**1. Technical Documentation (For Developers)**

**1.1 Introduction**

This document describes the architecture of the library management system, the main components (mini-framework, controllers, models and database) and guides developers on how to configure the local environment to run the project.

**1.2 Mini-Framework Architecture**

The system uses a mini-framework based on the MVC (Model-View-Controller) pattern.

* **Folder Structure**:
  + /app/: Contains the main logic folders (Controllers, Models and Views), and the routing and database connection file.
    - /Controllers/: Manages HTTP requests and executes appropriate logic.
    - /Models/: Contains the logic for accessing and manipulating data.
    - /Views/: Files responsible for the presentation (HTML, CSS, JS).
  + /public/: contains public files such as index.php, CSS, JS and images.
  + /vendor/MF/: Implementation of the mini-framework (Loading routes and Controllers, Rendering view files and connecting to Model data files).
  + /test/: General files for testing controllers and models.
* **Request Flow**:
  + Requests arrive via public/index.php.
  + The router determines which controller and method to execute.
  + The controller uses models to access or manipulate data.
  + The response is rendered in the appropriate view.

**1.3 Controllers Used**

Implemented controllers:

* **LoginController**:
  + Methods:
    - index(): Displays the login screen and checks if “login” is passed in the URL.
* **AuthController**:
  + Methods:
    - Autenticar(): Validates that the login form is sent correctly, and which type of user wants to access the system.
    - redirect(): Support method to redirect http url.
    - Sair(): destroys the session where the user is logged in.
* **LivrosController**
  + Methods
    - Livros(): Renders the books screen, checks the user type and validates other methods used on the same screen.
    - recuperaUsuario(): retrieves a list of all users and their identifiers, which will be used to borrow books.
    - CadastrarLivros(): Renders the book registration screen.
    - registrarLivro(): checks the user's permission to access the screen, obtains the form fields and validates whether they are being sent correctly, if so, registers them in the database otherwise issues an error message.
    - deletarLivro(): Gets the Identifier of the book to be deleted and deleted from the database.
    - Emprestar(): Method for lending books by checking access permissions, and making the loan within 10 days of the current date, checks whether the user and the manager to make the loan choosing which student user will keep the book, if it is a common user doing the loan loan he just needs to choose the books.
    - solicitarDevolucao(): Method where the student requests the return of a book before or after the delivery deadline, and waits for the manager to confirm the return.
    - confirmarDevolução(): For the manager, as he has access to all users, he receives a return request and this method confirms the return, setting the field to true in the database, and making the books available again to other people.
    - getLivrosList(): Lists all books registered in the database.
* **UsuariosController**:
  + Methods:
    - Usuarios(): Renders the list of users, and lists all of them.
    - Inscreverse(): renders the user registration screen, and leaves the form fields blank.
    - Registrar(): validates whether the user registration form fields have all the information, if everything is correct, it saves the data in the database and redirects the user to the list screen of all users.
    - Deletarusuario(): method to delete the selected user using the identifier registered in their database.
    - Direcionar(): it checks if the user is logged in correctly, if so, it takes them to the dashboard, if not, it redirects them to the login screen with an error.
* **EmprestimoController**:
  + Methods:
    - Emprestimos(): Renders the list of loans, and lists them all.

**1.4 Models Used**

Models represent database tables and encapsulate operations such as searches and saves.

* **Models**:
  + **Livros**: Represents the livros table.
    - Main methods:
      * getLivros(): Searches all data from the books in the table.
      * salvar(): Save a new book or update an existing one.
      * Deletar(): Deletes a book from the table.
      * validarCadastro(): validates the number of characters sent to be registered in the table.
      * getIsbn(): valid if we do not have a duplicate Isbn in the table, avoiding being able to register the same Isbn twice.
      * atualizarDisponibilidade(): Changes the available field so that the book can be borrowed again.
      * getLivrosById(): retrieves the id of a book.
      * getLivrosFiltrados(): used to search for a specific book by its name.
  + **Usuários**: Represents the Usuarios table.
    - Main methods:
      * listUsuario(): Fetches all data from table users who are not managers.
      * salvar(): Save a new user or update an existing one.
      * Deletar(): Deletes a user from the table.
      * validarCadastro(): The number of characters sent to be registered in the table is valid.
      * cpfExistente(): valid if we do not have a duplicate CPF in the table, avoiding being able to register the same number twice.
      * getUsuariosPorEmailCPF(): valid if we do not have a duplicate Email in the table, avoiding being able to register the same email twice and the CPF.
      * autenticar(): used to authenticate login data when requested.
      * getUsuariosFiltrados(): used to search specific users by their name.
  + **Empréstimos**: Represents the emprestimos table.
    - Main methods:
      * getTodosEmprestimos(): Searches for all loan data from the table that has not yet been returned.
      * getEmprestimosFiltrados(): used to search for specific loans that have not been returned.
      * getEmprestimsoUsuarios(): identify the user who has borrowed the books.
      * registrarEmprestimo(): registers the loan
      * solicitarDevolução(): updates the requested\_return field to true.
      * registrarEmprestimos(): changes the available field to true and the request field to false to stop issuing alerts in views.
      * verificaEmprestimos(): Check whether the book is already borrowed or not.
      * verificaDevolucaoPendente(): checks which books are pending return to show alerts.

**1.5 Database**

* **General Structure**:
  + Main tables:
    - livros: Manages data from the main books.
    - emprestimos: Manages requested loans and returns.
    - usuarios: Stores user data and permissions.
* **Technology**: MySQL.
* **Settings**: The /App/Connection.php file contains the credentials for connection.

**1.6 How to Run the System Locally**

1. **Prerequisites**:
   * PHP 7.4+.
   * MySQL 5.7+.
   * Composer.
2. **Steps**:
   * Clone the repository:

**git clone <url-do-repositorio>**

* + Install dependencies:

**composer install**

* + Configure the database in /App/ Connection.php.

**Start an apache server and access Mysql, and import the database into it, just configure your credentials in the Connection.php file**

* + Start the server:

If you have git bash installed, go to the project's public folder and open git bash then run the command:

**php -S localhost:8080**

After that just access localhost:8080 in your browser.

If not, use the command prompt to go to the project's public folder and pass the above field to start the server.

**1.7 Screen patterns**

* Login

Estante com livros

Descrição gerada automaticamente

* Dashboard

Calendário

Descrição gerada automaticamente

* Books

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* RegisterBooks

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

* Users

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* RegisterUsers

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* Loans

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* LoansManager

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente