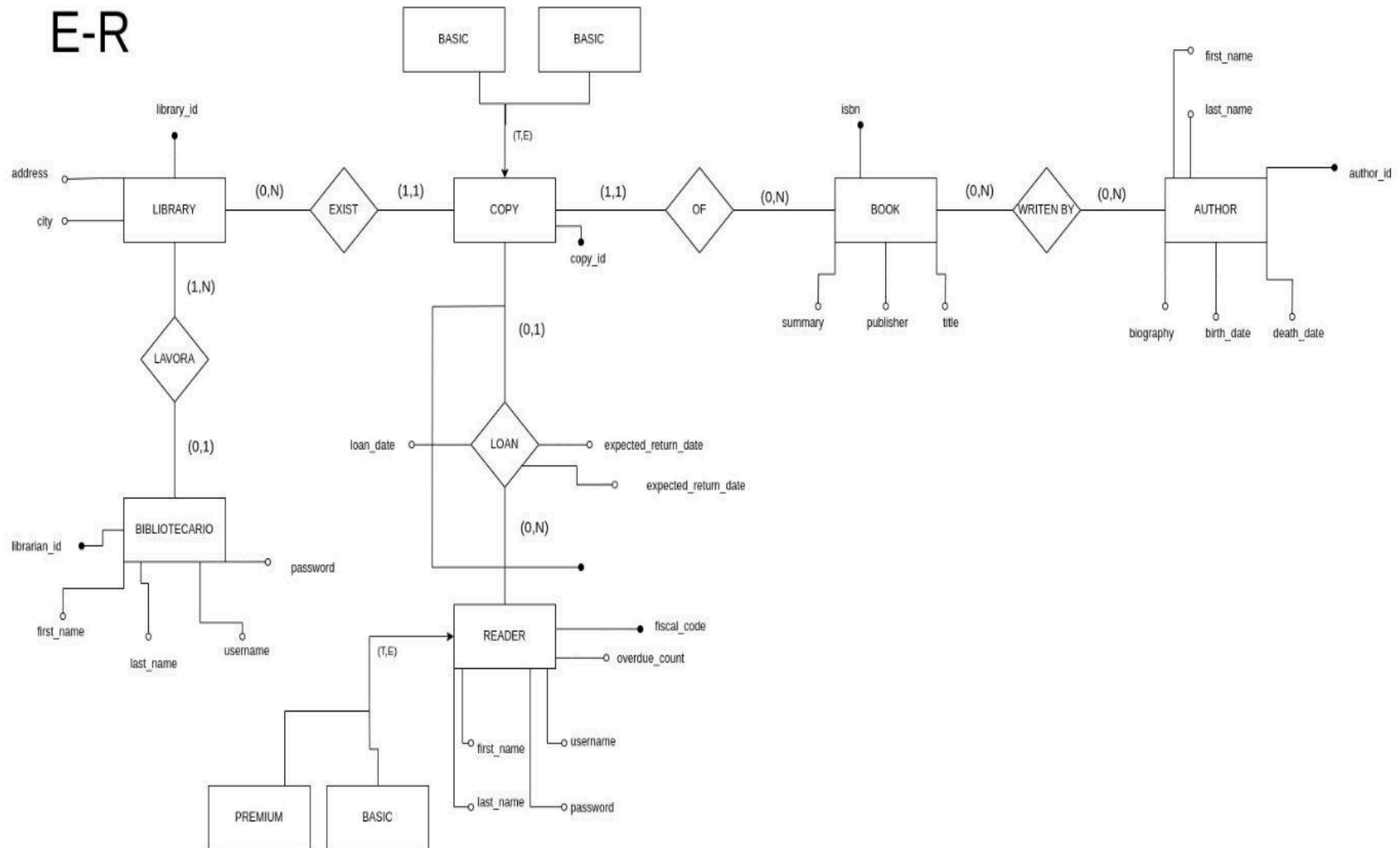
The following keys can be used to access the system

| USERNAME | PASSWORD | ROLE |
|----------|----------|---------------|
| user | user | Lettore |
| reader | reader | Lettore |
| admin | admin | Bibliotecario |



Schema concettuale (ER)

E-R





Schema logico (relazionale) della base di dati

| LIBRARY | |
|---------|-------------------------------|
| PK | <u>library_id INT</u> |
| | city VARCHAR(100) |
| | address VARCHAR(255) NOT NULL |

| BOOK | |
|------|---------------------------------|
| PK | <u>isbn VARCHAR(13)</u> |
| | title VARCHAR(255) NOT NULL |
| | summary TEXT |
| | publisher VARCHAR(255) NOT NULL |

| WRITTEN_BY | |
|------------|--|
| PK | <u>author_id INT</u> |
| PK | <u>isbn VARCHAR(13)</u> |
| | FOREIGN KEY (isbn) REFERENCES BOOK(isbn) |
| | FOREIGN KEY (author_id) REFERENCES AUTHOR(author_id) |

| COPY | |
|------|--|
| PK | <u>library_id INT</u> |
| PK | <u>isbn VARCHAR(13)</u> |
| PK | <u>copy_id INT DEFAULT nextval('copy_copy_id_seq'::regclass)</u> |
| | status VARCHAR(10) DEFAULT "available" |
| | FOREIGN KEY (isbn) REFERENCES BOOK(isbn) |
| | FOREIGN KEY (library_id) REFERENCES LIBRARY(library_id) |
| | CHECK (status IN ('available', 'loaned')) |

| LOAN | |
|------|---|
| PK | <u>fiscal_code VARCHAR(20)</u> |
| PK | <u>copy_id INT</u> |
| PK | <u>loan_date DATE DEFAULT CURRENT_TIMESTAMP</u> |
| | expected_return_date DATE DEFAULT CURRENT_DATE + INTERVAL |
| | actual_return_date DATE |
| | FOREIGN KEY (copy_id) REFERENCES COPY(copy_id) |
| | FOREIGN KEY (fiscal_code) REFERENCES READER(fiscal_code) |

| AUTHOR | |
|--------|---|
| PK | <u>author_id INT NOT NULL</u> |
| | first_name VARCHAR(100) NOT NULL |
| | last_name VARCHAR(100) NOT NULL |
| | birth_date DATE DEFAULT CURRENT_TIMESTAMP |
| | death_date DATE DEFAULT CURRENT_TIMESTAMP |
| | biography TEXT |

| READER | |
|--------|--|
| PK | <u>fiscal_code VARCHAR(20)</u> |
| | first_name VARCHAR(100) NOT NULL |
| | last_name VARCHAR(100) NOT NULL |
| | services VARCHAR(10) |
| | overdue_count INT DEFAULT 0 |
| | username VARCHAR(50) UNIQUE NOT NULL |
| | password VARCHAR(255) NOT NULL |
| | CHECK (services IN ('basic', 'premium')) |

| BIBLIOTECARIO | |
|---------------|---|
| PK | <u>library_id INT</u> |
| PK | <u>librarian_id INT</u> |
| | first_name VARCHAR(100) NOT NULL |
| | last_name VARCHAR(100) NOT NULL |
| | username VARCHAR(50) UNIQUE NOT NULL |
| | password VARCHAR(255) NOT NULL |
| | FOREIGN KEY (library_id) REFERENCES LIBRARY(library_id) |



Explanation of the Schema logico

LIBRARY

The *LIBRARY* table represents the different library branches within the system. Each library is uniquely identified by *library_id*. Additional attributes include:

- **city**: The city where the library is located.
- **address**: The address of the library.

BOOK

The *BOOK* table stores information about the books available in the library system. Each book is uniquely identified by *isbn*. Additional attributes include:

- **title**: The title of the book.
- **summary**: A brief summary of the book.
- **publisher**: The publisher of the book.

WRITTEN_BY

The *WRITTEN_BY* table represents the relationship between books and their authors. It has two foreign keys:

- **author_id**: References AUTHOR(author_id).
- **isbn**: References BOOK(isbn).

COPY

The *COPY* table allows for multiple copies of the same book to exist within the library system, each identified by a unique *copy_id*. Attributes include:

- **library_id**: References the LIBRARY where the copy is located.
- **isbn**: References the BOOK for which this is a copy.
- **copy_id**: Unique identifier for each copy, automatically incremented.
- **status**: Indicates whether the copy is 'available' or 'loaned'.

AUTHOR

The *AUTHOR* table stores information about the authors of the books. Each author is uniquely identified by *author_id*. Additional attributes include:

- **first_name**: The first name of the author.
- **last_name**: The last name of the author.
- **birth_date**: The birth date of the author.



- **death_date**: The death date of the author, if applicable.
- **biography**: A brief biography of the author.

LOAN

The **LOAN** table keeps track of the book loans in the system. The primary key consists of three values (**fiscal_code**, **copy_id**, and **loan_date**) to allow the same user to borrow the same copy on different dates. Attributes include:

- **fiscal_code**: References the **READER** who borrowed the book.
- **copy_id**: References the **COPY** that was borrowed.
- **loan_date**: The date when the book was borrowed.
- **expected_return_date**: The expected date by which the book should be returned.
- **actual_return_date**: The date when the book was actually returned.

READER

The **READER** table stores information about the library's registered readers. Each reader is uniquely identified by **fiscal_code**. Additional attributes include:

- **first_name**: The first name of the reader.
- **last_name**: The last name of the reader.
- **services**: Indicates the type of services the reader is subscribed to ('basic' or 'premium').
- **overdue_count**: The number of overdue books the reader has had.
- **username**: The unique username of the reader.
- **password**: The password for the reader's account.

BIBLIOTECARIO

The **BIBLIOTECARIO** table stores information about the librarians. Each librarian is uniquely identified by **librarian_id**. Additional attributes include:

- **library_id**: References the **LIBRARY** where the librarian works.
- **first_name**: The first name of the librarian.
- **last_name**: The last name of the librarian.
- **username**: The unique username of the librarian.
- **password**: The password for the librarian's account.



SQL used for the creation of the tables

```
-- Table LIBRARY
CREATE TABLE LIBRARY (
    library_id INT PRIMARY KEY,
    city VARCHAR(100),
    address VARCHAR(255) NOT NULL
);

-- Table BOOK
CREATE TABLE BOOK (
    isbn VARCHAR(13) PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    summary TEXT,
    publisher VARCHAR(255) NOT NULL
);

-- Table AUTHOR
CREATE TABLE AUTHOR (
    author_id INT PRIMARY KEY NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    birth_date DATE DEFAULT CURRENT_TIMESTAMP,
    death_date DATE DEFAULT CURRENT_TIMESTAMP,
    biography TEXT
);

-- Table WRITTEN_BY (Relationship between BOOK and AUTHOR)
CREATE TABLE WRITTEN_BY (
    isbn VARCHAR(13),
    author_id INT,
    PRIMARY KEY (isbn, author_id),
    FOREIGN KEY (isbn) REFERENCES BOOK(isbn) ON DELETE CASCADE,
    FOREIGN KEY (author_id) REFERENCES AUTHOR(author_id) ON DELETE
    CASCADE
);

-- Table COPY
CREATE TABLE COPY (
    copy_id INT PRIMARY KEY DEFAULT
nextval('copy_copy_id_seq'::regclass),
    isbn VARCHAR(13),
    library_id INT,
```




```
status VARCHAR(10) DEFAULT "available",
FOREIGN KEY (isbn) REFERENCES BOOK(isbn) ON DELETE CASCADE,
FOREIGN KEY (library_id) REFERENCES LIBRARY(library_id) ON DELETE
CASCADE,
CHECK (status IN ('available', 'loaned'))
);

-- Table READER
CREATE TABLE READER (
    fiscal_code VARCHAR(20) PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    services VARCHAR(10),
    overdue_count INT DEFAULT 0,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    CHECK (services IN ('basic', 'premium'))
);

-- Table BIBLIOTECARIO (Librarian who works in a library)
CREATE TABLE BIBLIOTECARIO (
    librarian_id INT PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    library_id INT,
    FOREIGN KEY (library_id) REFERENCES LIBRARY(library_id) ON DELETE
SET NULL
);

-- Table LOAN
CREATE TABLE LOAN (
    copy_id INT,
    fiscal_code VARCHAR(20),
    loan_date DATE DEFAULT CURRENT_TIMESTAMP,
    expected_return_date DATE DEFAULT CURRENT_DATE + INTERVAL '10 days',
    actual_return_date DATE,
    PRIMARY KEY(copy_id, fiscal_code, loan_date),
    FOREIGN KEY (copy_id) REFERENCES COPY(copy_id) ON DELETE CASCADE,
    FOREIGN KEY (fiscal_code) REFERENCES READER(fiscal_code) ON DELETE
CASCADE
);
```



Esauriente descrizione delle funzioni realizzate

check_loan_extension()

```
-- Proroga della durata di un prestito.
CREATE OR REPLACE FUNCTION check_loan_extension()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si el préstamo está vencido al momento de la
    actualización usando la fecha original de devolución
    IF OLD.expected_return_date <= CURRENT_DATE THEN
        RAISE EXCEPTION 'Cannot extend the loan period for an overdue
loan.';
    END IF;

    -- Permitir la actualización si no está vencido
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_update_loan
BEFORE UPDATE ON loan
FOR EACH ROW
WHEN (OLD.expected_return_date <> NEW.expected_return_date)
EXECUTE FUNCTION check_loan_extension();
```

The function *check_loan_extension* and the trigger *before_update_loan* work together to manage the extension of loan durations. The *check_loan_extension* function verifies whether the original expected return date of a loan is past the current date. If the loan is overdue, it raises an exception with the message *'Cannot extend the loan period for an overdue loan,'* thereby preventing the extension. If the loan is not overdue, the function allows the update to proceed by returning the new row values.

The *before_update_loan* trigger is activated before any update on the loan table. It executes the *check_loan_extension* function only if there is a change in the expected return date. This trigger ensures that any attempt to change the return date of a loan triggers the function to check for overdue status. If the loan is overdue, the update is blocked; otherwise, it is permitted. This mechanism enforces that overdue loans cannot be extended, maintaining the integrity and discipline of loan management.



```
select * from loan
```

| copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
|---------|-------------|------------|--------------------|----------------------|
| 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| 15 | FC000009J | 2024-06-02 | NULL | 2024-06-24 |
| 45 | 1 | 2024-06-25 | NULL | 2024-07-05 |
| 29 | 1 | 2024-06-25 | NULL | 2024-07-05 |
| 4 | 2 | 2024-06-24 | NULL | 2024-07-05 |
| 40 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| 59 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |

The update in the loan *copy_id* = 15 and *fiscal_code* = FC000009J cannot be extended because it has passed the *expected_return_date*.

Error de SQL:

ERROR: Cannot extend the loan period for an overdue loan.
CONTEXT: PL/pgSQL function check_loan_extension() line 5 at RAISE

En la sentencia:

UPDATE loan SET expected_return_date = CURRENT_DATE + INTERVAL '10 days' WHERE fiscal_code = 'FC000009J' AND copy_id = 15 AND loan_date = '2024-06-02'

The update in the loan *copy_id* = 29 and *fiscal_code* = 1 can be extended because it has not passed the *expected_return_date*.

```
SELECT * FROM "juan_barearojo"."loan";
```

| Acciones | copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
|-----------------|---------|-------------|------------|--------------------|----------------------|
| Editar Eliminar | 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| Editar Eliminar | 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| Editar Eliminar | 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| Editar Eliminar | 15 | FC000009J | 2024-06-02 | NULL | 2024-06-24 |
| Editar Eliminar | 45 | 1 | 2024-06-25 | NULL | 2024-07-05 |
| Editar Eliminar | 4 | 2 | 2024-06-24 | NULL | 2024-07-05 |
| Editar Eliminar | 40 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| Editar Eliminar | 59 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| Editar Eliminar | 29 | 1 | 2024-06-25 | NULL | 2024-07-07 |



statistiche_per_ogni_sede (integer)

```
CREATE OR REPLACE FUNCTION statistiche_per_ogni_sede(branch_id INTEGER)
RETURNS TABLE (
    total_copies BIGINT,
    total_isbns BIGINT,
    total_loans_in_progress BIGINT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        -- Número total de ejemplares gestionados por la sucursal
        (SELECT COUNT(*) FROM juan_barearojo.copy WHERE library_id =
branch_id) AS total_copies,

        -- Número total de códigos ISBN gestionados por la sucursal
        (SELECT COUNT(DISTINCT isbn) FROM juan_barearojo.copy WHERE
library_id = branch_id) AS total_isbns,

        -- Número total de préstamos en curso para los volúmenes
mantenidos por la sucursal
        (SELECT COUNT(*) FROM juan_barearojo.copy WHERE library_id =
branch_id AND status = 'loaned') AS total_loans_in_progress;
END;
$$ LANGUAGE plpgsql;
```

The *statistiche_per_ogni_sede* function is designed to return statistics for a specific library branch identified by *branch_id*. It returns a table with three columns: *total_copies*, *total_isbns*, and *total_loans_in_progress*. The function executes a query that calculates the total number of copies managed by the branch, the total number of distinct ISBNs, and the total number of copies currently loaned out.



```
SELECT * FROM copy where status = 'loaned'
```

Enviar

| copy_id | isbn | library_id | status |
|---------|---------------|------------|--------|
| 4 | 9780345535528 | 4 | loaned |
| 15 | 9781849700611 | 5 | loaned |
| 45 | 9781849706279 | 5 | loaned |
| 29 | 9781849703551 | 4 | loaned |
| 40 | 9781849705517 | 5 | loaned |
| 59 | 9781111111111 | 5 | loaned |

6 fila(s)

```
SELECT * FROM juan_barearojo.statistiche_per_ogni_sede(3)
```

Enviar

| total_copies | total_isbns | total_loans_in_progress |
|--------------|-------------|-------------------------|
| 45 | 31 | 0 |

```
SELECT * FROM juan_barearojo.statistiche_per_ogni_sede(5)
```

Enviar

| total_copies | total_isbns | total_loans_in_progress |
|--------------|-------------|-------------------------|
| 33 | 30 | 4 |

```
SELECT * FROM juan_barearojo.statistiche_per_ogni_sede(4)
```

Enviar

| total_copies | total_isbns | total_loans_in_progress |
|--------------|-------------|-------------------------|
| 34 | 31 | 2 |



ritardi_per_ogni_sede (integer)

```
--Ritardi per ogni sede.
CREATE OR REPLACE FUNCTION ritardi_per_ogni_sede(branch_id INTEGER)
RETURNS TABLE (
    copy_id INTEGER,
    isbn VARCHAR(13),
    title VARCHAR(255),
    reader_fiscal_code VARCHAR(20),
    reader_name VARCHAR(200),
    expected_return_date DATE,
    actual_return_date DATE
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        c.copy_id,
        c.isbn,
        b.title,
        r.fiscal_code AS reader_fiscal_code,
        CAST(CONCAT(r.first_name, ' ', r.last_name) AS VARCHAR(200)) AS
reader_name,
        l.expected_return_date,
        l.actual_return_date
    FROM
        juan_barearojo.loan l
    JOIN
        juan_barearojo.copy c ON l.copy_id = c.copy_id
    JOIN
        juan_barearojo.book b ON c.isbn = b.isbn
    JOIN
        juan_barearojo.reader r ON l.fiscal_code = r.fiscal_code
    WHERE
        c.library_id = branch_id
        AND l.actual_return_date IS NULL
        AND l.expected_return_date < CURRENT_DATE;
END;
$$ LANGUAGE plpgsql;
```



The *ritardi_per_ogni_sede* function is designed to return information about overdue loans for a specific library branch identified by *branch_id*. It returns a table with columns including *copy_id*, *isbn*, *title*, *reader_fiscal_code*, *reader_name*, *expected_return_date*, and *actual_return_date*. The function executes a query that joins the loan, copy, book, and reader tables to gather all necessary information about overdue loans. It selects the relevant columns and filters results to include only loans from the specified branch, loans that have not been returned yet, and loans that are overdue.

The only loan that is on 'ritardo' is the one with *fiscal_code* = FC000009J, *copy_id* = 15 and *loan_date* = 2024-06-02

```
SELECT * FROM "juan_barearojo"."loan";
```

Enviar

| Acciones | | copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
|----------|----------|---------|-------------|------------|--------------------|----------------------|
| Editar | Eliminar | 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| Editar | Eliminar | 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| Editar | Eliminar | 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| Editar | Eliminar | 15 | FC000009J | 2024-06-02 | NULL | 2024-06-24 |
| Editar | Eliminar | 45 | 1 | 2024-06-25 | NULL | 2024-07-05 |
| Editar | Eliminar | 4 | 2 | 2024-06-24 | NULL | 2024-07-05 |
| Editar | Eliminar | 40 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| Editar | Eliminar | 59 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| Editar | Eliminar | 29 | 1 | 2024-06-25 | NULL | 2024-07-07 |

```
SELECT * FROM juan_barearojo.ritardi_per_ogni_sede(5)
```

Enviar

| copy_id | isbn | title | reader_fiscal_code | reader_name | expected_return_date | actual_return_date |
|---------|---------------|--------------|--------------------|---------------|----------------------|--------------------|
| 15 | 9781849700611 | Horus Rising | FC000009J | Jaghatai Khan | 2024-06-24 | NULL |



update_copy_status ()

```
-- Update status after loaning
CREATE OR REPLACE FUNCTION update_copy_status()
RETURNS TRIGGER AS $$
BEGIN
    -- Actualizar el estado del ejemplar a 'loaned'
    UPDATE copy
    SET status = 'loaned'
    WHERE copy_id = NEW.copy_id;

    -- Permitir la inserción en la tabla loan
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Crear el trigger
CREATE TRIGGER after_insert_loan
AFTER INSERT ON loan
FOR EACH ROW
EXECUTE FUNCTION update_copy_status();
```

The **update_copy_status** function is designed to update the status of a book copy to 'loaned' whenever a new loan is inserted into the loan table. The function executes an update on the **copy** table, setting the status to 'loaned' for the corresponding **copy_id** of the newly inserted loan. After performing this update, the function allows the insertion of the new loan record by returning NEW. This function is triggered by the **after_insert_loan** trigger, which activates after each row insertion into the loan table, ensuring that the status of the borrowed book is appropriately updated to reflect its loaned state.



```
SELECT * FROM "juan_barearojo"."copy";
```

Enviar

| Acciones | | copy_id | isbn | library_id | status |
|----------|----------|---------|---------------|------------|-----------|
| Editar | Eliminar | 5 | 9780345535528 | 5 | available |
| Editar | Eliminar | 6 | 9780060850524 | 1 | available |
| Editar | Eliminar | 7 | 9780060850524 | 2 | available |
| Editar | Eliminar | 8 | 9780060850524 | 3 | available |
| Editar | Eliminar | 9 | 9780060850524 | 4 | available |
| Editar | Eliminar | 10 | 9780060850524 | 5 | available |
| Editar | Eliminar | 11 | 9781849700611 | 1 | available |
| Editar | Eliminar | 12 | 9781849700611 | 2 | available |
| Editar | Eliminar | 13 | 9781849700611 | 3 | available |
| Editar | Eliminar | 14 | 9781849700611 | 4 | available |
| Editar | Eliminar | 16 | 9781849701021 | 1 | available |

AFTER ->INSERT INTO loan (copy_id, fiscal_code) VALUES (5, 2);

```
SELECT * FROM copy where copy_id = 5
```

Enviar

| copy_id | isbn | library_id | status |
|---------|---------------|------------|--------|
| 5 | 9780345535528 | 5 | loaned |



update_copy_status_to_available

```
-- Update status after returning copy
CREATE OR REPLACE FUNCTION update_copy_status_to_available()
RETURNS TRIGGER AS $$
BEGIN
    -- Actualizar el estado del ejemplar a 'available'
    UPDATE copy
    SET status = 'available'
    WHERE copy_id = NEW.copy_id;

    -- Permitir la actualización en la tabla loan
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_update_loan_return
AFTER UPDATE ON loan
FOR EACH ROW
WHEN (OLD.actual_return_date IS NULL AND NEW.actual_return_date IS NOT
NULL)
EXECUTE FUNCTION update_copy_status_to_available();
```

The `update_copy_status_to_available` function updates the status of a book copy to 'available' when a loan record is updated to indicate that the book has been returned. This function updates the copy table, setting the status to 'available' for the `copy_id` of the returned book. The function is triggered by the `after_update_loan_return` trigger, which activates after an update on the loan table for each row where the `actual_return_date` changes from NULL to a non-null value, ensuring that the status of the book is correctly updated to reflect its availability.



```
SELECT * FROM loan
```

Enviar

| copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
|---------|-------------|------------|--------------------|----------------------|
| 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| 15 | FC000009J | 2024-06-02 | NULL | 2024-06-24 |
| 45 | 1 | 2024-06-25 | NULL | 2024-07-05 |
| 4 | 2 | 2024-06-24 | NULL | 2024-07-05 |
| 40 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| 59 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| 29 | 1 | 2024-06-25 | NULL | 2024-07-07 |
| 5 | 2 | 2024-06-25 | NULL | 2024-07-05 |

AFTER -> UPDATE loan SET actual_return_date = CURRENT_TIMESTAMP
WHERE copy_id = 5 AND loan_date = '2024-06-25'

```
select * from copy where copy_id = 5
```

Enviar

| copy_id | isbn | library_id | status |
|---------|---------------|------------|-----------|
| 5 | 9780345535528 | 5 | available |

```
select * from loan where copy_id = 5
```

Enviar

| copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
|---------|-------------|------------|--------------------|----------------------|
| 5 | 2 | 2024-06-25 | 2024-06-25 | 2024-07-05 |



update_overdue_count()

```
-- Ritardi nelle restituzioni.
CREATE OR REPLACE FUNCTION update_overdue_count()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si la devolución es tarde
    IF NEW.actual_return_date > NEW.expected_return_date THEN
        -- Incrementar el contador de retrasos del lector
        UPDATE reader
        SET overdue_count = overdue_count + 1
        WHERE fiscal_code = (SELECT fiscal_code FROM loan WHERE copy_id
= NEW.copy_id);
    END IF;

    -- Permitir la actualización de la fila
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_update_loan
AFTER UPDATE ON loan
FOR EACH ROW
EXECUTE FUNCTION update_overdue_count();
```

The **update_overdue_count** function increments the overdue count for a reader if a book is returned late. When a loan record is updated with an **actual_return_date** that is later than the **expected_return_date**, the function increases the **overdue_count** in the reader table for the corresponding reader. This function is triggered by the **after_update_loan** trigger, which activates after each row update on the loan table, ensuring that any late returns are correctly reflected in the reader's overdue count.



| Acciones | | copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
|----------|----------|---------|-------------|------------|--------------------|----------------------|
| Editar | Eliminar | 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| Editar | Eliminar | 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| Editar | Eliminar | 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| Editar | Eliminar | 15 | FC000009J | 2024-06-02 | NULL | 2024-06-24 |
| Editar | Eliminar | 45 | 1 | 2024-06-25 | NULL | 2024-07-05 |
| Editar | Eliminar | 4 | 2 | 2024-06-24 | NULL | 2024-07-05 |
| Editar | Eliminar | 40 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| Editar | Eliminar | 59 | FC000001E | 2024-06-25 | NULL | 2024-07-05 |
| Editar | Eliminar | 29 | 1 | 2024-06-25 | NULL | 2024-07-07 |
| Editar | Eliminar | 5 | 2 | 2024-06-25 | 2024-06-25 | 2024-07-05 |
| Editar | Eliminar | 43 | FC000001E | 2024-06-03 | NULL | 2024-06-13 |
| Editar | Eliminar | 23 | 4 | 2024-05-02 | NULL | 2024-05-13 |
| Editar | Eliminar | 20 | FC000003R | 2024-04-02 | NULL | 2024-05-02 |

```
select * from reader where fiscal_code = '4'
```

| fiscal_code | first_name | last_name | services | overdue_count | username | password |
|-------------|------------|-----------|----------|---------------|---------------|---------------|
| 4 | reader | Premium | premium | 0 | readerPremium | readerPremium |

AFTER -> UPDATE loan SET actual_return_date = CURRENT_TIMESTAMP
WHERE copy_id = 23 AND loan_date = '2024-05-02'

```
SELECT * FROM READER WHERE fiscal_code='4'
```

| fiscal_code | first_name | last_name | services | overdue_count | username | password |
|-------------|------------|-----------|----------|---------------|---------------|---------------|
| 4 | reader | Premium | premium | 1 | readerPremium | readerPremium |



check_max_loans(fiscal_code_given TEXT)

```
CREATE OR REPLACE FUNCTION check_max_loans(fiscal_code_given TEXT)
RETURNS VOID AS $$
DECLARE
    loan_count INTEGER;
    max_loans INTEGER;
BEGIN
    -- Obtener el número actual de préstamos del lector
    SELECT COUNT(*)
    INTO loan_count
    FROM loan
    WHERE fiscal_code = fiscal_code_given AND actual_return_date IS
NULL;

    -- Determinar el máximo de préstamos permitidos según la categoría
del lector
    IF (SELECT services FROM reader WHERE fiscal_code =
fiscal_code_given) = 'basic' THEN
        max_loans := 3;
    ELSE
        max_loans := 5;
    END IF;

    -- Verificar si el lector ha alcanzado el número máximo de préstamos
permitidos
    IF loan_count >= max_loans THEN
        RAISE EXCEPTION 'The reader has reached the maximum number of
loans.';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

The *check_max_loans* function ensures that a reader has not exceeded their maximum allowed loans before issuing a new one. It takes a reader's *fiscal_code_given* as input, retrieves the current number of active loans for that reader, and determines the maximum loans allowed based on their service category (basic or premium). If the reader's current loan count meets or exceeds the maximum allowed (3 for basic, 5 for premium), the function raises an exception with the message 'The reader has reached the maximum number of loans,' thus preventing the issuance of additional loans.



```
select * from loan where fiscal_code = '2'
```

| Enviar | | | | |
|---------|-------------|------------|--------------------|----------------------|
| copy_id | fiscal_code | loan_date | actual_return_date | expected_return_date |
| 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| 4 | 2 | 2024-06-24 | NULL | 2024-07-05 |
| 5 | 2 | 2024-06-25 | 2024-06-25 | 2024-07-05 |
| 64 | 2 | 2024-06-25 | NULL | 2024-07-05 |
| 119 | 2 | 2024-06-25 | NULL | 2024-07-05 |

7 fila(s)

```
select * from reader where fiscal_code = '2'
```

| Enviar | | | | | | |
|-------------|------------|-----------|----------|---------------|----------|----------|
| fiscal_code | first_name | last_name | services | overdue_count | username | password |
| 2 | prueba2 | prueba2 | basic | 0 | prueba2 | prueba2 |

Error de SQL:

```
ERROR: The reader has reached the maximum number of loans.
CONTEXT: PL/pgSQL function check_max_loans(text) line 21 at RAISE
SQL statement "SELECT check_max_loans(NEW.fiscal_code)"
PL/pgSQL function check_loan_conditions() line 4 at PERFORM
```

En la sentencia:

```
INSERT INTO "juan_barearojo"."loan" ("copy_id","fiscal_code","loan_date","actual_return_date","expected_return_date")
VALUES ('117','2',CURRENT_TIMESTAMP,NULL,(CURRENT_DATE + '10 days'::interval))
```

The loan table shows that the reader with `fiscal_code` '2' currently has multiple active loans (indicated by NULL in the `actual_return_date` column). The reader table shows that this reader has a 'basic' service, which typically allows a maximum of 3 concurrent loans. The attempted insertion of a new loan fails because the reader already has 3 or more active loans, violating the maximum allowed loans condition for 'basic' service readers.



check_overdue_count(fiscal_code_given TEXT)

```
CREATE OR REPLACE FUNCTION check_overdue_count(fiscal_code_given TEXT)
RETURNS VOID AS $$
BEGIN
    -- Chequear el overdue_count del lector que intenta tomar prestado
    un libro
    IF (SELECT overdue_count FROM reader WHERE fiscal_code =
fiscal_code_given) >= 5 THEN
        -- Si el lector tiene 5 o más préstamos vencidos, lanzar una
        excepción
        RAISE EXCEPTION 'Cannot lend to a reader with 5 or more overdue
returns.';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

The `check_overdue_count` function prevents a reader with five or more overdue loans from borrowing additional books. It takes a `fiscal_code_given` as input, checks the `overdue_count` of the reader from the reader table, and raises an exception with the message 'Cannot lend to a reader with 5 or more overdue returns' if the count is five or more.

```
select * from reader where fiscal_code ='FC000009J'
```

Enviar

| fiscal_code | first_name | last_name | services | overdue_count | username | password |
|-------------|------------|-----------|----------|---------------|----------|------------|
| FC000009J | Jaghatai | Khan | premium | 8 | jaghatai | white scar |

Error de SQL:

```
ERROR: Cannot lend to a reader with 5 or more overdue returns.
CONTEXT: PL/pgSQL function check_overdue_count(text) line 6 at RAISE
SQL statement "SELECT check_overdue_count(NEW.fiscal_code)"
PL/pgSQL function check_loan_conditions() line 7 at PERFORM
```

En la sentencia:

```
INSERT INTO "juan_barearojo"."loan" ("copy_id","fiscal_code","loan_date","actual_return_date","expected_return_date")
VALUES ('76','FC000009J',CURRENT_TIMESTAMP,NULL,(CURRENT_DATE + '10 days'::interval))
```




check_loan_conditions()

```
CREATE OR REPLACE FUNCTION check_loan_conditions()
RETURNS TRIGGER AS $$
BEGIN
    -- Llamar a la sub-función para chequear el máximo de préstamos
    PERFORM check_max_loans(NEW.fiscal_code);

    -- Llamar a la sub-función para chequear el número de préstamos
    vencidos
    PERFORM check_overdue_count(NEW.fiscal_code);

    -- Si ambos chequeos pasan, permitir la inserción
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_loan
BEFORE INSERT ON loan
FOR EACH ROW
EXECUTE FUNCTION check_loan_conditions();
```

The **check_loan_conditions** function is designed to ensure that certain conditions are met before a new loan is inserted into the loan table. This function first calls **check_max_loans**, which verifies if the reader has reached their maximum allowed number of loans. It then calls **check_overdue_count** to check if the reader has five or more overdue loans. If either of these conditions fails, an exception is raised, preventing the insertion of the new loan.

The **before_insert_loan** trigger is set to execute the **check_loan_conditions** function before each row insertion into the loan table. This ensures that every new loan is subject to these checks, maintaining adherence to the library's borrowing policies. By enforcing these conditions, the system helps manage loan limits and overdue returns effectively, promoting responsible borrowing behavior among readers.



Prove di funzionamento LETTORE

A screenshot of a web browser window displaying a login page. The browser's address bar shows the URL "https://studenti.di.unimi.it/juan.bareaajo@studenti.unimi.it/". The page has a dark header with the title "Login". Below the header, there are three input fields: "Username:", "Password:", and "Role:". The "Role:" field is a dropdown menu with "Lettore" selected. A blue "Login" button is positioned below the "Role:" field. The browser's tab bar shows several open tabs, including "BIG DATA", "MILAN", "HACKING", "Desarrollo Web", "Data engineri...", "PACIENTES", "Blackbox AI C...", "97 cosas que t...", "Problems - Lee...", "The crypto wal...", "Relational Dat...", "pandas docu...", and "Otros favoritos".

Login

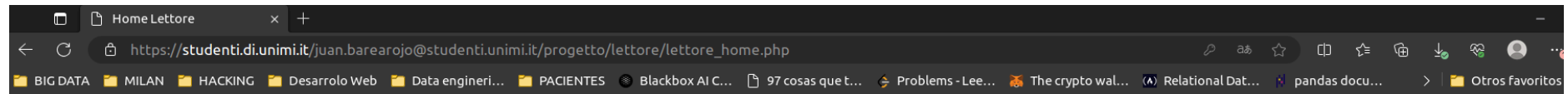
Username:

Password:

Role:

Lettore

Login



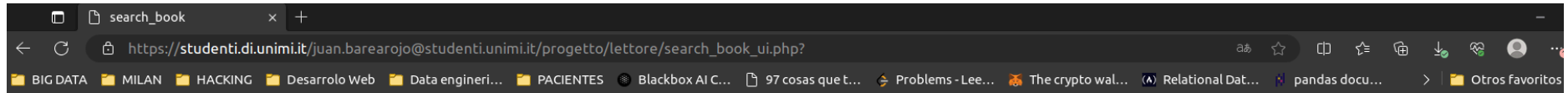
Ciao user

Search a book

Your Profile

Loans

Logout



Search book by ISBN or Title

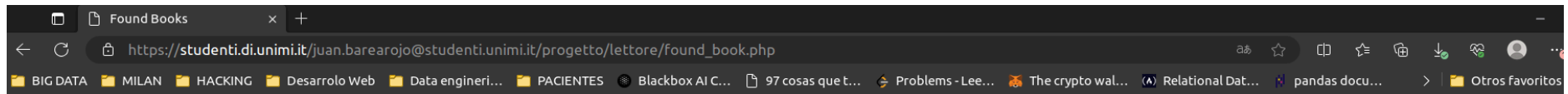
ISBN:

Search by ISBN

Title:

Search by title

Go HOME



Books Information

ISBN: 9781849700611

Title: Horus Rising

Author: Dan Abnett

Publisher: Black Library

Summary: The seeds of heresy are sown in the Imperium of Man.

Copy ID: 11

Status: available

Library Id: 1

Address: 123 Fictional St.

ISBN: 9781849700611

Title: Horus Rising

Author: Dan Abnett

Publisher: Black Library

Summary: The seeds of heresy are sown in the Imperium of Man.

Copy ID: 12

Status: available

Library Id: 2

Address: 456 Imaginary Ave.



Found Books

https://studenti.di.unimi.it/juan.bareaajo@studenti.unimi.it/progetto/lettore/found_book.php

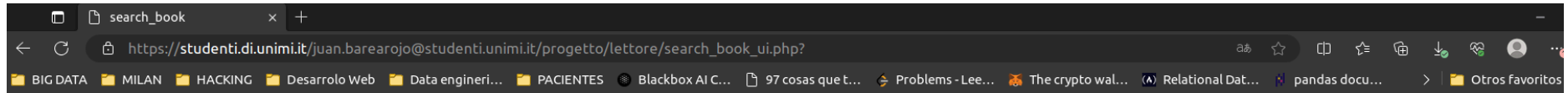
BIG DATA MILAN HACKING Desarrollo Web Data engineri... PACIENTES Blackbox AI C... 97 cosas que t... Problems - Lee... The crypto wal... Relational Dat... pandas docu... Otros favoritos

Address: 456 Imaginary Ave.

ISBN: 9781849700611
Title: Horus Rising
Author: Dan Abnett
Publisher: Black Library
Summary: The seeds of heresy are sown in the Imperium of Man.
Copy ID: 13
Status: available
Library Id: 3
Address: 789 Fantasy Blvd.

ISBN: 9781849700611
Title: Horus Rising
Author: Dan Abnett
Publisher: Black Library
Summary: The seeds of heresy are sown in the Imperium of Man.
Copy ID: 14
Status: available
Library Id: 4
Address: 1 Sorcery Path

Search Another Book



Search book by ISBN or Title

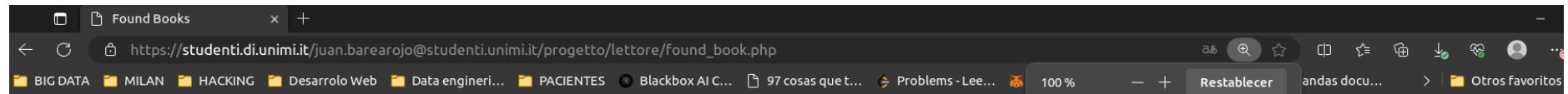
ISBN:

Search by ISBN

Title:

Search by title

Go HOME



Books Information

ISBN: 9780345535528

Title: Dune

Author: Frank Herbert

Publisher: Ace

Summary: A science fiction novel about a desert planet and its valuable spice.

Copy ID: 5

Status: available

Library Id: 5

Address: 999 Chaos Realm

ISBN: 9780345535528

Title: Dune

Author: Frank Herbert

Publisher: Ace

Summary: A science fiction novel about a desert planet and its valuable spice.

Copy ID: 3

Status: available

Library Id: 3

Address: 789 Fantasy Blvd.



Found Books

https://studenti.di.unimi.it/juan.bareaajo@studenti.unimi.it/progetto/lettore/found_book.php

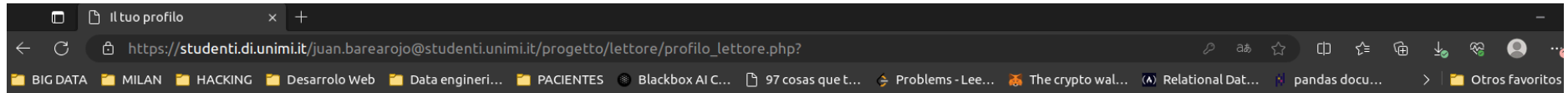
BIG DATA MILAN HACKING Desarrollo Web Data engineri... PACIENTES Blackbox AI C... 97 cosas que t... Problems - Lee... The crypto wal... Relational Dat... pandas docu... Otros favoritos

Address: 789 Fantasy Blvd.

ISBN: 9780345535528
Title: Dune
Author: Frank Herbert
Publisher: Ace
Summary: A science fiction novel about a desert planet and its valuable spice.
Copy ID: 1
Status: available
Library Id: 1
Address: 123 Fictional St.

ISBN: 9780345535528
Title: Dune
Author: Frank Herbert
Publisher: Ace
Summary: A science fiction novel about a desert planet and its valuable spice.
Copy ID: 2
Status: available
Library Id: 2
Address: 456 Imaginary Ave.

Search Another Book



Il tuo profilo

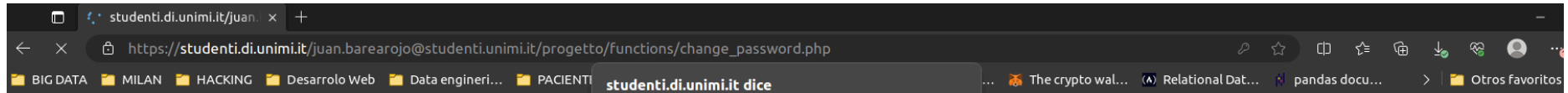
Il tuo Username: [user](#)

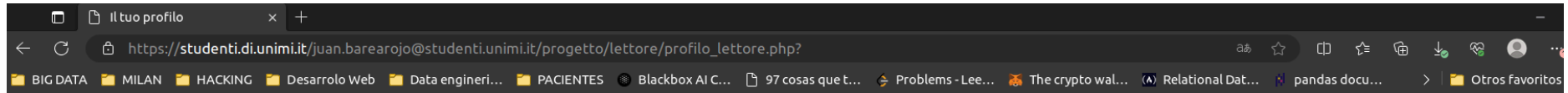
Il tuo Password: [user](#)

New Password:

Change Password

Go HOME





Il tuo profilo

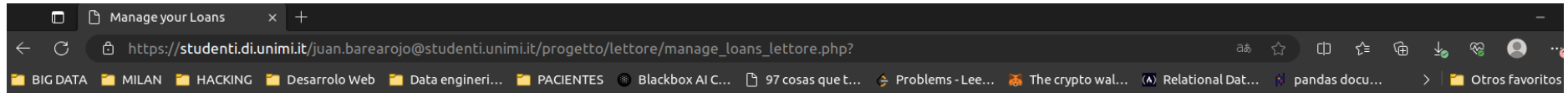
Il tuo Username: [user](#)

Il tuo Password: [user1](#)

New Password:

Change Password

Go HOME

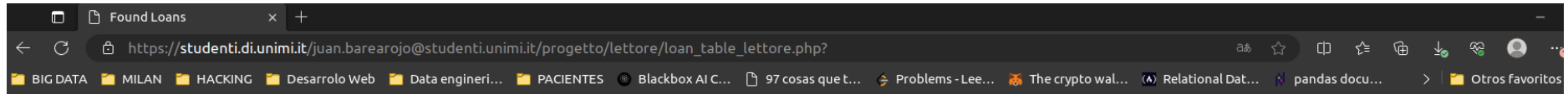


Manage your Loans

All your loans

Ask for a loan

Go HOME



Loans Information

| Copy ID | Fiscal Code | Loan Date | Actual Return Date | Expected Return Date |
|---------|-------------|------------|--------------------|----------------------|
| 45 | 1 | 2024-06-25 | | 2024-07-05 |
| 29 | 1 | 2024-06-25 | | 2024-07-07 |

[Go HOME](#)



Prove di funzionamento BIBLIOTECARIO

Browser window showing the Login page of the Library Management System.

URL: <https://studenti.di.unimi.it/juan.barearojo@studenti.unimi.it/index.php>

Browser tabs: BIG DATA, MILAN, HACKING, Desarrollo Web, Data engineri..., PACIENTES, Blackbox AI C..., 97 cosas que t..., Problems - Lee..., The crypto wal..., Relational Dat..., pandas docu..., Otros favoritos

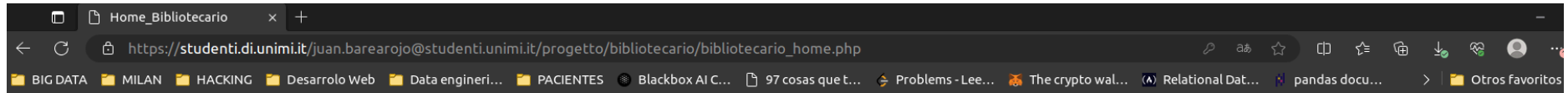
Login

Username:

Password:

Role:

Login



Ciao, admin!

Manage books

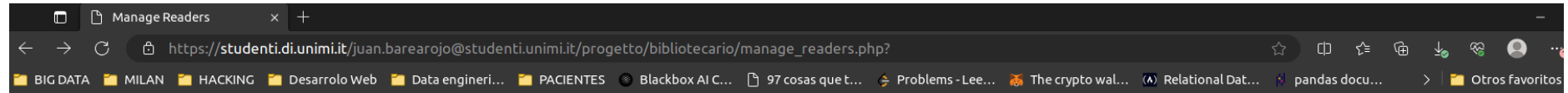
Il tuo profilo

Manage Readers

See statistics

Manage loans

Logout



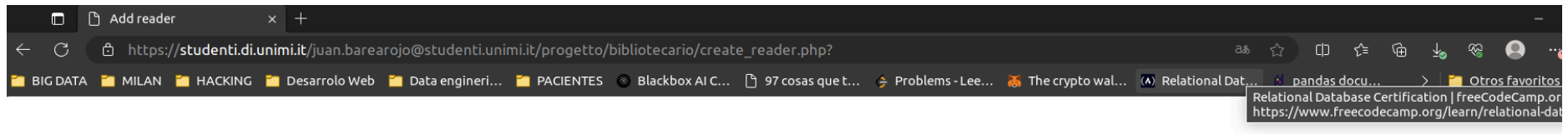
Manage Readers

Create a reader account

Eliminate reader debt

Eliminate reader from system

Go HOME



Add reader

Fiscal Code:

First Name:

Last Name:

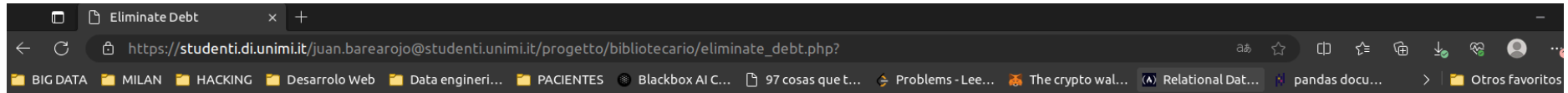
Username:

Password:

Services:

Add reader

Go HOME

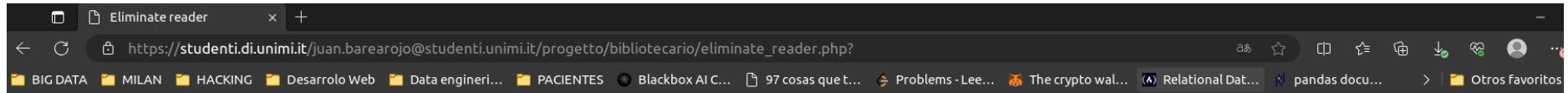


Eliminate debt from a reader

Fiscal code of the reader:

Eliminate debt

Go HOME

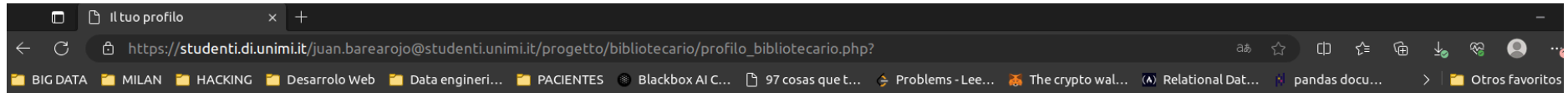


Eliminate a reader from the database

Fiscal code of the reader yo want to eliminate:

Eliminate reader

Go HOME



Il tuo profilo

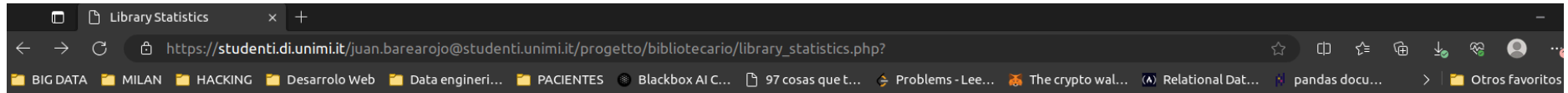
Il tuo Username: [admin](#)

Il tuo Password: [admin](#)

New Password:

Change Password

Go HOME



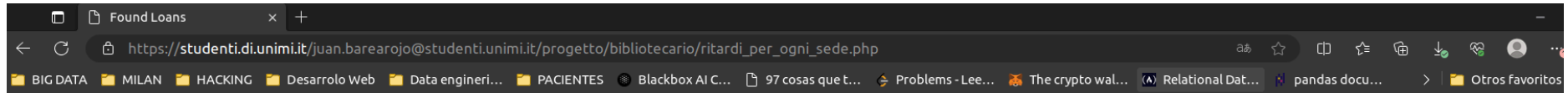
Get Library Statistics

Library ID:

Statistiche per sede

Ritardi per sede

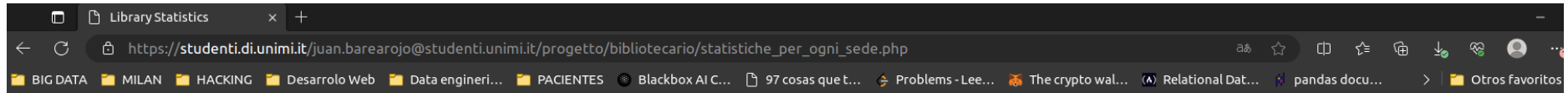
Go HOME



Loans Information

| Copy ID | ISBN | Title | Reader Fiscal Code | Reader Name | Expected Return Date | Actual Return Date |
|---------|---------------|--------------|--------------------|-------------------|----------------------|--------------------|
| 15 | 9781849700611 | Horus Rising | FC000009J | Jaghatai Khan | 2024-06-24 | |
| 20 | 9781849701021 | False Gods | FC000003R | Roboute Guilliman | 2024-05-02 | |

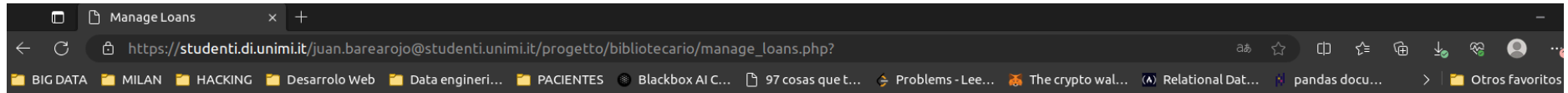
[Go HOME](#)



Library Statistics

| Total Copies | Total ISBNs | Total Loans in Progress |
|--------------|-------------|-------------------------|
| 33 | 30 | 6 |

[Go HOME](#)



Manage Loans

All loans

Extension of Loan

Return a copy

Go HOME

Found Loans

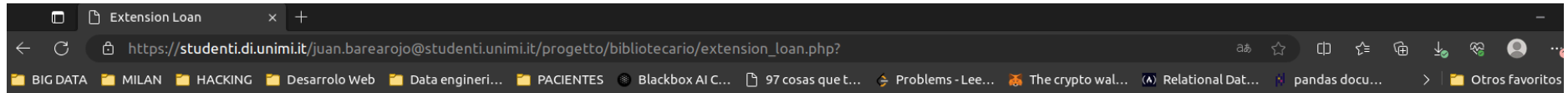
https://studenti.di.unimi.it/juan.bareaajo@studenti.unimi.it/progetto/bibliotecario/loan_table.php?

BIG DATAMILANHACKINGDesarrollo WebData engineri...PACIENTESBlackbox AI C...97 cosas que t...Problems - Lee...The crypto wal...Relational Dat...pandas docu...Otros favoritos

Loans Information

| Copy ID | Fiscal Code | Loan Date | Actual Return Date | Expected Return Date |
|---------|-------------|------------|--------------------|----------------------|
| 2 | 2 | 2024-06-24 | 2024-06-24 | 2014-07-14 |
| 1 | 2 | 2024-06-24 | 2024-06-24 | 2015-07-04 |
| 3 | 2 | 2024-06-24 | 2024-06-24 | 2024-07-04 |
| 15 | FC000009J | 2024-06-02 | | 2024-06-24 |
| 45 | 1 | 2024-06-25 | | 2024-07-05 |
| 4 | 2 | 2024-06-24 | | 2024-07-05 |
| 40 | FC000001E | 2024-06-25 | | 2024-07-05 |
| 59 | FC000001E | 2024-06-25 | | 2024-07-05 |
| 29 | 1 | 2024-06-25 | | 2024-07-07 |
| 5 | 2 | 2024-06-25 | 2024-06-25 | 2024-07-05 |
| 43 | FC000001E | 2024-06-03 | | 2024-06-13 |
| 20 | FC000003R | 2024-04-02 | | 2024-05-02 |
| 23 | 4 | 2024-05-02 | 2024-06-25 | 2024-05-13 |
| 64 | 2 | 2024-06-25 | | 2024-07-05 |
| 32 | FC000009J | 2001-05-02 | | 2002-05-13 |

Go HOME



Extension loan reader

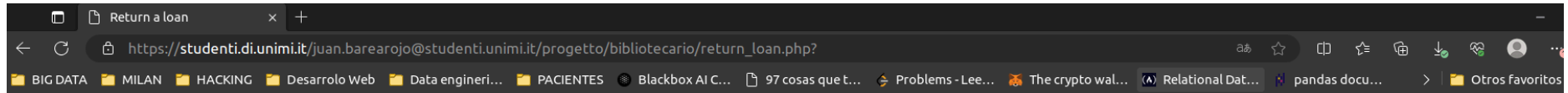
Fiscal code of the reader:

Copy ID:

Start Date:

Extension loan

Go HOME



Return a loan

Copy of id of the book you want to return:

Date where the loan was taken:

Return book

Go HOME